SENSORIMOTOR GRAPH: Action-Conditioned Graph Neural Network for Learning Robotic Soft Hand Dynamics

João Damião Almeida∗ 1, Paul Schydlo∗ 1,2, Atabak Dehban1 and José Santos-Victor1

Abstract—Soft robotics is a thriving branch of robotics which takes inspiration from nature and uses affordable flexible materials to design adaptable non-rigid robots. However, their flexible behavior makes these robots hard to model, which is essential for a precise actuation and for optimal control. For system modelling, learning-based approaches have demonstrated good results, yet they fail to consider the physical structure underlying the system as an inductive prior. In this work, we take inspiration from sensorimotor learning, and apply a Graph Neural Network to the problem of modelling a non-rigid kinematic chain (i.e. a robotic soft hand) taking advantage of two key properties: 1) the system is compositional, that is, it is composed of simple interacting parts connected by edges, 2) it is order invariant, i.e. only the structure of the system is relevant for predicting future trajectories. We denote our model as the “Sensorimotor Graph” since it learns the system connectivity from observation and uses it for dynamics prediction. We validate our model in different scenarios and show that it outperforms the non-structured baselines in dynamics prediction while being more robust to configurational variations, tracking errors or node failures.

I. INTRODUCTION

A. Motivation

Soft robotics rose as a branch of robotics that uses soft materials, drawing inspiration from organic systems, to generate agents with a more flexible and adaptive behavior. These robots however are particularly challenging to model since: 1) they do not have a rigid structure, instead they obey to a continuous topology with infinite degrees of freedom, 2) the elastic contact of soft gippers with the manipulated objects alters their shape and kinematics over time, and 3) imperfections that are inherent to some manufacturing techniques together with the time-varying parameters (e.g. stiffness, friction, etc.) caused by the mechanical and abrasive wear of the rubbery materials [1] call for self-calibration.

The need for self-calibration is not unique to soft robots. In fact, during the first few months of life, newborns go through a phase of intensive sensorimotor learning, performing endless “experiments” with their own bodies, using motor babbling and observing the motor outcomes, to learn how to control their arms and hands [2]. This learning process to understand the structure and behaviour of our own body, called sensorimotor learning, is also observed when, for example, an amputee has to learn how to adapt to a prosthesis after a replacement surgery.

Consider now an analogous situation for a robotic manipulator, Fig. 1a), where a robot needs to understand how to use its end-effector. To effectively control this system we would first need to write down the equations which govern its motion. This approach is useful when the relations between the different parts are simple, but it becomes unfeasible for systems with complex non-trivial dynamics such as soft robotics and rubbery materials commonly used in prosthetic limbs. In this work we explore a self-calibration strategy inspired by infant sensorimotor learning through motor babbling: by observing the effects of actuation signals on a robotic soft-hand we infer the underlying structure in the form of a “Sensorimotor Graph” and predict future states conditioned on this graph.

B. Related Work

1) Forward Models: In classical approaches, the system model and parameters are assumed to be known in advance, and the controller design makes use of this information [6]. However, when the complexity of the system increases,
considering the kinematics, under-actuation, and dynamics of dexterous hands, it becomes very difficult to explicitly write precise and detailed model equations, which hampers the model-based control system design. Adaptive control emerged as a partial solution to this problem [7]. By allowing parameters of the model to be estimated, it adapts to some levels of uncertainty regarding model parameters, such as e.g.: weight of objects. However, this approach is limited to a fixed structure of the model and is unsuitable for complex systems.

This has lead the community to consider learning based approaches to tackle the effects of under-modeling [8], [9]. In the control domain these methods fall into two broad classes: model free and model based control [10]. Model free variants of Reinforcement learning [11] can model and control a system by learning to map a system state to a control input, trained by optimizing a reward signal. Notwithstanding, reinforcement learning assumes the availability of interaction samples and a reward signal, that can be costly to collect.

Model-based methods, on the other hand, decouples learning of the system dynamics and control of the system, reducing the need for costly interaction samples. Optimal control formulations based on differential models [5], [12] optimize a cost function conditioned on the system states, reference states and actuation conditions.

Learning based approaches are generally based on two kinds of representations; a user-defined space such as 2D/3D position, or directly on the unprocessed sensory inputs, such as the image space. Image space representations take as input a sequence of frames and predict future frames, modelling the dynamics of the environment in pixel space by compressing the information into a latent space [9]. This representation is not invariant to distribution shifts such as environmental, lighting or camera pose changes.

