2020 3rd IEEE International Conference on Soft Robotics (RoboSoft)
Yale University, USA

Soft Robot Control With a Learned Differentiable Model

James M. Bern, Yannick Schnider, Pol Banzet, Nitish Kumar and Stelian Coros

Abstract— **Soft robots are inherently safe and comply readily to their environment. They are therefore exciting for applications like search and rescue or medicine, which involve a high degree of uncertainty, and require interacting with humans. However, the best way to model and control soft robots largely remains an open question. One promising approach is to leverage physically-based modeling techniques such as the finite element method. However, such techniques are inherently limited by their physical assumptions. Indeed, real-world soft robots are often made from unpredictable materials, using imprecise techniques. Data-driven approaches provide an exciting alternative, as they can learn real-world fabrication defects and asymmetries. In this paper we present our first investigation into using machine learning to do soft robot control. We learn a differentiable model of a soft robot's quasi-static physics, and then perform gradient-based optimization to find optimal open-loop control inputs. We find that our learned model captures phenomena that would be absent from an idealized physically-based simulation. We also present practical techniques for acquiring high-quality motion capture data, and observations the effect of network complexity on model accuracy.**

Index Terms— **Soft robots, Modeling, Control, Learning**

I. Introduction

Soft robotics promises to change the way people interact with robots in multiple areas such as search and rescue [1], entertainment [2], assistive robotics [3]–[5] and medical robotics [6]. Because they are inherently safe, and able to adapt to their surroundings, soft robots have the potential to interact more closely and organically with people. Much prior work has explored new soft materials and actuation technologies [7]–[9], including pneumatically-actuated silicone, and cable-driven foam [10].

However, the question of how to best model and control soft robots remains largely open. This is because soft materials exhibit highly nonlinear behavior, that is further complicated by imperfect fabrication and actuation techniques. A general methodology for modeling and controlling soft robots–especially one that can account for unforeseen variation in physical hardware–is therefore an especially exciting research challenge. Such a methodology also promises to be quite useful. Accurate simulation models of soft robots open the door to multiple tasks in optimal control such as trajectory tracking, trajectory optimization, and collision avoidance [11]–[13].

Previous work on controlling soft robots can be loosely divided into two general categories: those that make use of physically-based simulation (model-based), and those that do not (data-driven). Physically-based modeling includes simulation models based on e.g. elastic rods [14] or the finite element method [15]–[17]. Such techniques have been shown to be effective tools for controlling soft robots. However they are typically limited in their ability to account for unforeseen variations in physical hardware, though methods certainly exist to fit or tune such models to better match reality [18].

Data-driven control methods typically use machine learning or regression techniques to build a model from real-world sensor data. Specific techniques include e.g. Gaussian mixture models (GMMs) [19], Gaussian process regressions (GPR) [20], K-Nearest-Neighbors Regression (K-NNR) [21], and identification of an approximate Koopman operator [22]. An example of such an approach is the recent work of Thuruthel et al., which uses a recurrent neural network to sense the contact force and position of an inflatable robot using arbitrarily-placed redundant strain sensors [23]. Another example of such an approach is the work of Li et al., which uses a Kalman filter to estimate the manipulator Jacobian of a continuum robot on the fly [24].

Data-driven methods for soft robotics often directly learn a robot's inverse kinematics [20], [21], [25]–[30]. However soft robots may have redundant forward kinematics, meaning different control inputs can lead to the same end effector position. This makes the problem of inverse kinematics ill-suited to standard regression or supervised learning techniques [31]. This is because when multiple valid choices of control input exist, such techniques return an interpolation of these choices, which has no guarantee of itself being a good choice.

An approach called distal supervised learning (DSL) was developed to solve the problem of finding the inverse of a nonlinear many-to-one mapping [31], which is exactly the problem we face when controlling a redundant soft robot. Our approach falls under the umbrella of DSL, specifically the use of gradient-based optimization to search a learned forward model. DSL works by exploiting a forward model that is learned in a supervised fashion. This can be done either by inverting the learned forward model using another neural network [32], or by searching the learned forward model using gradient-based optimization [33]. Here we choose the latter strategy.

We show how to learn a model of a soft robot's forward kinematics, and how to search that model using gradient-based techniques. This allows us to accurately track open-loop trajectories. Additionally, we put our approach into context with our previous work on a model-based technique called *Soft IK* [17].

James M. Bern, Yannick Schnider, Pol Banzet, Nitish Kumar and Stelian Coros are with the Computational Robotics Lab in the Institute for Intelligent Interactive Systems (IIIS), ETH Zurich Switzerland. {jamesmbern@gmail.com, yannicks@ethz.ch, pol.banzet@protonmail.com, nitish.kumar@inf.ethz.ch, stelian.coros@inf.ethz.ch}

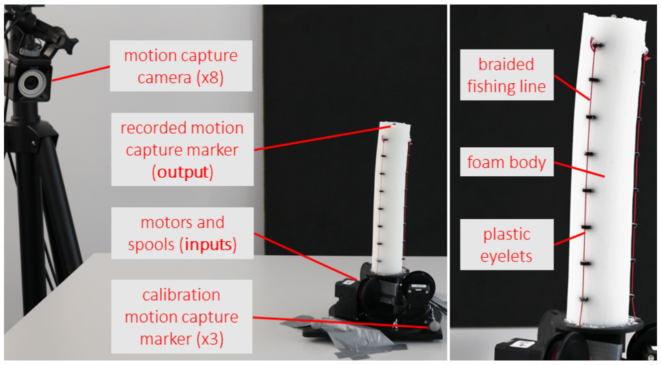
motion capture camera (x8)
recorded motion capture marker (output)
motors and spools (inputs)
calibration motion capture marker (x3)
braided fishing line
foam body
plastic eyelets

Fig. 1. We use motion capture to acquire training data for a three degree of freedom soft foam robotic arm.

