Learning Structured World Models From and For Physical Interactions

by

Yunzhu Li

B.S., Peking University (2017)
S.M., Massachusetts Institute of Technology (2020)

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

September 2022

Author .....
Department of Electrical Engineering and Computer Science
August 10, 2022

Certified by .....
Antonio Torralba
Delta Electronics Professor of Electrical Engineering and Computer Science
Thesis Supervisor

Certified by .....
Russ Tedrake
Toyota Professor of Electrical Engineering and Computer Science,
Aeronautics and Astronautics, and Mechanical Engineering
Thesis Supervisor

Accepted by .....
Leslie A. Kolodziejski
Professor of Electrical Engineering and Computer Science
Chair, Department Committee on Graduate Students

Learning Structured World Models From and For Physical Interactions

by

Yunzhu Li

Submitted to the Department of Electrical Engineering and Computer Science
on August 10, 2022, in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy

Abstract

Humans have a strong intuitive understanding of the physical world. We observe and interact with the environment through multiple sensory modalities and build a mental model that predicts how the world would change if we applied a specific action (i.e., intuitive physics). This dissertation presents my research that draws on insights from humans and develops model-based reinforcement learning (RL) agents. The agents learn from their interactions and build predictive models of the environment that generalize widely across a range of objects made with different materials. The core idea behind my research is to introduce novel representations and integrate structural priors into the learning systems to model the dynamics at different levels of abstraction. I will discuss how we can make structural inferences about the underlying environment. I will also show how such structures can make model-based planning algorithms more effective and help robots to accomplish complicated manipulation tasks (e.g., manipulating an object pile, pouring a cup of water, and shaping deformable foam into a target configuration). Beyond visual perception, touch also plays a vital role when humans perform physical interactions. I will discuss how we bridge the sensing gap between humans and robots by building multi-modal sensing platforms with dense tactile sensors in various forms (e.g., gloves, socks, vests, and robot sleeves) and how they can lead to more structured and physically grounded models of the world.

This dissertation consists of three parts. In Part I, we show how we can learn world models at different levels of abstractions and how the learned models allow model-based planning to accomplish challenging robotic manipulation tasks both in simulation and in the real world. Part II investigates the use of a learned structured world model for physical inference that infers the causal relationships between different components within the environment and performs state and parameter estimations. Part III goes beyond the previous two parts that only assume vision as input by considering touch as an additional sensory modality. I will discuss the novel tactile sensors we developed and how they can be used in understanding hand-object and human-environment physical interactions.

Thesis Supervisor: Antonio Torralba
Title: Delta Electronics Professor of Electrical Engineering and Computer Science

Thesis Supervisor: Russ Tedrake
Title: Toyota Professor of Electrical Engineering and Computer Science,
Aeronautics and Astronautics, and Mechanical Engineering

Acknowledgments

First and foremost, I would love to express my sincere gratitude to my advisors, Antonio Torralba and Russ Tedrake. Antonio has been one of the most creative people I have ever known, and I am constantly amazed by all the brilliant, sometimes crazy, ideas he came up with during our discussions. Antonio has been an amazingly encouraging advisor who enthusiastically supports all my research adventures and is also extremely sharp in terms of spotting research insights. He constantly told me the importance of visualization and how a simple but creative visualization can address issues that take hours or even days of debugging, which also led to some of the most active research directions in the lab. I still remember the time when we are working on the tactile glove project. I was stuck on how I could interpret the data. It was precisely in his office that he pulled out the MATLAB and started to make plots showing the statistics of the collected data. Within just a few minutes, some interesting patterns with strong physical interpretations emerge, which paved the way for making this paper accepted by *Nature*. I am grateful for being a student of Antonio's, and I will continue to use all the lessons he taught me throughout my research journey.

Like Antonio, Russ is also one of my role models. I came to the lab with very little robotics background. Russ taught me how to think from a roboticist's perspective, and I have been constantly impressed by how sharp Russ is about various problems. Russ's sharpness about problems comes from years of focusing on every detail of the robotic systems, where he builds systems from scratch, and he may be the one who codes the most among all senior professors I know. Russ is also an excellent teacher. Besides his enthusiasm about teaching, he once told me that teaching also keeps him sharp as he will have to go through every detail of the course, the equations, the derivations, and the implementation. As a result, whenever I see a robotics paper that seems to be hyped up, I would always want to learn Russ's opinion about it. This sharpness and deep understanding of the details allows him to be extremely successful not only in academia but also in the industry, where he leads teams producing amazing results keeping both the Robot Locomotion Group at MIT and the company's research lab at the forefront of robotics. Being able to have a profound impact in both academia and industry is something I wish the most for myself, and Russ sets an excellent role model of how it works out.