Methods based on explicit 3D representations on the other hand decouple the challenges related to lighting and camera poses. These approaches are based on either manually annotating the data, adding additional instrumentation in the form of markers, or learning to extract and identify key-point representations in an unsupervised manner [13], [14], [12]. More structured representations have gained attention with the recent growing adoption of graph neural networks, which enable learning-based methods to exploit the structure inherent in the problem domain [12], [15], as we will see in the next section.

We focus on the question of how to learn to model a non-rigid kinematic chain, that is, a dynamical system of inter-connected parts, learning a good model is key for model-based control applications. We will assume explicit 3D point position representations, which allow us to limit the scope of this work to the problem of modelling the non-rigid kinematic chain of a robotic soft hand. Additionally, this representation is invariant to lighting and camera pose changes. In a real system the points can be obtained by either placing markers on the different fingers or by learning to segment a RGB-D camera stream into different finger segments.

2) Graph Neural Networks: Neural networks have enabled great progress in different areas of robotics and computer vision, yet they are very sensitive to input distribution shifts such as changes to order of the input features [16]. This is particularly relevant in situations where we expect tracking or sensing failures. Recently, graph neural network models attempt to bridge the divide between classical structured and more recent deep learning data-driven approaches [14]. The combination mitigates shortcomings by reasoning directly over explicit entities and their relations, the so called relational inductive bias. Imposing architectural constraints, this class of methods captures the inherent compositionality of the system, reasoning about the environment in terms of entities and relations. This is postulated to be similar to how humans reason about their environment [14].

Graph neural networks are structured around a set of entities or nodes/vertices, \( v_i \), and relations or edges between them, \( e_{ik} \). Together the nodes and edges form a graph that defines the dependencies among variables.

Information is propagated through a message passing mechanism in the form of successive edge and node update [17]. Each edge is updated with the function \( \phi^e \) accumulating and conditioning on the neighbouring nodes. Each node is updated with a function \( \phi^v \) accumulating and conditioning on the surrounding edges. Successive iterations of these two functions converge towards the final node values. This mechanism is similar to the way information is collected in a convolutional kernel, where a pixel value is updated as a function of the information of the neighbouring pixels. Following the notation in [14] this can be written as:

\[
\begin{align*}
\mathbf{e}'_k &= \phi^e (\mathbf{e}_k, \mathbf{v}_r, \mathbf{v}_s) \\
\mathbf{v}'_i &= \phi^v (\mathbf{v}_i)
\end{align*}
\]

Where \( r_k \) is the receiving node and \( s_k \) the sending node. \( \mathbf{e}'_k, \mathbf{v}'_i \) and \( N \) denote the new edge values, accumulated messages on the edge, new node values and number of edges respectively. The messages are grouped in the following manner.

\[
E'_i = \{(\mathbf{e}'_k, r_k, s_k)\}_{r_k=i,k=1:N^e}
\]

Since the accumulation function, \( \rho \), is order invariant, this kind of network is especially robust to changes in the order of the points, as long as the topology remains invariant. We take advantage of this property by exploiting the order invariance to make the model robust to possible re-identification or tracking issues. In other words this implies the network has no information about the node order, only the relation between them.

This structure is specially interesting for physical systems with structural constraints and has been applied to the problem domains of non-rigid or articulated systems [15], [18], [12]. Here we continue this line of work and extend it to the domain of soft materials in robotic soft hands.

GNN’s have fueled a growing interest in learning models over graph structured data [19], these methods generally send messages over a fully connected graph, not explicitly
capturing the connectivity in the underlying system. In this work we follow the model proposed in [13] which considers explicit system graph structure, that is, the connectivity of the graph mirrors the system connectivity.

C. Our Approach

We propose to learn a structured differentiable action-conditioned dynamics model based on an explicit joint connectivity graph, the “Sensorimotor Graph”, of a robotic soft hand. By observing node position sequences (trajectories) and actuation signals (finger contractions), we infer the underlying system connectivity and then, to model the system dynamics, we condition our prediction model on the explicit connectivity graph structure. Imposing an explicit relational inductive bias on our system.

Model predictive control formulations can then take the learnt differentiable system dynamics model as constraints and optimize for an actuation signal that minimizes a control objective [5].

