Co-Learning of Task and Sensor Placement for Soft Robotics

Andrew Spielberg, Alexander Amini, Lillian Chin, Wojciech Matusik, and Daniela Rus

***Abstract*—Unlike rigid robots which operate with compact degrees of freedom, soft robots must reason about an infinite dimensional state space. Mapping this continuum state space presents significant challenges, especially when working with a finite set of discrete sensors. Reconstructing the robot's state from these sparse inputs is challenging, especially since sensor location has a profound downstream impact on the richness of learned models for robotic tasks. In this work, we present a novel representation for co-learning sensor placement and complex tasks. Specifically, we present a neural architecture which processes on-board sensor information to learn a salient and sparse selection of placements for optimal task performance. We evaluate our model and learning algorithm on six soft robot morphologies for various supervised learning tasks, including tactile sensing and proprioception. We also highlight applications to soft robot motion subspace visualization and control. Our method demonstrates superior performance in task learning to algorithmic and human baselines while also learning sensor placements and latent spaces that are semantically meaningful.**

***Index Terms*—Soft robot materials and design, soft sensors and actuators, modeling, control, and learning for soft robots, deep learning methods.**

I. Introduction

RECENT efforts in soft robotics research have led to breakthroughs in soft robot modeling and design [1]–[3]. Most of these victories, however, have relied on virtual simulation environments where full state information of a system's dynamics is visible to a controller. However, in the physical world, soft robots' continuum bodies are high/infinite dimensional. Full state information can thus only be achieved by sensorizing every point on the body, which is both impossible to manufacture and computationally intractable for efficient processing of downstream tasks. Although several approaches have been proposed for efficient reasoning about the high-dimensional state of soft robots [4], [5], they rely on *extrinsic* information to estimate robot state in a global reference frame, such as external cameras or motion capture devices. In order to realize the dream of fully untethered soft robots, *intrinsic*, on-board information must be used.

To address this need for intrinsic soft robotic modeling, we focus on the problem of co-learning optimal sensor placements and models for general supervised tasks. Different locations on a soft robot experience different dynamic responses during motion; thus, sensor distribution can have a profound impact on a robot's ability. Better understanding the impacts of sensor placement can create the rich representation of soft bodies needed for complex modeling and control tasks.

We propose a neural architecture for simultaneously learning soft robotic tasks and the optimal sensor placement for that task. Our model relies purely on intrinsic measurements — specifically, strains and strain rates — and is amenable to physical realization through off-the-shelf sensors. Since many soft robot representations are nodal in nature, we propose a novel architecture which adopts existing work in point-cloud-based learning and probabilistic sparsification. Our method treats sensor design as the dual of learning, combining physical and digital design in a single end-to-end training process.

Although computational sensing is traditionally thought of as an algorithmic problem, the choice of sensor placements has vital ramifications on hardware as well. For many sensor fabrication methods, such as directly embedding strain sensors into a solid elastic substrate, physical sensor placement is immutable and often laborious. Even in cases where sensor design allows for reconfiguration, sensorizations robust to a wide variety of environments or state distributions are still needed in order to maximize deployment time without expensive human-in-the loop hardware modifications.

To our knowledge, ours is the first method for *general* task learning for *dynamic* soft robots while optimizing sensor placement, focusing on simulated soft robots. We contribute the following: 1) A neural architecture for reasoning about soft robot state through measured strains and strain rates. 2) A probabilistic sensor representation suited to sparsification to a minimal set for downstream tasks, and a co-design algorithm for achieving this which outclasses both automated and human baselines. 3) Demonstrations of task learning and sensor placement co-design in two applications — tactile sensing and proprioception — across seven soft robot morphologies. The representations learned from these tasks can be applied to unlock a novel task — closed-loop computational control of soft walking robots from solely intrinsic sensor measurements, and no external sensors.

Manuscript received October 23, 2020; accepted January 5, 2021. Date of publication February 2, 2021; date of current version February 17, 2021. This letter was recommended for publication by Associate Editor H. Hauser and Editor C. Laschi upon evaluation of the reviewers' comments. This work was supported in part by National Science Foundation under Grant 1138967, IARPA Grant 2019-1902010000, in part by the National Science Foundation Graduate Research Fellowship under Grant 1122374, and in part by Fannie, and John Hertz Foundation. *(Andrew Spielberg and Alexander Amini contributed equally to this work.) (Corresponding author: Alexander Amini.)*

The authors are with the MIT Computer Science and Artificial Intelligence Laboratory (CSAIL), Massachusetts Institute of Technology, Cambridge, Massachusetts 02141 USA (e-mail: aespielberg@csail.mit.edu; amini@mit.edu; ltchin@mit.edu; wojciech@csail.mit.edu; rus@csail.mit.edu).

This article has supplementary downloadable material available at https://doi.org10.1109/LRA.2021.3056369, provided by the authors.

Digital Object Identifier 10.1109/LRA.2021.3056369

Although our architecture is general to supervised learning tasks with nodal inputs, we demonstrate its utility to important soft robotic problems, such as in-hand sensing and full state reconstruction. Despite the fact that these problems have not been successfully handled with manually specified sensor locations, we are able to demonstrate success on a wide range of topologies of fully dynamic soft robots operating on intrinsic sensor data.

II. Related Work

a) Automated sensor placement: Task-optimal sensor placement has been extensively studied in robotics , typically in the context of vision [6]–[8]. These works focus on optimal placement of cameras for tasks such as target detection, but do not involve learning. Classical methods for co-design of sensor placement and learning models come in two flavors. The first employs a heuristic search for sensor placement as an outer loop to an inner learning problem [9], which wraps a genetic algorithm around learning. Such a search would be prohibitively expensive for deep learning approaches. The second exploits specific problem structure to formulate an efficient co-optimization problem, as in [10]. Unfortunately, these approaches are task specific and are not applicable to complex soft robotics problems presented, nor general task learning. Morphological sensorimotor optimization has also been explored in evolutionary robotics [11], but in the context of rigid robots.

The problem of optimally sensorizing soft robots is less explored. [12] demonstrated curvature sensor placement algorithms for kinematic reconstruction of a soft arm's pose. [13] demonstrated algorithms for the placement of piezoresistive sensors on soft structures for static pose reconstruction. [14] demonstrated an algorithm which minimized sensor placement for static pose reconstruction; however, their method was specific to a specialized sensor type that is only amenable to soft arms and tendrilic structures. Furthermore, these preceding works are focused on static systems reconstructing a specific pose, a far cry from the goal of dynamic soft robotic systems.