II. Method

A. The Control Problem: Inverse kinematics

The traditional inverse kinematics problem asks the following question: Given a target end effector position $\boldsymbol{y}'$, what are optimal motor angles $\boldsymbol{u}^*$? In other words, how should we drive the robot's motors in order to bring its resulting end effector position as close as possible to the given target? This problem can be solved using numerical optimization, and is well-suited to gradient-based methods such as gradient descent or L-BFGS. For traditional rigid robots–which only rotate about prespecified joints–the relationship $\boldsymbol{y}(\boldsymbol{u})$ is analytic. This makes gradients straightforward to compute in closed form. Soft robots however, are quite a different story. They are free to exhibit continuous deformations across their entire bodies, and so more sophisticated methods are required to solve their inverse kinematics.

B. Prior Work: Soft IK, a sim-only method

One previously proposed solution to the soft robot inverse kinematics problem is called *Soft IK* [17]. This method leverages a simulation model based on the finite element method (FEM) to control a soft robot. We summarize the approach here. Given a target position $\boldsymbol{y}'$ for the soft robot, a control objective $\mathcal{O}(\boldsymbol{y}(\boldsymbol{u}))$ is written in terms of the robot's deformed pose. Optimal motor angles

$$\boldsymbol{u}^*(\boldsymbol{y}) = \arg\min_{\boldsymbol{u}} \mathcal{O}(\boldsymbol{y}(\boldsymbol{u}))$$

are then found by performing a gradient-based minimization. The difficulty is that for soft robots the relationship $\boldsymbol{y}(\boldsymbol{u})$ between control and state is typically not an analytic expression. For a soft robot modeled as a finite element mesh, $\boldsymbol{y}(\boldsymbol{u})$ is in fact computed by performing a highly nonlinear minimization (see Equation (2) in Section IV). However, by leveraging a powerful technique called *sensitivity analysis* we can relate motor angles to robot state through the nodal forces of the mesh, and exploit this relationship to compute the Jacobian $\frac{d\boldsymbol{x}}{d\boldsymbol{u}}$. This enables the use of a gradient-based minimizer to find optimal motor angles.

C. Proposed Method: Using a learned differentiable model

In this work we use a different approach. Rather than using a carefully-designed FEM simulation to guide our control, we instead use a learned differentiable model.

In their most general form, the forward kinematics of a soft robot are a map from motor angles $\boldsymbol{u}$ to end effector position $\boldsymbol{y}$, just like the forward kinematics equations of a traditional rigid robotic manipulator. We can learn this relationship in a supervised fashion. To do this, we train a small feedforward neural network to map from control inputs $\boldsymbol{u}$ to predicted corresponding quasi-static end effector position $\boldsymbol{y}(\boldsymbol{u})$. We train on data $\{(\boldsymbol{u}^{(i)}, \boldsymbol{y}^{(i)})\}_{i=1}^N$, acquired either in the real-world via motion capture (see Section III), or in simulation (see Section IV).

Once the net has been trained, we can use it to find optimal control inputs $\boldsymbol{u}^*$ that bring the network's prediction of robot end effector position $\boldsymbol{y}(\boldsymbol{u})$ as close as possible to the target position $\boldsymbol{y}'$. We minimize the control objective

$$\mathcal{O}(\boldsymbol{u}) = \frac{1}{2}\left\|\boldsymbol{y}(\boldsymbol{u}) - \boldsymbol{y}'\right\|^2 \qquad (1)$$

to find optimal control signals $\boldsymbol{u}^*$. We use gradient descent as our minimizer, and compute the gradient of the objective $\frac{d\mathcal{O}}{d\boldsymbol{u}} = (\boldsymbol{y}(\boldsymbol{u}) - \boldsymbol{y}')\frac{d\boldsymbol{y}}{d\boldsymbol{u}}$ using the chain rule. The quantity $\frac{d\boldsymbol{y}}{d\boldsymbol{u}}$ here is the *network Jacobian*, which relates changes in the network's input to changes in its output. In our particular case, the network Jacobian explains how changes in control input $\boldsymbol{u}$ affect changes in the predicted robot state $\boldsymbol{y}(\boldsymbol{u})$. We explain the Jacobian of a feedforward neural network in more detail in the appendix.

The use of a neural network as the model of forward kinematics has several advantages over FEM. First its use of real world data gives our controller knowledge of unknown imperfections and asymmetries in the real-world robot, which would be absent from an idealized simulation model. Additionally, instead of having to perform a costly energy minimization to evaluate the quality of each new candidate control vector, we must only evaluate a small neural net.

Finally, we note that this overall approach works by finding control signals that are optimal *according to the neural network*. Therefore its success hinges on this net providing accurate predictions of the robot's real-world physics. If the net itself is inaccurate, then the overall control method will fail hopelessly when applied in reality. However if given a properly trained net, the presented method proves to be quite accurate. Quantitative results are shown in Section V.