Learning dynamics in this manner enables us to decouple control and model learning. This implies we can learn the system in a self-supervised setting, by "babbling" motor actuations, that is, using a wide range of different control signals on the cable-driven actuator and observing the outcomes, avoiding costly reward samples. While we apply the methodology to the concrete example of a robotic soft hand, the method is general to any actuable system of interacting parts.

D. Contributions

We formulate the problem of learning a sensorimotor model for a robotic soft hand’s non-rigid kinematic chain as learning an explicit connectivity graph which we designate Sensorimotor Graph. Additionally, we handled variable number of input nodes and propose the inclusion of explicit actuation signals to the graph forward model proposed in [13]. This enables the down-stream application of the learnt model as a differentiable system dynamics model in an optimal control application.

Furthermore, we validate the proposed action-conditioned graph forward model by simulating robotic soft gripper data with varying parameters and configurations, showing that it can capture the finger’s coupled dynamics, and comparing it to different non-structured baseline models. Lastly we benchmark the model to understand robustness to conditions with increasing adversity.

II. PROBLEM STATEMENT

In this work, we look at modelling a robotic soft hand to obtain a differential dynamics model. We assume a hand composed of smaller parts (e.g. markers placed on multiple rubbery fingers) forming a non-rigid kinematic chain. In a formal way, given \( N_c \) nodes and their observations over time, \( v_k^{1:t} \), \( k \in [1, N_c] \), and actuation signals \( u^{1:t+T} \), \( u^t \in \mathbb{R}^{N_a} \), where \( N_a \) is the number of actuation signals (e.g. one per finger), we wish to predict a sequence \( v_k^{t+1:t+T} \) into the future. We model the system dynamics as the interaction of \( N_c \) nodes represented as a “Sensorimotor Graph”, this graph has nodes \( V = \{ v_k \in \mathbb{R}^n \} \), (representing e.g.: keypoints on a soft gripper), and a set of discrete edges, \( z_{ij} \) which represent the existence or non-existence of a relation between two nodes.

The system input vector, \( x_t \) is given by the concatenation of the node state, \( v_k^t \) with the finger contraction actuation signals \( a^t \). To be driven by external actuation signals, we consider the actions as part of the node input. This allows us to either map specific actions to specific nodes (providing the same action for the nodes belonging to the same finger), or provide all actions to all nodes and let the model infer the mapping between nodes and actuation signals. We consider the more general case where the model infers the mapping between nodes and actions.

In this work we propose to decouple the problem in two parts: 1) given node states, \( v^{1:t} \) and actuation signals, \( u^{1:t+T} \) infer the Sensorimotor Graph, \( S \), 2) then, given \( S \), an initial state \( v^t \), and the actuation sequence, \( u^{t+1:t+T} \), predict a future sequence, \( v^{t+1:t+T} \).

III. MODEL

In this work we consider a framework for modelling and self-calibrating a robotic soft hand non-rigid kinematic chain. The model is inspired by two key ideas: 1) Sensorimotor babbling enables infants to understand their own kinematic structure and 2) we want to explicitly consider this physical structure, which we call the Sensorimotor Graph as an inductive bias in the prediction process.

For estimating this Sensorimotor Graph we integrate the Neural Relational Inference (NRI) model proposed in [13], which enables us to: 1) consider an explicit connectivity graph, 2) estimate this connectivity graph in an unsupervised manner while learning the system dynamics.

A. Structure

The model is defined in two parts. A first part receives a sequence of system observations and infers the underlying edge structure, and a second part which takes this edge structure and system state, predicting the future state trajectory. We call these parts the encoder and decoder, respectively. Both these parts are trained jointly, either by providing ground truth labels for the edges or by learning them in an unsupervised manner end-to-end.

B. Encoder

Following the notation in [13], the encoder takes a sequence of observed node trajectories and actuations to infer pair-wise interaction probabilities represented as edges \( z_{ij} \).