Outside of soft robotics, task-driven co-learning of task and sensor design has been explored in computer vision and signal processing. For example, [15], [16] examined camera sensor design for imaging performance in specific domains. [17] presented a differentiable machine learning model for simultaneously sparsifying over beacon location and learning neural representations for RF localization tasks. Although this work is similar in approach, our method relies on a drastically different, point-cloud-based architecture in order to learn in challenging intrinsic material coordinates (rather than the extrinsic coordinates of RF localization, which can exploit a simpler, more general multilayer-perceptron (MLP) architecture), and uses an adaptive sparsifying layer with a carefully chosen regularizer rather than viewing sensor selection as a multicategorical selection.

b) Computational intrinsic sensing for soft robotics: Although the majority of the literature on computational soft robot reasoning has abstracted away sensing / representation through piecewise approximations or external motion capture systems, there is some work on computational soft robot reasoning using intrinsic sensors. [18]–[20] introduced novel actuators with embedded sensors for understanding static grasped object shapes and estimating motion profiles. Other works [21]–[23] have combined established sensor and actuator designs with statistical methods for classification of grasps, texture identification, and pose reconstruction. Recent success has been found in the wearables literature [24], [25], with work presenting model-free learning-based methods for human hand reconstruction or grasp classification. For fully soft structures, [26], [27], have demonstrated significant promise for pose reconstruction. The former of these works employs a finite element model to interpolate limb sheath stretches through capacitive strain data in a model-based way; the latter takes a model-free approach to predict soft finger pose using a recurrent neural networks on physical hardware with strain sensing. However, none of these works reason about system dynamics or examine co-design with strain sensor placement.

III. Preliminaries

Given a budget for the maximum number of sensors we wish to place on a robot, τ, we seek to learn both the optimal locations on a soft robot body to place those sensors, along with a model that uses those sensor inputs to solve complex tasks. Such optimal sensor locations depend both on the type of motion the robot will experience as well as which regions of the robot have the highest task relevancy. The combination of complex geometries and dynamics of soft robots make this a highly difficult task, not easily solved by naïve strategies such as simply sensorizing the most dynamic portions of the robot (which may be densely clustered and are possibly irrelevant to the target task). Furthermore, while this paper deals with soft robots in simulation, we wish for our sensor designs to be physically deployable, meaning only the intrinsic strain and strain rate information operating in a relative reference frame (*i.e.*, material space) can be used.

In this paper, we consider dynamic soft robots simulated using the material point method (MPM). MPM approximates continuum robots as collections of particles. For a d-dimensional robot ($d = 2$ or 3), each particle i contributes a small bit of volume to the overall soft robot, and stores relevant information about its dynamic state; these include its position $\mathbf{p}_i \in \mathbb{R}^d$, its velocity $\mathbf{v}_i \in \mathbb{R}^d$, its deformation gradient $\mathbf{F}_i \in \mathbb{R}^{d\times d}$ (a measure of kinematic stretch and shear), and its affine velocity $\mathbf{C}_i \in \mathbb{R}^{d\times d}$ (a measure of kinematic stretch and shear rate). We employ the open source, physically-based ChainQueen simulator [3], whose differentiability can be used to solve motion-planning tasks, useful for generating task-driven trajectory datasets. We emphasize that although we choose MPM as our soft robot simulation method and its particle representation as input to our models, our learning method is not specific to MPM. Our method is amenable to any simulation method which has a node-based representation, including finite element methods. For the curious reader, however, [28] and [29] present in-depth tutorials on MPM.

We make the following assumptions about our strain sensor model: 1) Strain (rate) measurements have perfect accuracy and

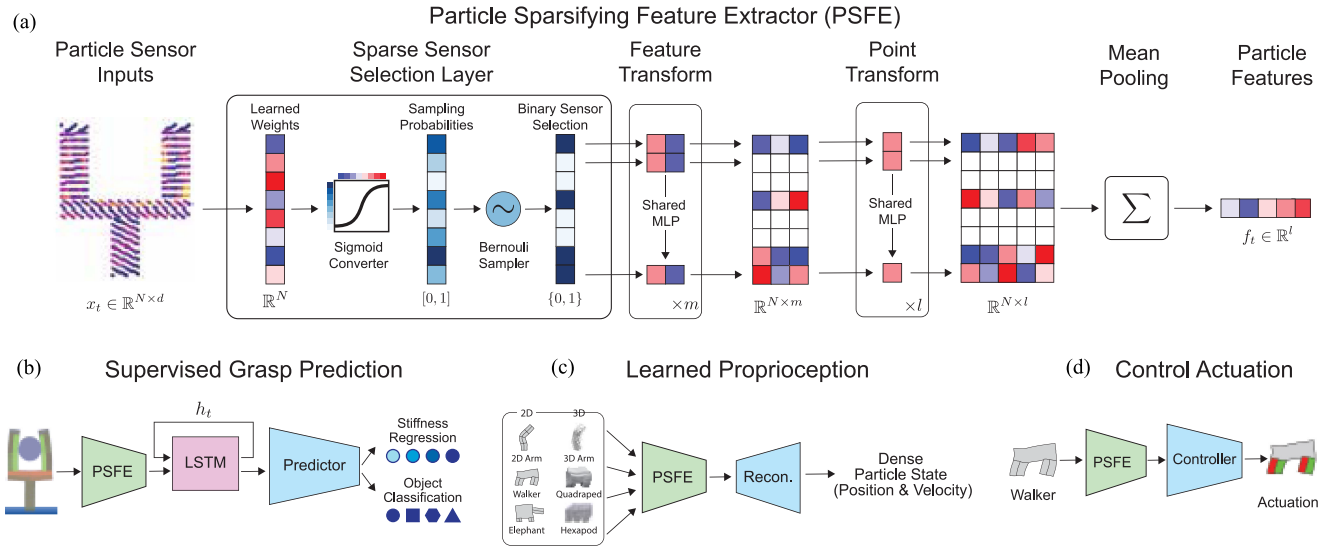

Fig. 1. (a) The foundation of our models is a Particle Sparsifying Feature Extractor (PSFE) which takes as input the full, dense sensory information (left) and extracts a global feature representation (right) from a sparse subset of the inputs. Our model simultaneously learns this representation and sparsification of the input. Since the input is an unordered point cloud, the PSFE also maintains order invariance through shared feature and point transformations as well as global pooling operations. We employ the PSFE on various complex tasks: (b) Supervised regression and classification of object characteristics from grasp data. (c) Learned proprioception by combining PSFE with a variational decoder network. (d) Learned control policies for a soft robot.