D. Open-loop trajectory following

The control method presented above is formulated to reach a single target position, but can also be applied to do quasi-static trajectory following. Given a sequence of target positions, the objective in Equation (1) is minimized repeatedly for each subsequent target position. To improve convergence, the solution for a given target position can be used to warm-start the optimization for the next target. This simple method proves quite powerful, and is used to generate the results in Section V.

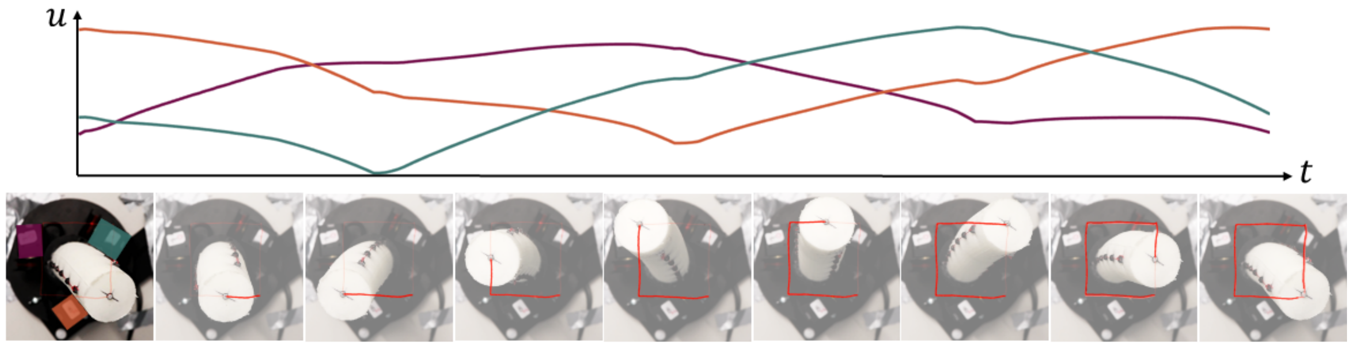

Fig. 2. A three-degree of freedom soft robot following a square-shaped target end-effector trajectory using motor angles $\boldsymbol{u}$ found using the method in this paper. Motors are colored according to their corresponding control input trajectory.

III. Implementation details

Our first goal is to learn a quasi-static model of the soft robot's physics. In order to train a suitable neural network, we need data pairs of the form $(\boldsymbol{u}^{(i)}, \boldsymbol{y}^{(i)})$. Data acquisition is in theory simple: We set the motors to angles $\boldsymbol{u}^{(i)}$, read the resulting end effector position $\boldsymbol{y}^{(i)}$ from motion capture, and repeat for the next vector of motor angles. However the reality of capturing high-quality data is more involved.

The soft robots used in this work are made from flexible foam. This material has the advantages that it is light and easy to deform. However, the foam has some other properties that make it somewhat difficult to capture training data for a quasi-static model. If we contract and release a cable, we observe the following behavior: 1) the motion of the robot lags behind the action of releasing the cable, and 2) the robot never quite returns to its initial position (preferring to remain slightly bent). The robot therefore exhibits both viscous effects and hysteresis. This is due to the material properties of the foam, as well as friction.

We employ two tricks to ensure that our sampling is not corrupted by these effects. First to combat the viscous effect, we simply wait until the robot has stopped moving before recording its position. This can be done in quite a reliable fashion, since the robot is already inside a motion capture setup: we wait until the robot's current end effector position is approximately equal to the average of the previous twenty end effector positions. Next, to combat hysteresis we perform a simple routine to erase the physical system's memory before taking each sample: We contract all cables together by some nominal value, and then release them. We observe in practice for the three-actuator arm in section V that this technique effectively returns the robot to a neutral position.

We make a final note on the trade-off between quality and speed when it comes to data capture. The techniques presented in this section capture data much more slowly than e.g. sweeping the robot through its workspace while continually recording sample points, but they are also quite reliable. For the example in this paper, we were able to acquire a sufficient amount of high-quality data to train an accurate network.

IV. Simulated data

The method in this paper does not require a simulation, as the neural network can learn the forward kinematics purely from motion capture data. Motion capture proves to be incredibly useful for learning hard-to-model effects like friction, or accounting for the minute asymmetries of imperfect fabrication. However, motion capture also has an inherent disadvantage: it is slow. In practice the procedure described in the previous section captures real-world data at a rate of approximately 300 samples per hour. For robots with a high number of actuators, this makes it impossible to densely sample the control space. Not only would the motion capture simply take too long, but over the course of the capture process the robot's viscoelastic behavior would likely change. This means that even if such a huge number of samples could be acquired, the samples acquired earlier on might no longer be relevant.

However, there is an alternative approach. We can leverage numerical simulation to acquire some or all of our data. We discretize the robot into tetrahedral neo-Hookean finite elements, following the same method as [34], which models cables as one-sided quadratic springs running through frictionless via points. The total deformation energy of the system $E(\boldsymbol{x}, \boldsymbol{u})$ is written in terms of the mesh's nodal positions $\boldsymbol{x}$ and motor angles $\boldsymbol{u}$. For a given vector of motor angles $\boldsymbol{u}$, the corresponding statically stable pose

$$\boldsymbol{x}(\boldsymbol{u}) = \arg\min_{\boldsymbol{x}} E(\boldsymbol{x}, \boldsymbol{u}) \tag{2}$$

is found numerically, by minimizing the total energy of the system. While this idealized simulation will not capture any unanticipated effects or asymmetries, generating simulated data is far faster than using motion capture (in part because we do not need to employ the strategies from Section III, as our simulation does not include viscosity or hysteresis).