More formally, the encoder models the edge existence probability distribution, \( q_\phi (z_{ij} | x) \), as follows:

\[
q_\phi (z_{ij} | x) = \text{softmax} (f_{enc,\phi} (x)_{ij:1:N_c})
\]

(3)

Where \( f_{enc,\phi} (x) \) is a GNN acting on the fully-connected graph defined by all node pair permutations. The encoder computes iterative node-to-edge and edge-to-node message passing operations to simulate node interactions and then
calculates the edge type. The sequential node and edge updates follow:

\[ h^1_j = f_{emb} (x_j) \]
\[ v \rightarrow e : \quad h^1_{(i,j)} = f^e_1 (\{h^1_i, h^1_j\}) \]
\[ e \rightarrow v : \quad h^2_j = f^v_1 \left( \sum_{i \neq j} h^1_{(i,j)} \right) \]
\[ v \rightarrow e : \quad h^2_{(i,j)} = f^e_2 (\{h^2_i, h^2_j\}) \] (4)

More specifically, in the first pass the node to edge messages are computed using an embedding function, \( f_{emb} \). Embedded node values from nodes \( i \) and \( j \) are then used to update the edge \( e_{i,j} \) between them. After updating the edge state, another embedding function, \( f_e \), takes the edge value and uses it to compute edge-to-node messages. The edge-to-node messages are aggregated for all edges connected to node \( i \). Then, a final node-to-edge message passing operation infers the final edge value from which the edge existence probability is calculated using a softmax function.

\[ z_{ij} = \text{softmax} \left( \frac{h^2_{(i,j)} + g}{\tau} \right) \] (5)

Given the distribution, it is possible to either train the model in a supervised setting by providing ground truth labels for the edges, or training the model end-to-end by sampling edges from the inferred distribution and using these as graph structure for the subsequent trajectory prediction step. The NRI model uses a continuous approximation of the discrete distribution to effectively sample from a discrete distribution. Here, \( g \in \mathbb{R}^{K} \) is a vector of i.i.d. samples drawn from a Gumbel(0,1) distribution and \( \tau \) is a temperature parameter that adjusts the "softness" of the resulting distribution (when \( \tau \to 0 \), the distribution converges to one-hot samples, as described in [13]).

C. Decoder

The second part, called the decoder, takes the edges and uses them as explicit structure for the graph computations in the trajectory prediction step. Just like the encoder, the trajectory predictions are based on a sequence of node-to-edge and edge-to-node passes. An initial accumulation function updates the edge value by weighting adjacent nodes by the edge existence probability, where \( z_{i,j,k} \) represents the \( k \)-th entry of the vector \( z_{i,j} \) and \( f^e_k \) is the embedding function associated to this edge type \( k \). Then, edges are aggregated in the nodes through an order invariant summation and transformed by an embedding function, \( f_v \). Adding them to the current node state outputs the mean value, \( \mu_{j}^{t+1} \) for the normal distribution from which the next node state, \( x_{j}^{t+1} \) is sampled, with variance \( \sigma^2 \).
B. Training and Validation Sets

A total of six different data sets are extracted from the soft gripper simulation. The training set for all tests includes only four fingers in a total of thirty different configurations (relative positions) and with four different values of elasticity (Poisson ratio and Young modulus). We will reference this set as Trainset. For each sample, the type of motion applied to the cable pull actuation signal for every finger is the same, although the characteristics of the motion - amplitude, frequency and phase - are different and randomized. Three different types of motion are used (Fig. 2b). Our first set, TestBase, includes novel configurations, some combinations of motion and elasticity unseen at training time, as well as a different cable pull actuation signal.

We created four sets to test the model in different validation sets with single varying conditions. The validation sets are setup to understand: 1) TestMotion: How well the model generalizes to previously unseen motions. 2) TestConfig: Effect of system configuration (relative position of fingers) deviating from the training configurations, either due to sensor measurement errors or due to changes in the end-effector, 3) TestFingers: Adaptation to changes in the number of fingers, either due to mechanical failure of one of the joints or sensor measurements failing to provide the position for any of the fingers or points, 4) TestShuffle: Robustness to changes in the order of the joint measurements, either due to tracking errors or wrong correspondences between detected parts.

More specifically the four validation sets are used to test the models generalization to motion, relative position, number of fingers and order of the points, respectively. The first test set, TestMotion, uses three different motions (resulting from operations between trigonometric functions) and includes random noise. The second test set, TestConfig, uses the same motions and elasticities as the training set but for thirteen unseen configurations. The third test set, TestFingers, uses only three-finger configurations with the same motions and elasticities whereas the fourth, TestShuffle, and last validation set, shuffles the order of the twelve nodes for each sample.

C. Experiments

The model is evaluated in two separate sets of experiments: 1) comparison to other methods proposed in the literature and 2) ablation studies to validate the robustness of the proposed approach under different conditions.