no latency. 2) Strain sensors can be placed on each particle and measure the strain (rate) over the volume of that particle. 3) Each sensor provides both strain and strain rate information along each (material space) axis. We relax the first assumption in experiments in Sec. V-D. The Green Strain $\mathbf{S}_i$ of a particle i is defined as: $\mathbf{S}_i = \frac{1}{2}(\mathbf{F}_i^T\mathbf{F}_i - \mathbf{I})$ The Green strain rate $\frac{d\mathbf{S}_i}{dt}$ of a particle i can be efficiently computed analytically as: $\frac{d\mathbf{S}_i}{dt} = \frac{1}{2}\mathbf{F}_i^T(\mathbf{C}_i^T + \mathbf{C}_i)\mathbf{F}_i$. We provide a derivation in Appendix 1. As most real-world strain sensors can only measure stretch, but not shear, we work only with the diagonal entries of $\mathbf{S}$ and $\frac{d\mathbf{S}}{dt}$ in this paper. Most real-world soft robot sensing systems rely purely on stretch sensors (e.g. [27]), which are cheaper and easier to fabricate than shear sensors. Despite this limitation, efficient task learning is still possible, emphasizing that there is a *likely* subspace of robot motions that can be learned from sparse sensory input, a subspace which can be leveraged through data-driven methods like ours.

The remainder of this paper is organized as follows. First, we describe our point-based neural network architecture, which we use for all tasks considered. Next, we describe our sparse neuron model, which populates the input layer of our network and adaptively down-selects salient sensor locations. We then explain how we combine sensor and task learning in a simple learning algorithm. Finally, we demonstrate our co-learning approach on two novel soft robotics problems — dynamic in-hand sensing and dynamic state reconstruction — and show applications to further tasks such as control.

IV. Method

In this section, we formulate a neural architecture suited to sparse sensorization for soft robotics tasks. Our network is modular and takes instrinsic sensor readings of the body as input. A point sparsification and feature extraction (PSFE) network simultaneously learns a representation of these sensor readings along with a set of sparse sensor locations. Our PSFE (Fig. 1 A) is used as the core building block for all demonstrations and applications presented in this paper, including object grasping prediction (B), learned proprioception (C), and control (D).

a) Point sparsification and feature extraction: Core to our algorithm is a neural network (NN) architecture for learning representations of soft robots from their raw, dense, and intrinsic sensor readings. By using a NN architecture, our method can employ drop-out techniques targeted to NN *input* layers as a means of choosing sparse sensor location.

To achieve accurate state representation during simulation, soft robots must be discretized into many individual particles. Candidate sensor locations are co-located with our MPM particles, creating a particle set which can be viewed as a point cloud. Thus, our architecture builds upon prior models for point cloud scene understanding, but makes important adaptations for the physical soft robotics setting.

Our representation builds upon PointNet [30], treating our robot particle set as a set of N order-invariant points in space. This invariance is created using a series of shared feature transformations followed by point transformations into point features. The final representation is computed by pooling the features, again using order-invariant (mean) pooling. PSFE contributes two modifications in order to make the PointNet architecture amenable to soft robotics and sensor placement.

First, the original PointNet architecture reasoned purely on measured (x, y, z) Cartesian coordinates in space as input. Our sensors, however, are only able to measure axial strain and strain rates. While there is nothing preventing using these strain "coordinates" in lieu of Cartesian coordinates, such a representation would fail, due to PointNet's inherent symmetric interpretation

of point distributions. Each input to PointNet is processed as an unordered set, not an ordered list. Thus, a distribution of strain and strain rates would be interpreted the same way regardless of where they occurred on a soft robot; wholly different system dynamics could be interpreted as equivalent states. To avoid this problem, we carry along the local coordinates of the *undeformed* soft robot. Since the undeformed coordinates are the same regardless of robot state, this provides no feature information, but rather disambiguates particles in the point set while also providing spatial coherence which the architecture exploits. The result is that each point is represented by a $3 \times d$ vector: strains, strain rates, and static, undeformed position (material space) coordinates.

Second, the input layer to the PointNet architecture is deterministic. We modify our architecture to allow for stochastic weights in the input layer during training. During each training step, neurons corresponding to input i are sampled on or off according to Bernoulli random variable parameterized by some θ_i. Our training procedure will push θ_i to 0 or 1, thus learning a deterministic model of important inputs, and equivalently, optimal sensor placements.

b) Sparse sensor selection layer: Our representation uses stochastic neurons to sample different sensor candidates for extracting features as part of our Sparse Sensor Selection Layer (SSSL). By combining this representation with a sparsifying loss, we devise a training objective to learn an optimal sensor placement. Unlike classical NN weights, which are typically unbounded real values, our SSSL learns a set of binary $\{0, 1\}$ weights as a sparse mask over our dense input. This is done by first learning a continuous weight vector, transforming to probabilities $\in [0, 1]$, and using these probabilities to Bernoulli sample the binary mask. More concretely, given a set of continuous weights, W_i, we compute the probabilities, θ_b, and sample binary weights, b_i, as follows:

$$\theta_{b_i} = \sigma(W_i) \tag{1}$$

$$b_i = \text{Bernoulli}(\theta_i) = \text{ceil}\,(\theta_i - U) \tag{2}$$

where $\sigma(\cdot)$ is the sigmoid function, and $U \sim \text{Unif}(0, 1)$ is a uniform random number between 0 and 1. While the forward pass through these functions is well-defined, the backward pass is not, since the gradient of $ceil(\cdot)$ is zero for almost all inputs. To enable backpropogation we override the gradient:

$$\frac{\partial b}{\partial \theta_i} = \theta_i - U. \tag{3}$$

This modification, similar to BinaryConnect [31], allows the SSSL to forward propagate continuous, learned weights through a binary mask, transforming dense sensor candidates into sparse sensor locations. Fig. 2 illustrates the transformation of these random variables into discrete sensor locations.

c) Co-learning algorithm: While a stochastic representation of neurons is useful for testing their saliency (evidenced by gradients with respect to $\boldsymbol{\theta}$), this sampling does not sparsify the input layer in its own right. In order to sparsify over the input layer and sensor placement, we introduce a regularizer that can be added to the task loss $L_t((\mathbf{w})$ of any learning problem, where $\mathbf{w}$ are the non-sparsifying weights of the learning model. Given

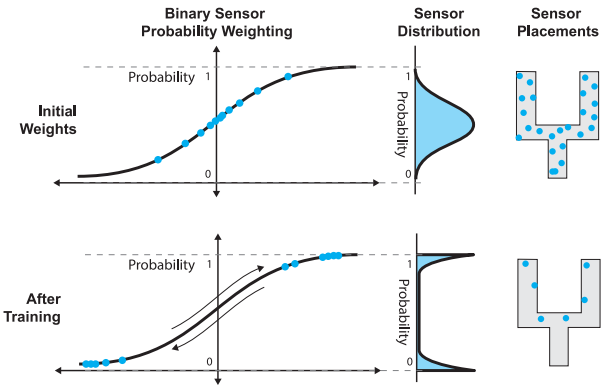

Fig. 2. Learning sparsification. A 1D sigmoid function independently parameterizes the probability of each sensor location with weight W_i. Sampled gradients evolve W_i toward $\pm\infty$, pushing the corresponding θ_i probabilities toward 0 or 1, sparsifying the candidate sensor set.