Care must be taken to match the simulation to the spatial position and orientation of the physical robot, in addition to the robot's physical dimensions and actuator layout. Once the simulation is set up though, data can be captured quite quickly, even in parallel or on multiple computers if necessary. We use simulated data to train the neural network model of the six-cable arm in Section V-B.

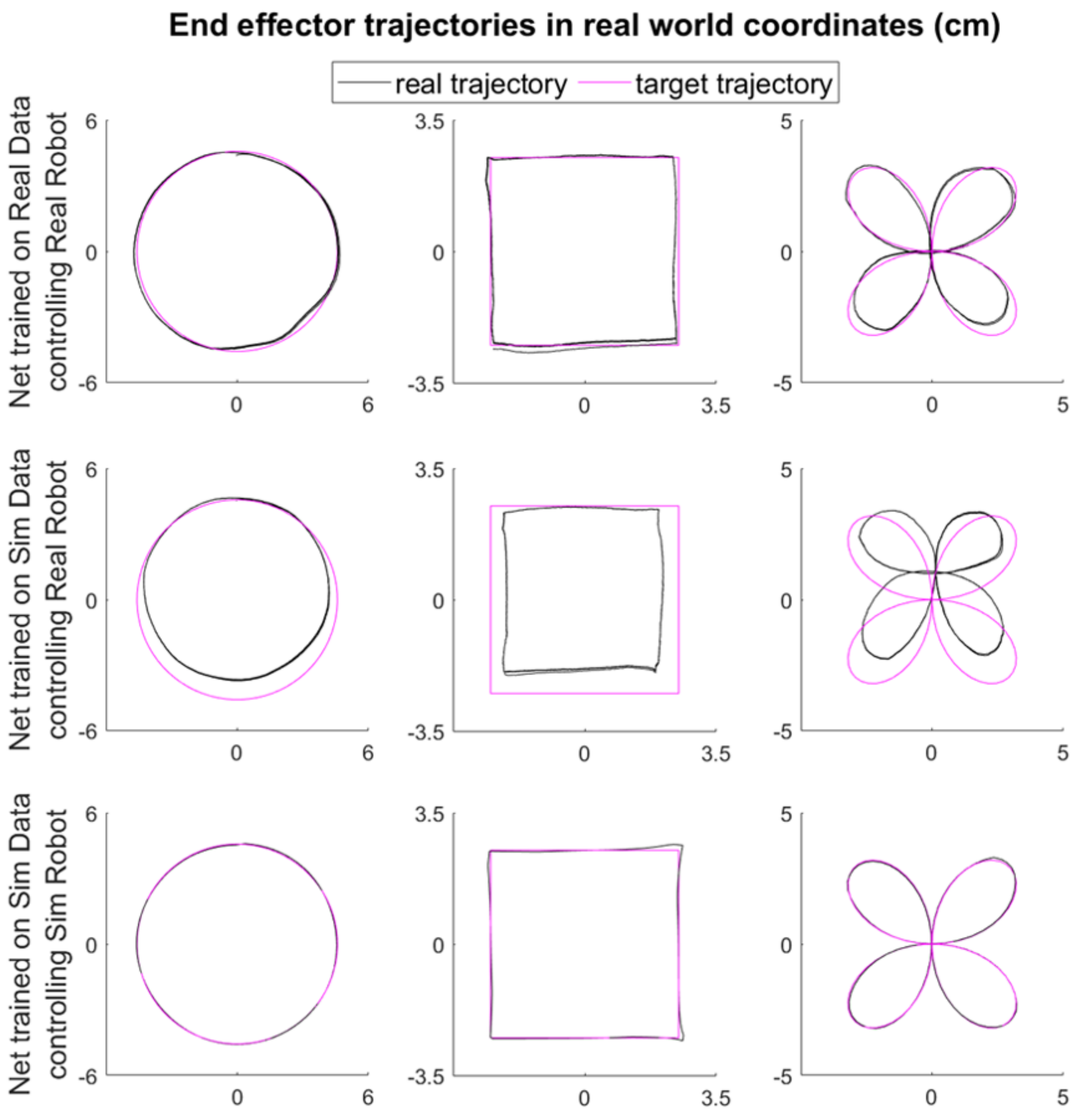

Fig. 3. Trajectory following for the three-cable robot. We show the real trajectories projected onto the plane for visual clarity, but note that the trajectories are captured and evaluated in 3D.

V. Results

A. Three-cable soft robotic arm

We control the three-cable foam robot shown in Figure 1 to follow various trajectories. This robot is a continuum soft cylinder measuring 20 cm in length and 4 cm in diameter.

1) Fabrication: The body of the robot was cast from expandable polyurethane foam (FlexFoam-iT!^{TM}III from Smooth-On Inc), using the same technique as [10] and [34]. Small plastic eyelets were shallowly embedded in the foam during casting, and braided fishing line cable routed through them. Cables are pulled onto spools by geared servomotors (Dynamixel XM-430's), which causes the robot to bend.

A single motion capture marker is affixed to the tip of the robot, and three additional markers are attached to its base to establish the ground plane. Finally, the robot is secured to the center of a motion capture setup, consisting of eight Optitrack Prime 13 cameras.