The first part looks at our approach in relation to baseline models. In this set of experiments the following models are quantitatively compared: 1) Non-learning method 2) Linear, 3) Multi-layer Perceptron (MLP), 4) Long Short Term Memory sequence model (LSTM), 5) Neural Relational Inference (NRI). In the non-learning method, the position remains the same as the last known point for the whole prediction time. We consider the linear and MLP multi-layer perceptron models as they validate if the task can be solved by simple linear or non-linear interpolation of the trajectory points. Long Short Term Memory model was chosen as being representative of a non-structured model, enabling us to compare the performance of a structured sequence model with a non-structured sequence model.

The second part looks more closely at how different conditions affect the model under analysis (ablation study). For this purpose, only the most relevant baselines are considered, alongside a structured model.

V. Results and Discussion

A. Setup

The experiments are structured around two parts. The first, is a quantitative comparison of how the different models perform over the first validation set, TestBase. The second, evaluates how each source of variability affects the different models and how they react to different types of generalization.

B. Model Comparison

For the first experiment, the structure-based approach (NRI) is compared to three different baselines: Linear, MLP and LSTM. The NRI model has two model variations on the decoder part of the model, it can either be based on a MLP (NRI-mlp) or an RNN structure (NRI-rnn), in this experiment we consider both variants. For each of the possible decoder structures the encoder has two additional variations: supervised and unsupervised. The supervised encoder receives
ground truth edges for the connections between the finger nodes during training where as the unsupervised learns the connections in an end-to-end manner while learning to predict future trajectories.

**TABLE II**

| Model   | MSE\(_{10}\) | MSE\(_{100}\) | MSE\(_{1000}\) |
|---------|--------------|--------------|---------------|
| Linear  | 7.987        | 1.268        | 1.839         |
| MLP     | 7.878        | 1.117        | 1.608         |
| LSTM    | 6.157        | 1.354        | 1.964         |
| uns. NRI-mlp | 4.962      | 1.336        | 1.926         |
| uns. NRI-rnn | 2.200      | 0.801        | 1.142         |
| sup. NRI-mlp | 4.753      | 1.281        | 1.846         |
| sup. NRI-rnn | 2.108      | 0.746        | 1.063         |

In this experiment, all models are trained for 10 predictions steps. In the variations of the NRI with no ground-truth edge information, a non-overlapping set of fifty time steps is provided to the encoder to estimate the connectivity graph. For all the baselines and for the NRI-rnn, there is an initialization or "burn-in" phase where the models have access to the five time steps before the sequence that is considered for the evaluation. The LSTM "burn-in" phase, similarly to the NRI, has fifty steps before the ten step prediction window. This allows the LSTM to initialize its internal state.

Table II shows how the proposed model compares to different baseline models, where the Mean Squared Error (MSE) represents the deviation between the prediction and the real trajectory and MSE\(_{1000}\) stands for the MSE normalized to the travelled distance (MSE\(_{1000}\)=1 means that the prediction of one point has deviated as much as the distance the point travelled). The values in the tables may appear low, but this is due to the node positions being normalized with respect to the data-set statistics. Therefore, this MSE has no direct physical meaning, this metric allows us to capture the quantitative performance difference between the models. We add a non-learning model to provide a baseline for the error when the predicted trajectory stands still. Here we can see that the proposed model (NRI) outperforms the baseline models in the first testset, TestBase, which has a diverse set of motions, configurations and elasticities. We will look at this more closely in the ablation studies.

C. Ablation Study

We will now look at the second set of experiments (Tab. III) which seek to see how the model performs under different input distribution shifts. Models are all trained for 10 steps and test sets are 25 steps long. The first testset, TestBase, establishes a baseline for how well models perform in auto-regressively predicting more steps ahead than they were trained on. The four test-sets introduced in the experimental setup all look at different variations on the input data. Refer to Table 1 for an overview of the different test-sets. This enables us to study the effects of the different sources of prediction error separately. For this ablation study we will consider the best performing NRI model (NRI-rnn) and two relevant baselines, LSTM and MLP.