Algorithm 1: Training Algorithm.

Given: Dataset $\mathcal{I}$, task loss function L_t, learning architecture with N inputs.
Randomly initialize network weights.
while $\sum_i \theta_i \geq \tau$ or $\sum_i \text{round}(\theta_i) > \tau$ **do**
  for Minibatch $I \in \mathcal{I}$ **do**
    $L = L_t + w_s\, L_s$
    $\mathbf{w} \leftarrow \mathbf{w} - \eta \nabla_{\mathbf{w}} L$ (Sec. IV-0b)
$\boldsymbol{\theta} \leftarrow \text{constant}(\text{round}(\boldsymbol{\theta}))$
while Not converged **do**
  for Minibatch $I \in \mathcal{I}$ **do**
    $\mathbf{w} \leftarrow \mathbf{w} - \eta \nabla_{\mathbf{w}} L$ (Sec. IV-0b)

a target sensor budget τ and a max sensor count/input layer size N, we have a sparsifying regularizer $L_s(\boldsymbol{\theta})$ and a total loss $L(\mathbf{w}, \boldsymbol{\theta})$ to be minimized:

$$L_s(\boldsymbol{\theta}) = \left| \frac{\sum_i \theta_i - \tau}{N} \right|, \quad L(\mathbf{w}, \boldsymbol{\theta}) = L_t(\mathbf{w}) + w_s L_s(\boldsymbol{\theta})$$

where w_s is a user-defined weight that trades off sparsification and the target learning task. The absolute value efficiently pushes most of the θ_i values to 0. Despite the sparsifying nature of L_s, it is possible for a few weights to have partial probability at convergence, achieving a "mixed strategy Nash" against a target dataset. To alleviate this concern, once $\sum_i \theta_i < \tau$ and $\sum_i \text{round}(\theta_i) \leq \tau$, we appropriately round all θ_i to 1 or 0, fix them, and continue training (Algorithm 1).

V. Results

We evaluate our method on two supervised learning problems — grasped object classification and proprioceptive state reconstruction. Since previous work [9], [10] is inapplicable to a general task learning regime, we focus on two baselines: human-specified sensor locations and random placement. These demonstrations show the full power of a sensor placement model in our co-learning pipeline as opposed to the fixed-sensor, static representation learning problem.

A. Tactile Sensing

Given a dynamic, soft robot gripper with a fixed, open loop trajectory, we seek an optimal sensor placement and

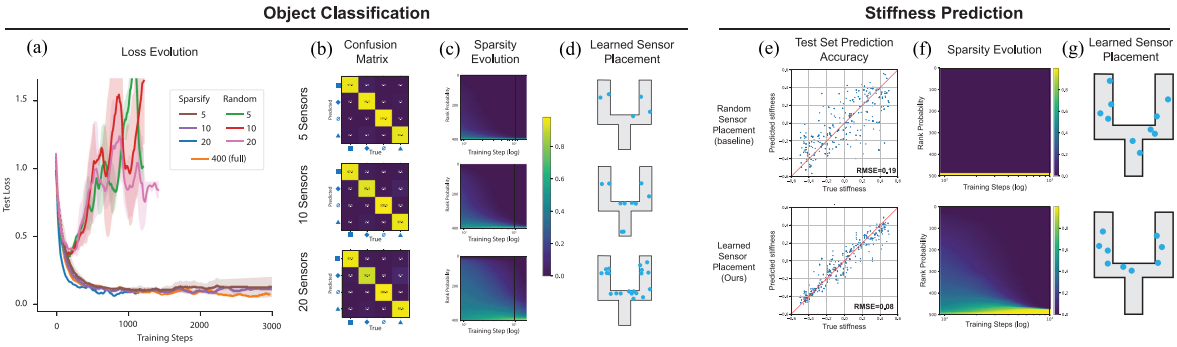

Fig. 3. Results for object classification ((a)–(d)) and stiffness regression ((e)–(g)) tasks. (a) Convergence of the loss for $\tau = 5, 10, 20$ on the sparifying algorithm, with comparison to baselines and full sensorization. (b) Confusion matrices for our co-learning algorithm across different sensor budgets. (c) Evolution of sensor probabilities over training time, validating that each θ_i converges toward 0 or 1, with final sensor placements shown in (d). (e) Prediction accuracy for the stiffness regression task, comparing the baseline (top) to our algorithm (bottom). (f) Evolution of sensor probabilities. (g) Optimized sensors have more task-relevant locations.

classification/regression network that can best reason about a grasped object. In the classification task, the gripper must disambiguate between four shapes (square, diamond, triangle and *no shape*) through manipulation. 120 simulations are performed for each class in total. In the regression task, a ball with unknown Young's modulus is manipulated and its stiffness must be inferred. 1000 simulations are captured in total for the stiffness regression problem with Young's modulus varying from 10^0 to 10^4. Example simulations can be seen in the video.

In all tasks, the location and size of the objects is varied between runs, with noise added to actuators. This makes the learning tasks challenging as the robot must learn to reason about the qualities of the different tactile responses – subtle differences that can be difficult to detect. The network cannot simply memorize motion trajectories, especially given the added noise. The classification task amplifies this difficulty, since contact with the object only happens at a few rather discrete moments, making supervision sparse.

Each sensor on the gripper provides $2 \times d$ inputs at each time step, over a time series of $T = 50$ simulation steps. $\tau \times 4 \times T$ time series inputs are captured from the gripper and provided to the network.