I would also like to thank Wojciech Matusik for serving on my thesis committee. Wojciech

has been a long-term collaborator of mine since the first year of my Ph.D. By working with Wojciech, I have realized how interesting a research project can be by incorporating knowledge from multiple disciplines and how some of the major progress can happen at precisely the intersection of multiple areas. I have always been constantly amazed by the diversity of knowledge Wojciech has, which is nicely reflected by his group's diversified research directions. For example, in Wojciech's group, there are people working on computational fabrication, manufacturing hardware, reinforcement learning, building physics-based simulators and making them differentiable, touch & knitting, and graphics & computational photography. You name an area, and there may be people working on it. I will have to continue learning from Wojciech how to diversify a group and maintain a good balance between different directions. I would also like to thank Wojciech for the tremendous help he offered during my faculty search.

Next, I want to offer special thanks to Jiajun Wu and Jun-Yan Zhu, especially for their help at the beginning of my Ph.D., when I did not have much research experience. They have walked me through all the stages of a research cycle, which gave me a much deeper understanding of high-level questions like how to select a research direction as well as low-level details like how to write a paper, make figures, respond to the reviewers, and deliver the presentation. Their help not only built the initial momentum of my Ph.D. work, but also made me understand what a good student mentor can be like, which turned out to be extremely valuable later when I started to move on to the supervising role. They are currently assistant professors at Stanford and CMU, respectively. Their success stories also made me determined to try the faculty job market.

I would also like to thank my undergraduate advisors at Peking University, Yizhou Wang and Tianfu Wu, who opened the gate of research and drew me into the research fields of computer vision and machine learning. I am also grateful to my advisors at Stanford University, Sebastian Thrun, Andre Esteva, and Stefano Ermon, for their tremendous support of my explorations at the beginning of my research career. The research internship at Stanford during my undergraduate was the first time I came to the United States. Thanks to them, it was an incredible seven-month experience immersed in the cutting edge of artificial intelligence, which made me determined to continue my research journey as a Ph.D. student.

Sincere thanks also go to all the senior mentors who have offered tremendous help and guidance during my time within and outside MIT: Dieter Fox, Joshua B. Tenenbaum,

Animesh Garg, Anima Anandkumar, Pulkit Agrawal, Abhinav Gupta, Daniel L. K. Yamins, David Held, Dina Katabi, Vincent Sitzmann, Wenzhen Yuan, Chuang Gan, Hao Su, Fei-Fei Li. Discussions and/or collaborations with them have profoundly shaped my view about different research fields and specific research directions. Moreover, their attitude and rigor about research have always been a great source of inspiration for me, making me constantly reflect on my research journey and steady the progress of my academic expedition.

I have also been fortunate to collaborate with many extremely talented colleagues and co-authors during my Ph.D. Many of them have greatly impacted the direction and momentum of my research. They have also significantly contributed to the works I will discuss in the dissertation: Yiyue Luo, Shuang Li, Kexin Yi, Hao He, Subramanian Sundaram, Petr Kellnhofer, Lucas Manuelli, Pete Florence, Wan Shou, Michael Foshey, Toru Lin, Daniel M. Bear, Huazhe Xu, Zhiao Huang, Haochen Shi, Danny Dreiss, Qiang Zhang, Pratyusha Sharma, Shaoxiong Wang, Tomas Palacios, Xingyu Lin, Hsiao-Yu Tung, Haotian Xue, Zhenfang Chen, Mingyu Ding, Lujie Yang, Kaiqing Zhang, Alexandre Amice, Marc Toussaint, Lirui Wang, Yonglong Tian, Katerina Fragkiadaki, Carl Qi, Joseph DelPreto, Chao Liu, Beichen Li, Daniela Rus, Junchi Yan, Chaofei Fan, Damian Mrowca, Seth Alter, Aran Nayebi, Jeremy Schwartz, Pushmeet Kohli, Ingmar Schubert, Yunchu Zhang, Zherong Pan, Kris Hauser, Andy Zeng, Jingjin Yu. I really appreciate their contributions to our work and all the lessons/knowledge they have taught me.