**TABLE III**

| Ablation study for different input conditions |
|---------------------------------------------|
| [x10\(^{-4}\)] | TestBase | TestMot. | TestConf. | TestFing. | TestShuff. |
| MLP             | 2.250    | 1.051    | 3.090     | -         | -          |
| LSTM            | 4.762    | 6.786    | 3.015     | 2.843     | 3.323      |
| NRI             | 3.303    | 4.482    | 2.056     | 1.469     | 1.502      |

From the initial baseline on TestBase, we can see that the NRI performs worse than the baseline models when generalizing to the distribution shift introduced by predicting longer sequences. In this first Testset, the MLP seems to be capable of predicting longer sequences with a lower mean error. The model also proves to have good generalization to new motions. This advantage of the MLP compared to the structure-based models can be justified with the fact that it has fewer model parameters and is able to exploit the relations between the nodes in adjacent positions of the input vector. The NRI, due to its order invariant graph-based structure, only has access to the nodes without explicit order like in a vector-based representation of the nodes.
This, however, makes the MLP quite fragile to changes or disturbances in the data. Testing the same model in a situation with new relative positions of the fingers, Test-Config, MLP’s prediction error increases considerably. Under the same conditions, we can see the NRI outperforming the baseline models. Under the TestFingers dataset, the MLP is not able to generalize and has an error significantly higher than the other models and is therefore not considered in this validation set. We can again see the NRI outperforming the baseline models in this condition. The last testset, TestShuffled, considers the situation where points are shuffled and the points at test are in a different order than the configurations it was trained on. This models a tracking or re-identification error. Here, again, we can see that, due to its order invariance properties, the NRI model outperforms the baseline models.

D. Connectivity

The model can be trained both in an unsupervised and supervised manner. In this section, we look at how well the model performs in recovering the edges when trained in both these regimes. For this, we trained the model with and without labels and measured the accuracy of the edge predictions given node trajectories against a held out ground-truth label set. In Table IV we see that: 1) connectivity inference contributes positively to the dynamics prediction, 2) supervised training recovers g.t. connectivity and outperforms the unsupervised variant, 3) unsupervised training does not usually recover ground-truth connectivity. This last phenomena happens since: 3.1) unsupervised edge estimation approach does not see any labels for the edge types and might arrive at a similar, but opposite, correspondence, 3.2) intricate counter-intuitive relations in the graph emerge, particularly finger-to-finger connections, similar to what was pointed out in [13], [21]. A future direction to encourage the model to infer the real physical connectivity could be to impose sparsity constraints on the edges and consider the possibility of curriculum learning by exploring the finger synergies from a more coarse to a fine level.

| TABLE IV Connectivity inference performance |
|---------------------------------------------|
| NRI-rnn                                    |
| training | $\text{MSE}_{10}$ [$10^{-4}$] | accuracy | F1score |
| supervised | 7.499 | 0.998 | 0.991 |
| unsupervised | 8.062 | 0.486 | 0.176 |

Moreover, we test increasing the number of edge types from 2 to 3 (for the supervised variant we encode an extra relation type as the opposite direction of the existing edge). In Table V we notice the performance of both the supervised and unsupervised NRI-rnn marginally improve when we have this additional degree of freedom.

E. Prediction step

Finally, we consider training the Sensorimotor Graph model for different prediction sizes. This hyper-parameter sets the size of the prediction length in training and test. The prediction step should be big enough to contribute to a relevant forecast (used in model-predictive control) but not too large as this may result in unstable and chaotic dynamics. All models are trained with teacher forcing and predict the positions auto-regressively. When training the model for larger sequences, as we can see in Table IV, the model performed better at test time as well. Moreover, it is also noticeable that increasing the prediction step does not compromise accuracy in the initial time steps: larger windows benefit predictions right from the start. The comparative inference speed of the NRI model is depicted in Table VII. For comparison, all models were benchmarked on a single core of an i7 2.6GHz CPU.

| TABLE V Influence of the number of edge types parameter in the performance of the decoder |
|-----------------------------------------------|
| NRI-rnn [$10^{-4}$] |
| d | training | $\text{MSE}_{10}$ |
| 2 | supervised | 8.045 |
| unsupervised | 8.206 |
| 3 | supervised | 7.588 |
| unsupervised | 8.083 |

| TABLE VI Influence of prediction length parameter during training in prediction performance |
|-----------------------------------------------|
| [10^{-3}] | PS$_5$ | PS$_{10}$ | PS$_{15}$ | PS$_{20}$ | PS$_{25}$ |
| MSE$_5$ | 0.115 | 0.114 | 0.110 | 0.099 | 0.097 |
| MSE$_{10}$ | 0.840 | 0.825 | 0.785 | 0.688 | 0.647 |
| MSE$_{15}$ | 2.498 | 2.433 | 2.291 | 1.935 | 1.768 |
| MSE$_{20}$ | 5.307 | 5.121 | 4.785 | 3.928 | 3.501 |
| MSE$_{25}$ | 9.020 | 8.615 | 8.008 | 6.518 | 5.602 |

| TABLE VII Inference speed of different models |
|-----------------------------------------------|
| Model | MLP | NRI-mlp | NRI-rnn | LSTM |
| Iteration time (ms) | 0.164 | 99.8 | 643 | 666 |