To solve these tactile sensing tasks, we augment our PSFE with a recurrent LSTM module to process the entire trajectory of data. At every timestep, features are extracted using our PSFE, fed into our LSTM with 50 recurrent units, and finally passed through a single fully connected layer which predicts the output. For the shape classification task, the network was to predict a categorical distribution, $f(\mathbf{x}; \mathbf{w})$ over K possible objects with softmax activation, and was trained using a cross entropy loss function. For the stiffness regression task, the final layer was to output a real scalar, and was trained using a mean squared error loss from the ground truth stiffness. These losses can be expressed as,

$$\underbrace{L_t = -\sum_{k=1}^{K} \mathbf{y}_k \log(f(\mathbf{x}; \mathbf{w})_k)}_{\text{(shape classification)}} \qquad \underbrace{L_t = ||\mathbf{y} - f(\mathbf{x}; \mathbf{w})||_2}_{\text{(stiffness regression)}}$$

Results for both tasks can be found in Fig. 3. We used Adam as our optimizer, set the sparsity regularizer to 0.01, the learning rate to 5×10^{-4}, and the minibatch size to 8. We ran our pipeline for $\tau = 5, 10, 20$. We compare to a baseline where τ sensors are assigned randomly *a priori* and fixed. Our co-learning method vastly outperforms the baseline in generalization on test sets. While the baseline overfits quickly due to poor sensor placement, our algorithm places sensors in regions most relevant to the tactile sensing task. Specifically, it places sensors at common gripper-object interaction points, such as the interior gripper base and fingertips.

B. Proprioception

Given strain and strain rate inputs from a collection of sensors (a $2 \times d \times \tau$ input space), the goal is to reconstruct the entire position map (deformation) and velocity map (deformation rate) of the robot (a $2 \times d \times N$ prediction space). Here, n is a set number of particles whose pose we wish to predict. This is difficult for two reasons. First, $\tau \ll N$; a low input dimension must predict a much larger output dimension. Second, four of our six robots are floating base robots; neither rigid degrees of freedom nor ground contact information are provided, making state estimation challenging.

We employ a similar architecture as used in Sec. V-A. The PSFE outputs an 8 dimensional latent space. Then, a fully connected MLP is appended of hidden layers (64–128) with ReLU activations everywhere but the final layer translates the latent space to pose predictions. While the bottlenecking latent space may seem a handicap, we still achieve excellent pose prediction, since soft robots have natural subspaces that are captured by neural architectures [5]. Further, a compact latent space unlocks applications such as motion subspace visualization and control, which we explore later.

Our loss function was chosen to be the L_2 difference of the reconstruction of the normalized positions and velocities of each particle compared to ground truth. We also add a small variational autoencoder loss [32], [33] to ensure a descriptive latent space. Formally, our task loss $L_t(\mathbf{w})$ is:

$$L_t = \sum_{(\mathbf{x},\mathbf{y}) \in \Upsilon} \|f(\mathbf{x}; \mathbf{w}) - \mathbf{y}\|_2 + w_r L_r(\mathbf{x}, \mathbf{y}; \mathbf{w}) \qquad (4)$$

TABLE I

AVERAGE MINIMUM RECONSTRUCTION ERRORS (LOWER IS BETTER) OF OUR SIX ROBOTS FOR THREE SENSOR COMBINATIONS, FOR OUR ADAPTIVE METHOD, FOR THE RANDOM SENSOR PLACEMENT BASELINE, AND WHERE APPLICABLE, FOR THE HUMAN BASELINE. FOR OUR ADAPTIVE METHOD, MINIMA WERE TAKEN ONLY ONCE THE TRAINING REACHED THE CLAMPED, FINE-TUNING PHASE. OUR METHOD ALWAYS OUTPERFORMS THE RANDOM AND HUMAN BASELINES AND TYPICALLY BEATS THEM BY A SIGNIFICANT MARGIN; BOLDED RESULTS INDICATE WHERE OUR ALGORITHM OUTPERFORMS THE RANDOM BASELINE BY AT LEAST A STANDARD DEVIATION ON EITHER SIDE. AN ESTIMATED "BEST" RECONSTRUCTION ERROR FOR OUR ARCHITECTURE IS PRESENTED IN THE FAR RIGHT COLUMNS, COMPUTED BY TRAINING WITH A SENSOR ON EACH TARGET PARTICLE

Minimum Reconstruction Error By Task (Mean/Std.)								
Robot + Sensor Count	Test Adapt.	Test Rand.	Train Adapt.	Train Rand.	Test Human	Train Human	Full Sensor Test	Full Sensor Train
2D Arm 4 Sensors	**9.29/0.403**	12.0/2.16	**4.09/0.114**	5.44/0.510	-	-	5.27/0.058	3.25/0.027
2D Arm 5 Sensors	**8.93/0.821**	12.2/2.22	**3.93/0.228**	5.38/0.569	14.245538/1.91	6.88/0.71	"	"
2D Arm 6 Sensors	**7.89/0.43**	10.1/1.34	**3.73/0.092**	4.66/0.307	-	-	"	"
2D Elephant 4 Sensors	9.363/0.333	10.0/0.687	**6.95/0.330**	8.03/0.524	-	-	6.29/1.69	5.60/0.14
2D Elephant 5 Sensors	**8.56/0.516**	9.89/0.460	**6.58/0.252**	7.83/0.353	10.03/1.00	7.69/0.48	"	"
2D Elephant 6 Sensors	**8.21/0.280**	8.86/0.420	**6.32/0.150**	7.10/0.247	-	-	"	"
2D Biped 4 Sensors	11.97/0.7106	12.16/0.5069	8.827/0.1919	9.319/0.2472	-	-	8.81/0.162	7.57/0.173
2D Biped 5 Sensors	11.90/0.5839	12.02/0.4421	8.724/0.2418	9.264/0.2446	12.60/0.86	9.46/0.44	"	"
2D Biped 6 Sensors	11.84/0.5308	11.95/0.5392	8.692/0.2264	9.222/0.2792	-	-	"	"
3D Hexapod 4 Sensors	**6.04/0.146**	6.68/0.263	**4.80/0.153**	5.03/0.045	-	-	5.53/0.110	4.71/0.113
3D Hexapod 5 Sensors	**6.01/0.142**	6.67/0.187	**4.81/0.131**	5.06/0.063	-	-	"	"
3D Hexapod 6 Sensors	**6.01/0.134**	6.58/0.229	4.843/0.132	5.019/0.085	-	-	"	"
3D Quadruped 4 Sensors	**6.60/0.260**	7.08/0.321	4.38/0.070	4.56/0.125	-	-	5.63/0.097	4.36/0.096
3D Quadruped 5 Sensors	6.45/0.294	6.85/0.345	4.34/0.087	4.51/0.113	-	-	"	"
3D Quadruped 6 Sensors	6.36/0.322	6.70/0.373	4.28/0.123	4.48/0.115	-	-	"	"
3D Arm 4 Sensors	**3.19/0.118**	3.95/0.433	**3.14/0.114**	3.86/0.390	-	-	2.65/0.078	2.60/0.076
3D Arm 5 Sensors	**3.17/0.103**	3.88/0.437	**3.11/0.101**	3.78/0.400	-	-	"	"
3D Arm 6 Sensors	**3.14/0.099**	3.79/0.400	**3.08/0.096**	3.70/0.367	-	-	"	"

where Υ is the dataset of strain (rate) $(\mathbf{x})$ and position/velocity $(\mathbf{y})$ pairs, $f(\cdot;\mathbf{w})$ is our network which predicts robot pose for a given input, and L_r is the VAE regularizer loss (see, e.g., [34]). $w_r = 10^{-4}$ is a regularization weight. Adam was used as the optimizer with a learning rate 10^{-3}. For each problem, 400 randomly downsampled particles were simultaneously used as reconstruction targets and candidate sensor locations. Other hyperparameters are in Appendix 2.