I would also like to thank the awesome labmates from Antonio's group: Wei-Chiu Ma, Bolei Zhou, Hang Zhao, Xavier Puig Fernandez, Ching-Yao Chuang, Hengshuang Zhao, Tongzhou Wang, Tianmin Shu, David Bau, Dim P. Papadopoulos, Jonas Wulff, Joanna Materzynska, Manel Baradad, Sarah Schwettmann, Adrià Recasens, Nadiia Chepurko. I have learned so much from them about the state-of-the-art of computer vision. I have also enjoyed the privilege of being part of Russ's Robot Locomotion Group with amazing colleagues: Wei Gao, Terry Suh, Abhishek Gupta, Jack Umenberger, Tobia Marcucci, Shen Shen, Vincent Tjeng, Tao Pang, Hongkai Dai, Greg Izatt, Robin Deits, Twan Koolen, Katy Muhlrad, Matt O'Kelly, Steve Proulx, Sadra Sadraddini, Boyuan Chen, Mark Petersen, Max Simchowitz, Michelle Tan, Albert Wu. All of them have offered tremendous support and friendship during my PhD. Mainly because of them, I have really enjoyed my everyday life here at MIT!

I have also received tremendous support from people from various institutions during my faculty job search. In addition to the people I mentioned above, I would also like to

thank Bill Freeman, Yuke Zhu, Danfei Xu, Qixing Huang, Tao Du, Mingmin Zhao, Vikash Kumar, Chen Feng, Shenlong Wang, Mani Srivastava, Lin Yang, Yang Zhang, Howie Choset, Rob Fergus, Lerrel Pinto, Bo Li, Derek Hoiem, Svetlana Lazebnik, David Forsyth, Nancy M. Amato, Julia Hockenmaier, Alex Schwing, Saurabh Gupta, Yuxiong Wang, Han Zhao, Nan Jiang, Yang Song, Shengjia Zhao, Xiaolong Wang, Rex Ying, Farshad Khorrami, Qi Sun. It was their help that helped me navigate the stressful but rewarding process, and I really appreciate it.

The assistance from the fantastic staff members at MIT has also made my Ph.D. life much easier and more enjoyable. I would like to share special thanks to Mieke Moran, Fern Keniston, Janet E. Fischer, and the Infrastructure Group (TIG) at MIT CSAIL for their help through different stages of my Ph.D.

Lastly, I would like to thank my parents, Dejun Li and Xiaohong Song, for their unconditional love. They have poured so much energy and effort into ensuring I have the proper upbringing and access to good education. It is their wholehearted support and unwavering encouragement that allows me to live a happy childhood and pursue my passion for research in a different country. I will be forever grateful for their presence in my life and everything they have done to make me a better person!

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Humans have a strong intuitive understanding of their surrounding physical world (Figure 1-1). We observe and interact with the environment through multiple sensory modalities, including vision and touch. During the interactions, our brains process the sensory data, build a mental model of the world, and predict how the environment would change if we applied a specific action (i.e., intuitive physics). This predictive ability does not rely on analytical physics equations yet applies to objects of different materials (e.g., rigid bodies, deformable objects, and fluids), enabling a tremendous amount of interactive skills far superior to those of current robotic systems [Lake et al., 2017].

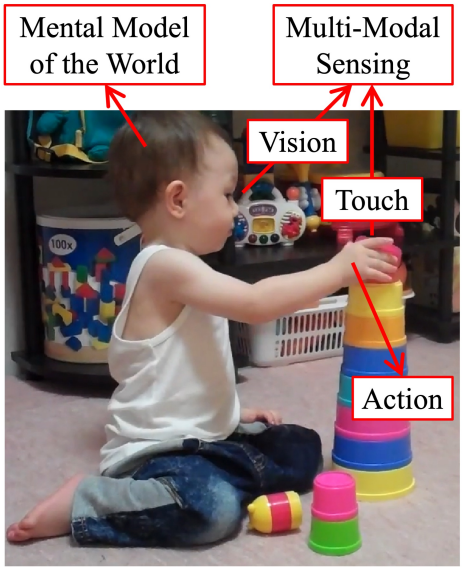


Figure 1-1: **Learning to perform physical interactions.** Humans perceive the world via multiple sensory modalities (e.g., vision and touch), and through interactions, we build a mental model of the world that predicts how the environment would change if we apply a specific action. We then use the learned predictive model in our brain to plan our actions to achieve specific goals.