2) Training: We sampled our training data from a $7\times7\times7$ grid in actuator space, with each axis between 0 (for no contraction) and $\overline{u}$ (a nominal max motor angle that struck a balance between having a large workspace and not damaging the robot). This gave us 343 equally-spaced sample points, of which we discarded 35 as too aggressive to be safe for the robot. To generate our training data, we sent the remaining 308 control signals to the robot following the procedure in Section III, and recorded the resulting end effector positions. Using the MATLAB Deep Learning Toolbox, we trained a feedforward neural network on the entire gathered data set. The overall network had three inputs, one for each motor angle, and three outputs, one for each spatial degree of freedom of the tip marker. Additionally the network had two hidden layers, containing 20 neurons each, with sigmoid activation functions. This neural net (trained on real data) was used to create the first row of Figure 3 and Table I.

We repeated the same process with data collected on the simulated robot, to train another (totally separate) network. This neural net (trained on simulated data) was used to create rows 2 and 3 of Figure 3 and Table I.

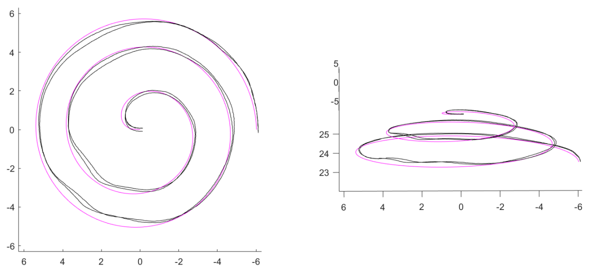

Fig. 4. Using a net trained on real data to control the real three-cable robot arm to follow a 3D helical target. The target trajectory is shown in pink.

3) Evaluation: To evaluate the degree of accuracy to which an FNN can model a soft robot's physics, we ran the following test. We split the real-world data into a training set and a test set (90% and 10% of the data respectively), and trained an FNN with the same architecture as the one used in the previous section ten times with different random data splits. The average mean squared error on the test set was $1.53 \times 10^{-5}\ mm^2$. The average Euclidean distance between the predicted and measured tip position was found to be $4.42\ mm$, corresponding to an average relative error of approximately 1.6% of the robot's workspace.

To evaluate the performance of our overall control method we ran several tasks of the trajectory following type described in Section II-D. The resulting end effector trajectories are shown in Figures 3 and 4, and quantitative data on accuracy is presented in Table I. Each row of Figure 3 and Table I corresponds to a particular condition (e.g. using a net trained on simulated data to control a real robot). Note that each target trajectory (circle, rose, square) is the same across conditions.

TABLE I
MEAN DISTANCE FROM TARGET IN CM FOR TRAJECTORY FOLLOWING TASKS SHOWN IN FIGURE 3.

	circle	square	rose
Real Data + Real Robot	0.15	0.14	0.22
Sim Data + Real Robot	0.68	0.62	0.92
Sim Data + Sim Robot	0.05	0.06	0.04

The first row of Figure 3 shows our primary result. Using the net trained on real data to control the real robot achieves mean absolute tracking error of between 0.15 cm and 0.22 cm depending on the task. The target trajectories in question are around 6 cm to 10 cm across, so this error corresponds to between 1.5% and 3.7% of the size of the trajectory.

The second row shows what happens when we control the real robot using a network trained on data from the idealized simulation described in Section IV. This method succeeds in matching the general trajectory, but is much less accurate, at around 0.62 cm to 0.92 cm mean absolute tracking error. We make note of two important qualitative discrepancies between the target trajectory and the trajectory in the second row (Sim Data + Real Robot). The first is scale. The real world trajectory is substantially smaller than the target. This suggests that the real robot is not as stiff as the simulation. This discrepancy could likely be reduced with more accurate finte element modeling (perhaps using a denser finite mesh element mesh, or running system identification to better estimate the material parameters). The second, more interesting discrepancy is translation. The real world trajectory is shifted up along the vertical. The simulation model expects a symmetric robot, but these trajectories reveal that this is not the case in reality! The action of the real world robot is apparently biased upwards, likely due to imprecise fabrication techniques. The network trained on real data manages to learn this physical asymmetry, giving the controller the ability to compensate. The network trained on simulated data has no knowledge of this asymmetry, leading to the vertical offset we see in the second row.

The third row shows the result of controlling the simulated robot using a network trained on simulated data. Tracking error is quite low at around half a millimeter. This serves to confirm that the network trained on simulated data does in fact accurately model the physics of the simulation.

B. Simulated six-cable soft robotic arm

To explore how our method scales, we built a simulation of a six-cable robotic arm. This robot has three pairs of cables, each consisting of one long cable and one short cable, both routed along the same path starting from the robot's base. The use of simulation enables us to quickly prototype the robot's design, as well as its controller. We train a feedforward neural network with three hidden layers of 25 neurons each on 15414 samples of simulated data. We use this network to perform various trajectory following tasks, some of which are shown in Figure 5. Please see our supplementary video for additional tasks and animations.