Our dataset was generated by capturing frames from trajectories generated during robot motion optimization. We used the control optimization from [3], which performs gradient descent on model-based simulation gradients. By capturing a variety of motions from different parts of the optimization, a wide variety of motion frame data was captured. Six pneumatically-powered robots were explored based on models in prior related work: a 2D Biped, a (fixed-base) 2D Arm, a 2D Elephant, a 3D Arm, 3D Quadruped, and a 3D Hexapod. Examples of trajectories used for data collection, along with sample reconstructions, can be found in the video. Each of the learning experiments was repeated five times for $\tau = 4, 5, 6$, and compared to a baseline of random sensor locations. Results can be seen in Table I; convergence plots can be found in Appendix 3.

Our adaptive co-learning of sensors and task vastly outperforms baselines for all tasks, often with strong statistical significance. We performed a paired t-test for the aggregate results of all robot morphologies; strong statistical significance was found for all morphologies (Table III) except for the 2D Biped, which is the easiest example. For the Arm examples, our algorithm achieves a 30% reduction in error compared to the baseline; the 2D Elephant and 3D Hexapod generally perform 10% better in the adaptive case. Reasonably, our algorithm delivers the largest improvements on robots with highly dynamic deformations, as these are more difficult problems where sensor placement matters most. Our algorithm outperforms baselines on dynamically consistent robots such as the Biped and Quadruped as well, but with smaller gains.

TABLE II

PERFORMANCE SCORES (LOWER IS BETTER) FOR MANIPULATION TASKS ON BOTH CLASSIFICATION (OBJECT DETECTION) AND REGRESSION (MATERIAL STIFFNESS) WITH VARYING NUMBER OF SENSORS. OUR METHOD SIGNIFICANTLY OUTPERFORMS BOTH RANDOM AS WELL AS HUMAN LABELLING BASELINES

	Classification			**Regression**		
	5	10	Full	5	10	Full
Random	0.19/0.03	0.18/0.03	0.003/0.005	0.18/0.07	0.18/0.04	0.01/0.0
Human	0.21/0.10	0.06/0.06	-	0.13/0.01	0.12/0.02	-
Sparse	**0.009/0.01**	**0.01/0.005**	-	**0.06/0.02**	**0.08/0.03**	-

TABLE III

P-VALUES FROM A PAIRED T-TEST COMPARING OUR ADAPTIVE ALGORITHM TO THE RANDOM BASELINE APPROACH

P-Values For Proprioceptive Tasks					
2D Arm	2D Elephant	2D Biped	3D Hexapod	3D Quadruped	3D Arm
5.4×10^{-4}	2.6×10^{-3}	0.58	2.5×10^{-3}	0.011	1.64×10^{-5}

Fig. 4 presents sample reconstructions of the elephant, along with the contributions of individual sensors. Here, three of the five sensors are placed on or near the trunk, which experiences the most extreme deformation (rates) and thus requires the most information to reconstruct accurately.

C. Comparison to Human Intuition

The main goal in this paper is to develop an *automated* means of co-learning soft robotic tasks and optimizing sensor placement. In order to further emphasize the value of such an automated sensor selection, we compare it to manual human selection. Similar to the random baselines, for these experiments, sensor locations were chosen *a priori* to training.

In order to gather data for these experiments, ten participants (all engineers with knowledge and experience in continuum mechanics) were provided renderings of all of the 2D robot morphologies, and provided videos from the captured motion dataset as well as a description of the target task. The participants were then asked to select particle locations, choosing where they

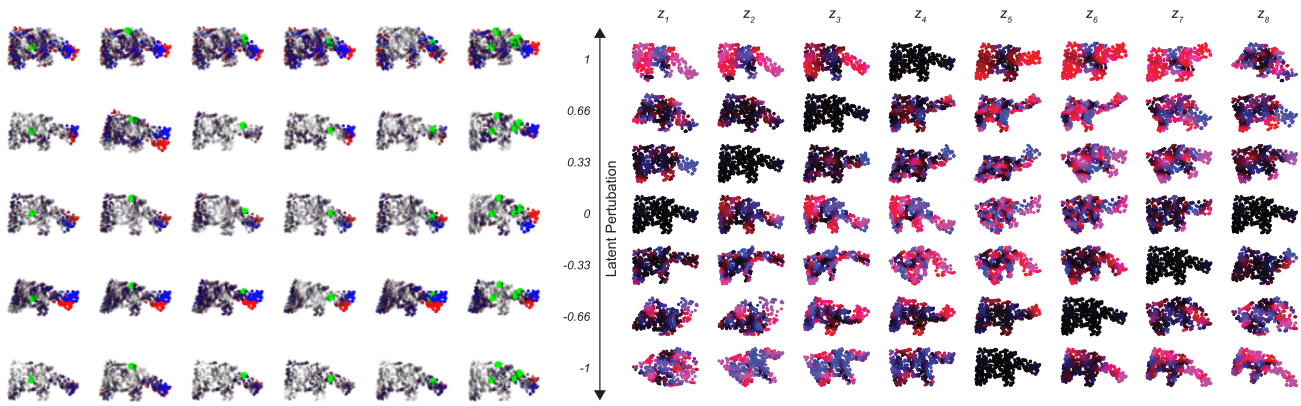

Fig. 4. *Left:* Visualization of the effects of sensor locations (large green circles) on the 2D elephant with different sensor placements (columns) over five exemplar poses (rows). Frames show the reconstruction error with all but the shown sensor turned off. Brighter colors indicate larger deviation of the reconstructed robot (blue) from ground-truth (red). Sensors lower reconstruction errors in their immediate neighborhoods the most. The rightmost column shows all five sensors turned on, yielding the best reconstructions. *Right:* Latent space of the 2D elephant. Columns represent different latent dimensions of the elephant, generated by one-hot latent vector activations. Each column ranges latent activations from -1.0 to 1.0. Redder particles indicate higher speeds in x; bluer particles indicate higher speeds in y.

thought would be the best sensor locations to learn the task. 2D morphologies were used in these experiments in order to facilitate the user data collection process. On the manipulation task of both object classification and material regression, humans were evaluated on their ability to select sensor placements for 5, and 10 sensors, aligning with Fig. 3.