It is, therefore, desirable to build predictive models of the environment from observations to help the robots understand the effect of their actions and make effective plans to achieve a given task goal. Physics-based models [Hogan and Rodriguez, 2016, Zhou et al., 2019] have excellent generalization ability yet typically require full-state information of the environment, which is hard and sometimes impossible to obtain in complicated robotic (manipulation) tasks. Learning-based dynamics modeling circumvents the problem by learning a predictive model directly from

raw sensory observations, and recent successes are rooted in the use of deep neural networks (DNNs) as the functional class [Finn and Levine, 2017, Ha and Schmidhuber, 2018].

This thesis aims to significantly expand upon the direction of learning world models from observations. I will show how we can develop robotic systems that learn from their physical interactions and build predictive models of the environment that can generalize widely across a diverse set of problems. To accomplish this kind of generalization, I will discuss how to bring together the flexibility of deep neural network models with structured inductive biases, which enables broad generalization to a variety of tasks and environments. In doing so, I will present a series of my works that have been among the first to take robot learning from the manipulation of rigid objects to the manipulation of a variety of dynamic and flexible objects (e.g., fluids and deformable or granular materials), unlocking a fundamentally new set of capabilities. The research theme, **learning structured world models from and for physical interactions**, requires learning algorithms and infrastructures vastly different from traditional model-based pipelines and unstructured reinforcement learning systems. Significant challenges exist in building inductive biases at the correct level of abstraction across sensing, perception, dynamics modeling, and optimization that form a coherent system for effective training and real-world deployment.

1.1 Problem Statement

To make the problem more concrete and more formally characterize the benefits of the proposed methods, we define the problem we aim to solve as the following. The goal is to minimize an objective function $\sum_t c(y_t, u_t)$ defined on the observation y_t and the action u_t over some time horizon. y_t is the raw sensory observation we get from different types of sensors at time t, which is potentially very high-dimensional in the form of RGBD/tactile images. u_t is the action we input to the system like the robot actions:

$$\min_{\pi} \mathbb{E}_{\pi} \left[\sum_t c(y_t, u_t) \right] \tag{1.1}$$

$$z_t = g(y_{1:t}, u_{1:t-1}) \tag{1.2}$$

$$z_{t+1} = f(z_t, u_t, \theta) \tag{1.3}$$

$$u_t = \pi(z_t). \tag{1.4}$$

The problem definition contains three very important constraints, and the interpretation of it requires us to think deeply through the following four aspects.

- **Sensing**: y_t denotes the raw sensory information we obtain, which is essentially a (partial) observation of the environment. The sensory data can contain multiple modalities (e.g., RGB images, depth, tactile/audio data) and should include all necessary information to accomplish the downstream interaction tasks. Determining the set of information we need and constructing the data capturing systems are no easy tasks, as many of the critical sensory information (e.g., tactile data) cannot be captured by off-the-shelf sensors. Therefore, besides assembling existing sensors, we are also tasked to develop novel multi-modal perception systems with innovations at both the hardware and software levels that provide detailed and sufficient modeling of the physical interaction process.

- **Perception** (Equation 1.2): The perception module $z_t = g(y_{1:t}, u_{1:t-1})$ maps the observation $y_{1:t}$ and the action $u_{1:t-1}$ into a latent representation of the environment z_t at time t. And you should notice from the problem formulation that the choice of z_t has a profound impact on the structure of the overall system. We should be asking the question: what specific forms should z_t be, and at what levels of abstraction should we represent the objects and the environment. The representation should contain information about the underlying environment sufficient to accomplish the downstream planning/control tasks.

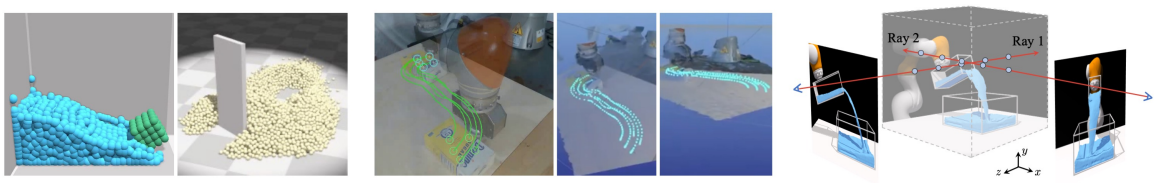
- **Dynamics** (Equation 1.3): The third aspect is the dynamics module $z_{t+1} = f(z_t, u_t, \theta)$ that takes the current representation z_t and the physical parameters θ (e.g., friction coefficient, stiffness, or latent vectors) as inputs. When given an external action u_t, it predicts the changes in the scene representation that describes the evolution of the system. To achieve better generalization, we should think carefully about how we can characterize the structures of the underlying physical world and what model class we should use (e.g., linear models, convolutional neural networks, or graph neural networks), especially when modeling the dynamics for objects of a diverse set of materials. To adapt the models to scenarios where the underlying structure and parameters are not directly observable, we will also have to formulate an inference problem and perform continual structural and parameter estimation from data in an online fashion.