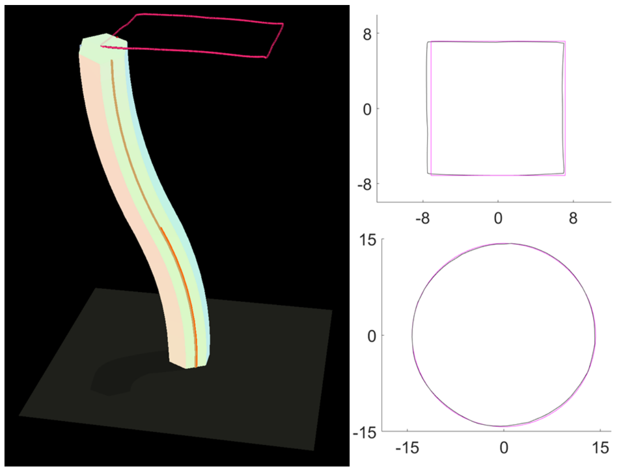

Fig. 5. Trajectory following task for the two-stage, 6 tendons simulated robot. Plots in cm, target in black, trajectory in pink.

VI. Discussion and Future Work

A. Network Complexity

To start understanding the effect of the network complexity on model accuracy, we varied the number of neurons per hidden layer while keeping the total number of hidden layers constant at two. For each such architecture we performed the following test ten times: We randomly split the data into training, test, and validation sets (80%, 10% and 10% of the data respectively), and trained until reaching either a stopping criterion on the validation set (performance on validation set allowed to decrease for maximum of 6 epochs), or on the gradient (magnitude less than 10^{-6}). We plot the average absolute distance between the predicted and real tip position against the number of neurons per hidden layer in Figure 6.

We observe the following trend. Very small networks perform poorly, likely because they are not expressive enough to capture the robot's physics. Increasing the size of the network yields greater accuracy, as the network now has sufficient expressive power. However, past a certain size we see model accuracy decrease again, which we hypothesize is because the data set we gathered was insufficiently large to properly train such a large network train.

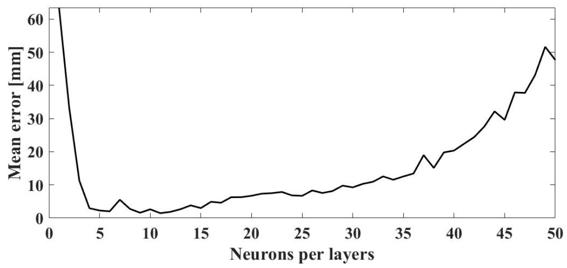

Fig. 6. Plot showing the observed trend between network complexity and accuracy. Smaller networks are observed to be insufficiently expressive, and larger networks seem to require more training data than we had available.

B. Efficient Sampling

Our control method relies on the trained neural network being an accurate model of the robot's physics. In practice, we find that one reliable way to train an accurate network is to sample data from a dense grid in actuator space. However, as the number of actuators increases, we will face the curse of dimensionality. This motivates the exploration of more efficient sampling methodologies.

The specific number of samples we chose (i.e. the density of the grid we sample from) could likely be reduced without a large impact on the accuracy of the pipeline. It would be interesting to explore the trade-off of sampling density and model accuracy in depth, and determine exactly how sparsely we can sample while still obtaining sufficiently accurate results. It is also very likely that some network architectures require less data to achieve accurate results, and this could be investigated in conjunction with sampling density. It could also be interesting to explore hybrid training strategies that use both motion capture and simulated data to achieve accurate results with limited real-world sampling. Such an approach might first train a network on a large amount of simulated data, and then fine-tune on real data.

The data gathering routine could also be sped up in the future. The rate-limiting step is the time we spend waiting for the viscous effects of the foam to disappear. To avoid this problem entirely we could explore the use of less viscous elastomers. We could also explore network architectures capable of learning the viscous behavior.

C. Actuation strategies and dynamic motions

In this work we focused specifically on building a quasi-static model of a soft foam robot's physics, for which we found a simple feed-forward neural network to be a sufficient function approximator. An exciting avenue of future research is to expand the space of soft robots we can reliably control.

We could apply this same general methodology to other breeds of hard-to-model systems, such as soft fluidic elastomer robots. In principle our method applies to any robot where a number of control inputs lead to an observable output state, however the specifics of the implementation–including data gathering, and network training–might have to be modified. Another exciting direction would be to learn and plan dynamic motions. This would likely require more sophisticated neural networks–capable of capturing a robot's dynamics and viscoelasticity [35]–as well as methods for harnessing such networks to do control.

VII. Conclusion

In this work we showed how to control a soft robot using a learned differentiable model. We gathered data from either real-world motion capture or finite element simulation, and trained a neural network to predict a soft robot's quasi-static physics. We defined a suitable control objective in terms of this neural network's output, and computed its gradient using the network Jacobian. We performed a gradient-based optimization to do open-loop trajectory following, and ran our method on a real-world cable-driven foam robot, achieving mean absolute error of only one or two millimeters. The overall method we present is general. It provides an effective framework for learning a soft robot's physics, and leveraging this knowledge to do control, even in the face of unanticipated fabrication defects and asymmetries.