These experiments provide striking results, which can be found in Table I (reconstructions) and Table II (manipulation). Notably, it appears that humans are actually antagonistically bad at choosing sensor locations for some tasks. Not only does our method dominate human baselines, but even the random baseline is competitive with the human baseline, outperforming it in the case of reconstruction and within statistical bounds in the case of manipulation. For example, in the case of the 2D Elephant, many labelers chose to put four sensors on the legs, and only one on the trunk — even though the legs experience relatively minor deformations while the trunk experiences rapid, large deformations. In other cases, the sensors were more equitably spaced in regions of high deformation, but human biases for geometrically salient locations, rather than locations relevant to the experienced strains and task, were noticeable. This insight emphasizes the importance of computational aid to compensate for flawed human intuition for sensor coverage.

D. Resilience to Sensor Noise

To further demonstrate our method's applicability to real-world systems, we compared our adaptive method to the random baseline for both the 2D Arm and the 3D Hexapod, in the presence of sensor noise. These morphologies were chosen because they experience highly dynamic motion where noise would have the most impact on results. For five runs, random Gaussian noise was added to the strain and strain rates, centered at the ground truth, with 1% standard deviation (roughly matching the accuracy of physical strain gauges). While reconstruction was predictably worse than in the no-noise case, our adaptive method still vastly outperformed the random baseline (Table IV), with lower variance.

TABLE IV
AVERAGE MINIMUM RECONSTRUCTION ERRORS (LOWER IS BETTER) OF OUR TWO MOST DYNAMIC ROBOTS FOR THREE SENSOR COMBINATIONS WITH 1% GAUSSIAN NOISE ADDED, FOR OUR ADAPTIVE METHOD, FOR THE RANDOM SENSOR PLACEMENT BASELINE. ANALYSIS METHOD IS THE SAME AS IN THE NO-NOISE CASE. OUR METHOD ALWAYS OUTPERFORMS THE RANDOM AND HUMAN BASELINES; BOLDED RESULTS INDICATE WHERE OUR ALGORITHM OUTPERFORMS THE RANDOM BASELINE BY AT LEAST A STANDARD DEVIATION ON EITHER SIDE

Minimum Reconstruction Error By Task With 1% Noise (Mean/Std.)				
Robot (Sensor #)	Test Adapt.	Test Rand.	Train Adapt.	Train Rand.
2D Arm (4)	**9.42/0.638**	11.24/1.17	4.24/0.071	5.04/0.381
2D Arm (5)	9.07/0.980	11.36/2.05	**4.16/0.222**	5.03/0.572
2D Arm (6)	8.70/0.969	11.09/1.74	**4.06/0.245**	4.91/0.502
3D Hexapod (4)	**6.09/0.03**	6.82/0.325	**4.85/0.091**	5.06/0.100
3D Hexapod (5)	6.13/0.104	6.56/0.365	4.89/0.109	5.01/0.096
3D Hexapod (6)	6.08/0.142	6.53/0.348	4.87/0.125	4.98/0.094

VI. APPLICATIONS

We have judiciously trained our translational proprioception network through a small latent space. This affords two advantages: interpretable latent motion subspaces, and coordinates for robot control.

Motion Subspace Visualization: Our latent space provides natural, low-dimensional coordinates through which to represent the soft robot. Fig. 4 visualizes our eight latent variables, demonstrating interpretable coordinates representing different aspects of the robot's motion.

Control: We demonstrate how the latent feature space learned during propioception training can be used as an observer model for control of the 2D Biped. To our knowledge, our control demonstration is the first example of closed-loop computational control of terrestrial soft robots from intrinsic sensors. The Biped was chosen, since its cyclic motion means a stable gait can be learned from limited training data. We save the learned encoder network during Biped proprioception training and use it as the

observation model during robot control tasks. We optimize a neural network MLP controller with the fixed PSFE observation model. *Only* intrinstic strain and strain information is provided as input to the observer network, meaning no translation, rotation, or contact information is provided in any way to the controller, making this task extra difficult. 20 sensors were required for a robust observation model. As the robot optimizes, it finds a cyclic gait. As one would expect, the latent space observed during this motion is cyclic as well. Please see the video for the optimized robot and a visual analysis of the latent space during simulation.

VII. Conclusion and Future Work

We have demonstrated our task and sensor co-learning algorithm on a wide array of learning tasks, including tactile sensing and dynamic state reconstruction. Our adaptive algorithm vastly outperforms baselines and our representation has useful applications to state visualization and control.

Our method reveals several directions for future exploration. First, it would be interesting to fabricate our simulated robots and demonstrate trained sensing capabilities in the wild. Developing a general pipeline for translating learned models to physical hardware would prove physical practicality of our method and address the difficult "sim-to-real" research question. Several considerations would have to be made to accommodate the physical world, including more accurate sensor noise modeling and accounting for sensor bulk, sensor latency, and potentially physical fabrication constraints. Second, many strain sensors only measure single-directional strain; thus, it would be useful to have a method for choosing sensor orientation during learning. Finally, while we focused on soft robots in this manuscript, we believe our method is general enough to be applied to high-dimensional sensor co-design problems in other fields, and hope it inspires such research.

References

[1] N. Cheney, R. MacCurdy, J. Clune, and H. Lipson, "Unshackling evolution: Evolving soft robots with multiple materials and a powerful generative encoding," *ACM SIGEVOlution*, vol. 7, no. 1, pp. 11–23, 2014.

[2] C. Duriez and T. Bieze, "Soft robot modeling, simulation and control in real-time," *Soft Robotics: Trends, Applications and Challenges*, Springer, 2017, pp. 103–109.

[3] Y. Hu *et al.*, "Chainqueen: A real-time differentiable physical simulator for soft robotics," in *Proc. Int. Conf. Robot. Automat.*, 2019, pp. 6265–6271.

[4] O. Goury and C. Duriez, "Fast, generic, and reliable control and simulation of soft robots using model order reduction," *IEEE Trans. Robot.*, vol. 34, no. 6, pp. 1565–1576, Dec. 2018.