- **Control** (Equation 1.4): Then is the control module $u_t = \pi(z_t)$ that derives the control signal based on the scene representation to minimize the task objective (Equation 1.1). To solve the optimization problem, we need to leverage the structure introduced by the representation and the learned dynamics model while considering the trade-off between effectiveness and efficiency. We will also need to investigate how to close the perception-control loop and consider all practical constraints when deploying the system on real robots.

Making progress in solving these problems requires a lot of systematic thinking, careful design of the connections between different components, and a thorough understanding of the advantages and limitations of various modeling choices. This is also an interdisciplinary area requiring knowledge/advances from multiple disciplines. The following section will provide an overview of the systems I built to advance the robots' capabilities and how I approach the problem by decomposing it into three main attacking angles.

1.2 Approach: Learning Structured World Models

To construct the system and answer the questions above, my approach leverages advances in robotics, computer vision, and machine learning, **integrating structural priors into deep neural networks** to capture the structure of the physical system and provide better generalization ability outside the training distribution (Figure 1-2). For example, my research has been the first to combine particle-based scene representation with novel graph-structured neural networks for the dynamics modeling of complicated objects, such as fluids and deformable foam. Leveraging the structures embedded in the learned models, I constructed and solved the model-based planning problem that enables robots to **accomplish complicated manipulation tasks** (e.g., manipulating a pile of boxes, pouring a cup of water, and shaping deformable foam into a target configuration; Figure 1-3), which goes far beyond the capabilities of prior systems. We have also built dense tactile sensors in various forms (e.g., gloves, socks, vests, and robot sleeves), **constructed multi-modal sensing and learning platforms** to study human-environment interactions, and shown success in full-body pose estimation and hand-object dynamics modeling (Figure 1-4). These sensing platforms allow for more scalable and expressive modeling of the interaction process in the natural environment, which can inject the desired physical priors into our system and build



(a) Particle Representation for Fluids and Granular Materials

(b) Keypoint Representation for the Box's Dynamic Movements in 3D

(c) Neural Implicit Representation for Fluid Dynamics

Figure 1-2: **Scene representation and dynamics modeling.** My research investigates representations of the environment at different levels of abstraction, ranging from object-centric models to keypoints and particles, and has demonstrated success in modeling objects of various materials, including rigid bodies, deformable objects, granular materials, and fluids.

more structured and physically grounded models of the world.

1.2.1 Scene Representation and Dynamics Modeling

As I have discussed before, unlike physics-based models, we humans are not constantly calculating analytical physics equations, such as $F = ma$, in our brains. Instead, we build an intuitive model of the environment that predicts the system's evolution purely from sensory observations. Such predictive ability, while intrinsic to humans and applicable to a wide variety of systems, remains challenging for state-of-the-art computational models. For example, representations commonly used in the literature, such as the 6-DoF pose and abstract latent vectors learned only from 2D supervisions, cannot handle deformable and compositional scenes because they are not expressive enough and generalize very poorly. We need representations that can capture the complexity of these scenes and impose inductive biases grounded to the underlying system (Equation 1.2).

My research introduced models that transform high-dimensional sensory data into structural representations at different levels of abstraction and exploit the structure for dynamics modeling (Equation 1.3). The learned dynamics models are more sample efficient and allow the modeling of dynamic and flexible objects, such as fluids and deformable foams, with a performance that goes far beyond what was possible before.