F. Discussion

We evaluated the model against multiple baseline models in different sets of experiments. In the first experiment (Fig 3) we found that the proposed action-conditioned NRI model outperforms the baseline models on TestBase with a test time prediction step of 10. This data-set tests for small variations in system attributes and validates the robustness of the proposed model. Furthermore, from this first set of experiments we can see that the supervised RNN variant of the NRI model outperforms the other variants.
In the second set of experiments, we found that the NRI model performs worse than the baseline models when tested to auto-regressively predict longer sequences on a TestBase testset of length 25. When tested under different configurations or number of nodes, it outperforms the baseline models, validating the robustness to sensor noise. Additionally, when tested on a test-set of shuffled points, it shows it is able to reconstruct the connectivity graph and outperform the baseline models, validating robustness to re-identification or tracking issues.

VI. CONCLUSIONS AND FUTURE WORK

In this work we propose applying a GNN - the “Sensorimotor Graph” model - to the domain of modelling a robotic soft hand. More specifically, we described the underlying compositional assumptions of the system and showed how a structured GNN successfully satisfies them. We denote our model as the “Sensorimotor Graph” as it is inspired on the way human infants acquire sophisticated motor capabilities (e.g. of the human hand) through sensorimotor learning and motor babbling. We chose the Neural Relation Inference as a candidate GNN for the problem domain. We showed how the model’s explicit graph representation enables us to exploit the underlying physical structure of the system. To evaluate the model, we benchmarked it against non-structured baselines and evaluated the model robustness to gradually more adverse conditions.

We found that the model performs well under conditions which require generalizing to changes in structure. More specifically, it performed well under conditions where the number and location of the fingers changed. Furthermore, we looked at different variations of the model and observed that out of two model variations on the encoder architecture, supervised and unsupervised, the supervised model performs better.

We showed that GNNs, more specifically the Neural Relational Inference model, can be used to robustly model a system of interacting soft material parts in the form of a differentiable dynamics model. This opens up promising avenues for combining the learnt model with a non-convex optimal control framework, using the differentiable dynamics model as systems constraints.

We believe that the encouraging results and generalization capabilities of our “Sensorimotor Graph” model pave the way to the ability to acquire the structure and dynamics of complex systems and will be of utmost importance for the deployment of complex non-rigid kinematic chains and affordable articulated end-effectors such as soft hands for advanced manipulation.

ACKNOWLEDGMENT

This work was supported by a doctoral grant from FCT (SFRH/BD/139092/2018).

REFERENCES

[1] T. George Thuruthel, Y. Ansari, E. Falotico, and C. Laschi, “Control Strategies for Soft Robotic Manipulators: A Survey,” Soft Robotics, vol. 5, no. 2, pp. 149–163, Apr. 2018. [Online]. Available: https://www.liebertpub.com/doi/10.1089/soro.2017.0007

[2] C. von Hofsten, “An action perspective on motor development,” Trends in cognitive sciences, vol. 8, no. 6, p. 266–272, June 2004.

[3] B. S. Homberg, R. K. Katzschmann, M. R. Dogar, and D. Rus, “Haptic identification of objects using a modular soft robotic gripper,” in 2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2015, pp. 1698–1705.

[4] F. Faure, C. Duriez, H. Delingette, J. Allard, B. Gilles, S. Marchesseau, H. Talbot, H. Courtecuisse, G. Bousquet, I. Peterlik, and S. Cotin, SOFA: A Multi-Model Framework for Interactive Physical Simulation, 06 2012, vol. 11.

[5] B. Amos, I. Jimenez, J. Sacks, B. Boots, and J. Z. Kolter, “Differentiable npe for end-to-end planning and control,” in Advances in Neural Information Processing Systems 31, S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, Eds. Curran Associates, Inc., 2018, pp. 8289–8300.