[5] A. Spielberg, A. Zhao, T. Du, Y. Hu, D. Rus, and W. Matusik, "Learning-in-the-loop optimization: End-to-end control and co-design of soft robots through learned deep latent representations," in *Adv. Neural Inf. Process. Syst.*, 2019, pp. 8284–8294.

[6] R. J. Spletzer and C. J. Taylor, "Dynamic sensor planning and control for optimally tracking targets," *Int. J. Robot. Res.*, vol. 22, no. 1, pp. 7–20, 2003.

[7] C. Giraud and B. Jouvencel., "Sensor selection: A geometrical approach," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. Human. Robot Interact. Cooperat. Robots*, vol. 2, Aug. 1995, pp. 555–560.

[8] C. Schroeter, M. Hoechemer, S. Mueller, and H. Gross, "Autonomous robot cameraman - Observation pose optimization for a mobile service robot in indoor living space," in *Proc. IEEE Int. Conf. Robot. Automat.*, May 2009, pp. 424–429.

[9] G. Olague and R. Mohr, "Optimal camera placement for accurate reconstruction," *Pattern Recognit.*, vol. 35, no. 4, pp. 927–944, 2002.

[10] A. Krause, J. Leskovec, C. Guestrin, J. VanBriesen, and C. Faloutsos, "Efficient sensor placement optimization for securing large water distribution networks," *J. Water Resour. Plan. Manage.*, vol. 134, no. 6, pp. 516–526, 2008.

[11] D. Cliff, P. Husbands, and I. Harvey, "Explorations in evolutionary robotics," *Adaptive Behav.*, vol. 2, no. 1, pp. 73–110, 1993.

[12] B. Kim, J. Ha, F. C. Park, and P. E. Dupont, "Optimizing curvature sensor placement for fast, accurate shape sensing of continuum robots," in *Proc. IEEE Int. Conf. Robot. Automat.*, May 2014, pp. 5374–5379.

[13] M. Bächer *et al.*, "Defsense: Computational design of customized deformable input devices," in *Proc. CHI Conf. Human. Fact. Comput. Syst.*, 2016, pp. 3806–3816.

[14] J. Tapia, E. Knoop, M. Mutnỳ, M. A. Otaduy, and M. Bächer, "Makesense: Automated sensor design for proprioceptive soft robots, *Soft Robot.*, vol. 7, no. 3, pp. 332–345, 2019.

[15] J. N. P. Martel, L. Mueller, S. J. Carey, P. Dudek, and G. Wetzstein, "Neural sensors: Learning pixel exposures for HDR imaging and video compressive sensing with programmable sensors," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 42, no. 7, pp. 1642–1653, Jul. 2020.

[16] A. Chakrabarti, "Learning sensor multiplexing design through back-propagation," in *Proc. 30th Int. Conf. Neural Inf. Process. Syst.*, pp. 3089–3097, 2016.

[17] C. Schaff, D. Yunis, A. Chakrabarti, and M. R. Walter, "Jointly optimizing placement and inference for beacon-based localization," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2017, pp. 6609–6616.

[18] Yiğit Mengüç *et al.*, "Wearable soft sensing suit for human gait measurement," *Int. J. Robot. Res.*, vol. 33, no. 14, pp. 1748–1764, 2014.

[19] N. Farrow and N. Correll, "A soft pneumatic actuator that can sense grasp and touch," in *Proc. IEEE/RSJ Int. Conf. Int. Robots Syst.*, Sep. 2015, pp. 2317–2323.

[20] B. S. Homberg, R. K. Katzschmann, M. R. Dogar, and D. Rus, "Haptic identification of objects using a modular soft robotic gripper," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Sep. 2015, pp. 1698–1705.

[21] T. Inoue and S. Hirai, "Modeling of soft fingertip for object manipulation using tactile sensing," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Oct. 2003, vol. 3, pp. 2654–2659.

[22] Y. Tada, K. Hosoda, Y. Yamasaki, and M. Asada, "Sensing the texture of surfaces by anthropomorphic soft fingertips with multi-modal sensors," *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, vol. 1, Oct. 2003, pp. 31–35.

[23] B. Shih *et al.*, "Custom soft robotic gripper sensor skins for haptic object visualization," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Sep. 2017, pp. 494–501.

[24] O. Glauser and D. Panozzo, "Otmar hilliges, and olga sorkine-hornung. deformation capture via soft and stretchable sensor arrays," *ACM Trans. Graph.*, vol. 38, no. 2, pp. 16, 2019.

[25] S. Sundaram, P. Kellnhofer, Y. Li, J.-Y. Zhu, A. Torralba, and W. Matusik, "Learning the signatures of the human grasp using a scalable tactile glove," *Nature*, vol. 569, no. 7758, pp. 698–702, 2019.

[26] O. Glauser, S. Wu, D. Panozzo, O. Hilliges, and O. Sorkine-Hornung, "A stretch-sensing soft glove for interactive hand pose estimation," in *ACM SIGGRAPH Emerg. Technol.*, 2019, pp. 1–2.

[27] T. G. Thuruthel, B. C. Shih Laschi, and M. T. Tolley, "Soft robot perception using embedded soft sensors and recurrent neural networks," *Sci. Robot.*, vol. 4, no. 26, 2019.

[28] C. Jiang, C. Schroeder, J. Teran, A. Stomakhin, and A. Selle, "The material point method for simulating continuum materials," in *ACM SIGGRAPH 2016 Courses*, 2016, pp. 1–52.

[29] Y. Hu, X. Zhang, M. Gao, and C. Jiang, "On hybrid lagrangian-eulerian simulation methods: Practical notes and high-performance aspects," in *ACM SIGGRAPH 2019 Courses*, 2019, pp. 1–246.

[30] C. R. Qi, H. Su, K. Mo, and L. J. Guibas, "PointNet: Deep learning on point sets for 3D classification and segmentation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 652–660.

[31] M. Courbariaux, Y. Bengio, and J.-P. David, "Binaryconnect: Training deep neural networks with binary weights during propagations," in *Adv. Neural Inf. Process. Syst.*, vol. 28, 2015, pp. 3123–3131.

[32] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," in *Proc. 2nd Int. Conf. Learn. Representations*, Banff, AB, Canada, 2014.

[33] D. J. Rezende, S. Mohamed, and D. Wierstra, "Stochastic backpropagation and approximate inference in deep generative models," in *Proc. Int. Conf. Mach. Learn.*, 2014, pp. 1278–1286.

[34] C. Doersch, "Tutorial on variational autoencoders," *CoRR*, vol. abs/1606.05908, 2016, *arXiv:1606.05908*, [Online]. Available: http://arxiv.org/abs/1606.05908.