Appendix

The network Jacobian of a feedforward neural network

We use a feedforward neural network to map from control input $\boldsymbol{u}$ to predicted robot end effector position $\boldsymbol{y}$. The control input forms the input layer $\boldsymbol{a}_0 = \boldsymbol{u}$, and the predicted robot end effector position forms the output layer $\boldsymbol{a}_n = \boldsymbol{y}$. In between the input and output layers, there are $n-2$ *hidden layers*, which we denote $\boldsymbol{a}_1, ..., \boldsymbol{a}_{n-1}$. The *activation* of the network's i-th layer is defined to be $\boldsymbol{a}_i = \sigma_i(\boldsymbol{w}_i^T \boldsymbol{a}_{i-1} + \boldsymbol{b}_i)$, where σ_i is a differentiable activation function, and $\boldsymbol{w}_i$ and $\boldsymbol{b}_i$ are the i-th layer's *weights* and *biases* respectively. The overall map from input to output is differentiable, and so we can apply the chain rule to yield the network Jacobian.

$$\frac{d\boldsymbol{y}}{d\boldsymbol{u}} = \frac{d\boldsymbol{a}_n}{d\boldsymbol{a}_0} = \frac{d\boldsymbol{a}_n}{d\boldsymbol{a}_{n-1}} \frac{d\boldsymbol{a}_{n-1}}{d\boldsymbol{a}_{n-2}} \frac{d\boldsymbol{a}_{n-2}}{d\boldsymbol{a}_{n-3}} \cdots \frac{d\boldsymbol{a}_1}{d\boldsymbol{a}_0}$$

References

[1] M. M. Tanouye and V. Vikas, "Optimal learning and surface identification for terrestrial soft robots," in *2018 IEEE International Conference on Soft Robotics*, Apr. 2018, pp. 443–448.

[2] J. Lee, J. Eom, W. Choi, and K. Cho, "Soft LEGO: Bottom-Up Design Platform for Soft Robotics," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Oct. 2018, pp. 7513–7520.

[3] M. Manti, A. Pratesi, E. Falotico, M. Cianchetti, and C. Laschi, "Soft assistive robot for personal care of elderly people," in *2016 6th IEEE International Conference on Biomedical Robotics and Biomechatronics*, June 2016, pp. 833–838.

[4] G. Lee, Y. Ding, I. G. Bujanda, N. Karavas, Y. M. Zhou, and C. J. Walsh, "Improved assistive profile tracking of soft exosuits for walking and jogging with off-board actuation," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Sept. 2017, pp. 1699–1706.

[5] E. C. Goldfield, Y.-L. Park, B.-R. Chen, W.-H. Hsu, D. Young, M. Wehner, D. G. Kelty-Stephen, L. Stirling, M. Weinberg, D. Newman, R. Nagpal, E. Saltzman, K. G. Holt, C. Walsh, and R. J. Wood, "Bio-Inspired Design of Soft Robotic Assistive Devices: The Interface of Physics, Biology, and Behavior," *Ecological Psychology*, vol. 24, no. 4, pp. 300–327, Oct. 2012.

[6] J. Burgner-Kahrs, D. C. Rucker, and H. Choset, "Continuum Robots for Medical Applications: A Survey," *IEEE Transactions on Robotics*, vol. 31, no. 6, pp. 1261–1280, Dec. 2015.

[7] K.-J. Cho, J.-S. Koh, S. Kim, W.-S. Chu, Y. Hong, and S.-H. Ahn, "Review of manufacturing processes for soft biomimetic robots," *International Journal of Precision Engineering and Manufacturing*, vol. 10, no. 3, pp. 171–181, July 2009.

[8] F. Schmitt, O. Piccin, L. Barbé, and B. Bayle, "Soft Robots Manufacturing: A Review," *Frontiers in Robotics and AI*, vol. 5, 2018.

[9] C. Laschi, B. Mazzolai, and M. Cianchetti, "Soft robotics: Technologies and systems pushing the boundaries of robot abilities," *Science Robotics*, vol. 1, no. 1, p. eaah3690, Dec. 2016.

[10] L. Somm, D. Hahn, N. Kumar, and S. Coros, "Expanding Foam as the Material for Fabrication, Prototyping and Experimental Assessment of Low-Cost Soft Robots With Embedded Sensing," *IEEE Robotics and Automation Letters*, vol. 4, no. 2, pp. 761–768, Apr. 2019.

[11] C. Shin, P. W. Ferguson, S. A. Pedram, J. Ma, E. P. Dutson, and J. Rosen, "Autonomous Tissue Manipulation via Surgical Robot Using Learning Based Model Predictive Control," in *2019 International Conference on Robotics and Automation*, May 2019, pp. 3875–3881.

[12] M. T. Gillespie, C. M. Best, E. C. Townsend, D. Wingate, and M. D. Killpack, "Learning nonlinear dynamic models of soft robots for model predictive control with neural networks," in *2018 IEEE International Conference on Soft Robotics*, Apr. 2018, pp. 39–45.

[13] T. G. Thuruthel, E. Falotico, F. Renda, and C. Laschi, "Model-Based Reinforcement Learning for Closed-Loop Dynamic Control of Soft Robotic Manipulators," *IEEE Transactions on Robotics*, vol. 35, no. 1, pp. 124–134, Feb. 2019.

[14] N. N. Goldberg, X. Huang, C. Majidi, A. Novelia, O. M. O'Reilly, D. A. Paley, and W. L. Scott, "On planar discrete elastic rod models for the locomotion of soft robots," *Soft robotics*, 2019.

[15] M. Thieffry, A. Kruszewski, C. Duriez, and T. Guerra, "Control Design for Soft Robots Based on Reduced-Order Model," *IEEE Robotics and Automation Letters*, vol. 4, no. 1, pp. 25–32, Jan. 2019.