[6] J. L. Pons, R. Ceres, and F. Pfeiffer, “Multifingered dextrous robotics hand design and control: A review,” Robotics, vol. 17, no. 6, p. 661–674, Nov. 1999.

[7] T. Hsia, “Adaptive control of robot manipulators - a review,” in Proceedings. 1986 IEEE International Conference on Robotics and Automation, vol. 3, 1986, pp. 183–189.

[8] E. Piccione, L. Siegwart, and S. Levine, “Self-supervised visual planning with temporal skip connections,” in 1st Annual Conference on Robot Learning, CoRL 2017, Mountain View, California, USA, November 13-15, 2017. Proceedings, ser. Proceedings of Machine Learning Research, vol. 78. PMLR, 2017, pp. 344–356.

[9] H. Fafnir, T. Lillicrap, I. Fischer, R. Villegas, D. Ha, H. Lee, and J. Davidson, “Learning latent dynamics for planning from pixels,” in ICLR, 2019.

[10] A. S. Polychronis and L. Nalpantidis, “Survey of Model-Based Reinforcement Learning: Applications on Robotics,” Journal of Intelligent & Robotic Systems, vol. 86, no. 2, pp. 153–173, May 2017. [Online]. Available: http://link.springer.com/10.1007/s10846-017-0468-y

[11] J. Kober, J. Bagnell, and J. Peters, “Reinforcement learning in robotics: A survey,” The International Journal of Robotics Research, vol. 32, no. 13–14, pp. 1238 – 1274, 2013.

[12] Y. Li, J. Wu, R. Tedrake, J. B. Tenenbaum, and A. Torralba, “Learning particle dynamics for manipulating rigid bodies, deformable objects, and fluids,” in ICLR, 2019.

[13] T. Fetaya, E. Wang, K.-C. Welling, M. Zemel, T. Kipf, E. Fetaya, K.-C. Wang, M. Welling, and R. Zemel, “Neural relational inference for interacting systems,” ICML, 2018.

[14] P. Battaglia, J. B. Hamrick, V. Bapst, A. Sanchez-Gonzalez, V. Zambaldi, M. Malinowski, A. Tacchetti, D. Raposo, A. Santoro, R. Faulkner, Çağlar Gülçehre, H. Song, A. Ballard, J. Gilmer, G. Dahl, A. Vaswani, K. R. Allen, C. Nash, V. Langston, C. Dyer, N. Heess, D. Wierstra, P. Kohli, M. Botvinick, O. Vinyals, Y. Li, and R. Pascanu, “Relational inductive biases, deep learning, and graph networks,” ArXiv, vol. abs/1806.01261, 2018.

[15] A. Sanchez-Gonzalez, J. Godwin, T. Pfaff, R. Ying, J. Leskovec, and P. Battaglia, “Learning to simulate complex physics with graph networks,” ArXiv, vol. abs/2002.09405, 2020.

[16] F. Scarselli, M. Gori, A. C. Tsoi, M. Hagenbuchner, and G. Monfar-dini, “The graph neural network model,” IEEE Transactions on Neural Networks, vol. 20, no. 1, pp. 61–80, 2009.

[17] J. Gilmer, S. M. Schoenholz, P. F. Riley, O. Vinyals, and G. E. Dahl, “Neural message passing for quantum chemistry,” in Proceedings of the 34th International Conference on Machine Learning - Volume 70, ser. ICML’17, 2017, p. 1263–1272.

[18] P. W. Battaglia, R. Pascanu, M. Lai, D. J. Rezende, and P. Battaglia, “Learning to simulate complex physics with graph networks,” ArXiv, vol. abs/1612.00222, 2016.

[19] S. van Steenkiste, M. Chang, K. Greff, and J. Schmidhuber, “Relational neural expectation maximization: Unsupervised discovery of objects and their interactions,” in 6th International Conference on Learning Representations, 2018.

[20] D. Holland, E. Park, P. Polygerinos, G. Bennett, and C. Walsh, “The soft robotics toolkit: Shared resources for research and design,” Soft Robotics, vol. 1, pp. 224–230, 09 2014.

[21] A. Jain, A. R. Zamir, S. Savarese, and A. Saxena, “Structural-rnn: Deep learning on spatio-temporal graphs,” in IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016.
This figure "cycle.png" is available in "png" format from:

http://arxiv.org/ps/2107.08492v1