[16] C. Duriez, "Control of elastic soft robots based on real-time finite element method," in *IEEE International Conference on Robotics and Automation*. IEEE, 2013, pp. 3982–3987.

[17] J. M. Bern, G. Kumagai, and S. Coros, "Fabrication, modeling, and control of plush robots," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2017, pp. 3739–3746.

[18] D. Hahn, P. Banzet, J. M. Bern, and S. Coros, "Real2sim: Visco-elastic parameter estimation from dynamic motion," *ACM Transactions on Graphics*, vol. 38, no. 6, pp. 1–13, 2019.

[19] H. Wang, J. Chen, H. Y. K. Lau, and H. Ren, "Motion Planning Based on Learning From Demonstration for Multiple-Segment Flexible Soft Robots Actuated by Electroactive Polymers," *IEEE Robotics and Automation Letters*, vol. 1, no. 1, pp. 391–398, Jan. 2016.

[20] G. Fang, X. Wang, K. Wang, K. Lee, J. D. L. Ho, H. Fu, D. K. C. Fu, and K. Kwok, "Vision-Based Online Learning Kinematic Control for Soft Robots Using Local Gaussian Process Regression," *IEEE Robotics and Automation Letters*, vol. 4, no. 2, pp. 1194–1201, Apr. 2019.

[21] J. Chen and H. Y. K. Lau, "Learning the inverse kinematics of tendon-driven soft manipulators with K-nearest Neighbors Regression and Gaussian Mixture Regression," in *2016 2nd International Conference on Control, Automation and Robotics*, Apr. 2016, pp. 103–107.

[22] D. Bruder, B. Gillespie, C. D. Remy, and R. Vasudevan, "Modeling and control of soft robots using the koopman operator and model predictive control," *arXiv preprint arXiv:1902.02827*, 2019.

[23] T. G. Thuruthel, B. Shih, C. Laschi, and M. T. Tolley, "Soft robot perception using embedded soft sensors and recurrent neural networks," *Science Robotics*, vol. 4, no. 26, p. eaav1488, 2019.

[24] M. Li, R. Kang, D. T. Branson, and J. S. Dai, "Model-free control for continuum robots based on an adaptive kalman filter," *IEEE/ASME Transactions on Mechatronics*, vol. 23, no. 1, pp. 286–297, 2017.

[25] M. Kovandžić, V. Nikolić, M. Simonović, I. Ćirić, and A. Al-Noori, "Soft robot positioning using artificial neural net," *Facta Universitatis, Series: Automatic Control and Robotics*, vol. 18, no. 1, pp. 019–030, Sept. 2019.

[26] H. T. Kalidindi, T. G. Thuruthel, C. Laschi, and E. Falotico, "Cerebellum-inspired approach for adaptive kinematic control of soft robots," in *2019 2nd IEEE International Conference on Soft Robotics*, Apr. 2019, pp. 684–689.

[27] F. Holsten, M. P. Engell-Nørregård, S. Darkner, and K. Erleben, "Data Driven Inverse Kinematics of Soft Robots using Local Models," in *2019 International Conference on Robotics and Automation*, May 2019, pp. 6251–6257.

[28] H. Jiang, Z. Wang, X. Liu, X. Chen, Y. Jin, X. You, and X. Chen, "A two-level approach for solving the inverse kinematics of an extensible soft arm considering viscoelastic behavior," in *2017 IEEE International Conference on Robotics and Automation*, May 2017, pp. 6127–6133.

[29] K.-H. Lee, D. K. Fu, M. C. Leong, M. Chow, H.-C. Fu, K. Althoefer, K. Y. Sze, C.-K. Yeung, and K.-W. Kwok, "Nonparametric Online Learning Control for Soft Continuum Robot: An Enabling Technique for Effective Endoscopic Navigation," *Soft Robotics*, vol. 4, no. 4, pp. 324–337, Aug. 2017.

[30] M. Giorelli, F. Renda, M. Calisti, A. Arienti, G. Ferri, and C. Laschi, "A two dimensional inverse kinetics model of a cable driven manipulator inspired by the octopus arm," in *2012 IEEE International Conference on Robotics and Automation*. IEEE, 2012, pp. 3819–3824.

[31] M. I. Jordan and D. E. Rumelhart, "Forward models: Supervised learning with a distal teacher," *Cognitive science*, vol. 16, no. 3, pp. 307–354, 1992.

[32] A. Melingui, O. Lakhal, B. Daachi, J. B. Mbede, and R. Merzouki, "Adaptive neural network control of a compact bionic handling arm," *IEEE/ASME Transactions on Mechatronics*, vol. 20, no. 6, pp. 2862–2875, 2015.

[33] S. Vijayakumar, "Machine learning & sensorimotor control lecture xi - learning with distal teachers (forward and inverse models)," http://www.inf.ed.ac.uk/teaching/courses/mlsc/Notes/Lecture11/MLSC_Lec11.pdf, February 2007.

[34] J. Bern, P. Banzet, R. Poranne, and S. Coros, "Trajectory optimization for cable-driven soft robot locomotion," in *Proceedings of Robotics: Science and Systems*, FreiburgimBreisgau, Germany, June 2019.

[35] S. Han, T. Kim, D. Kim, Y. Park, and S. Jo, "Use of Deep Learning for Characterization of Microfluidic Soft Sensors," *IEEE Robotics and Automation Letters*, vol. 3, no. 2, pp. 873–880, Apr. 2018.