Keypoints. In Li et al. [2020b] and Manuelli et al. [2021], we proposed the use of object keypoints, which are learned in a self-supervised manner and tracked over time (Figure 1-2b). These keypoints anchor our model-based predictions, and through concrete experimental evidence, we showed that keypoints provide the following appealing properties: (i) the output is interpretable and in 3D space, allowing us to analyze the visual model separately from the predictive model, and (ii) they can apply to deformable objects and (iii) achieve category-level

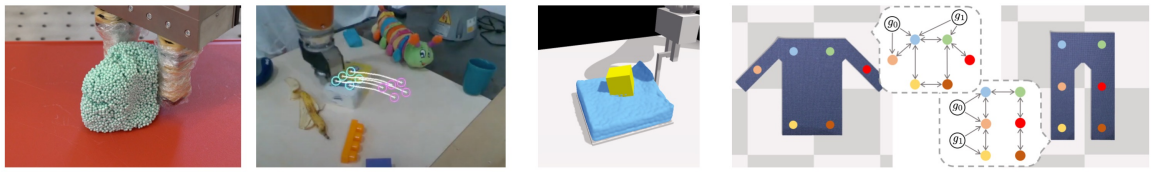
generalization.

Particles. Keypoints are useful as a low-dimensional structural representation but cannot model objects with higher degrees of freedom, such as fluids. In Li et al. [2018, 2020a], we proposed dynamic particle interaction networks (DPI-Nets) to learn a particle-based simulator using graph neural networks (GNNs). Combining GNNs with particle-based systems acts broadly across objects of different materials and injects a strong inductive bias for learning: particles of the same type are governed by the same dynamics (Figure 1-2a). Such structural prior embedded in the model allows more effective training and better extrapolation generalization performance. We showed that our model can predict the dynamics of a wide variety of objects, including rigid bodies and challenging objects such as plasticine and fluids, and generalize to physical systems that are much larger than what the model was originally trained on.

Object-centric and hierarchical representations. For tasks that operate on the object level, we have also investigated object-centric representations and graph-based dynamics models for neuro-symbolic reasoning [Yi et al., 2019]. We further extended it to hierarchical graph representations by mapping visual inputs to Physical Scene Graphs (PSGs) [Bear et al., 2020]. Within the learned graph, nodes represent objects or their parts, and edges represent relationships, in which many aspects of physical understanding become natural to encode, e.g., object permanence and shape constancy.

Latent and implicit representations. Prior works have shown impressive performance in learning world models in abstract latent space from visual observations yet are limited in generalization ability. Therefore, we incorporated the graph-based structural priors into the latent dynamics model and proposed propagation networks (PropNets) [Li et al., 2019b] that assume access only to partial observations and represent the system's state as a graph to capture the underlying compositionality to enable better generalization. Inspired by Koopman operator theory [Koopman, 1931], we took a step further in Li et al. [2019a] by enforcing linear constraints on the latent-space dynamics model in the PropNets for more efficient system identification and model-based optimization.

Besides the structures imposed on the dynamics model, we also seek to incorporate into our model inductive bias in the form of a learning-based differentiable renderer. In a recent paper selected for an **oral presentation** at a premier robot learning conference (i.e., CoRL) [Li et al., 2022], we drew inspirations from advances in neural implicit representations [Mildenhall et al.,



(a) Deformable Object Manipulation (b) Pusher-Slider in Visual Clutter (c) Manipulating Fluid and a Floating Cube (d) Causal Discovery in Physical Systems from Videos

Figure 1-3: **Physical inference and model-based control.** We tackled various downstream tasks by leveraging the structures in the learned dynamics models and powerful optimization tools from learning and control communities, and we demonstrated success in (d) physical scene reasoning and (a–c) manipulating objects made of different materials.

2020]; we combined the differentiable volumetric renderer with an autoencoding framework and a latent dynamics model (Figure 1-2c). The resulting system greatly expands the model's capability and allows for (i) future prediction, (ii) out-of-distribution viewpoint generalization, and (iii) novel view synthesis in dynamic scenes involving complicated interactions between fluids and rigid objects.

Structured representations at different abstraction levels imply different modeling and generalization capabilities. Therefore, it is essential to understand their advantages and limitations and select the one that suits the best with the task at hand. I will also show in the thesis how a suitable choice of representation and structure can lead to better generalization and sample efficiency.

1.2.2 Physical Inference and Model-Based Control

In the previous section, we discussed how representations could be derived from sensory observations and used to learn the dynamics model, which can then be used for downstream inverse tasks, such as physical inference (i.e., estimating the structure and the parameters θ in Equation 1.3) and model-based control (i.e., deriving the action signals in Equation 1.4). For example, we humans can estimate the weight of an object using the discrepancy between our mental model's prediction and the actual observation. We can also plan our behavior by imagining how our actions would change the environment and choosing the actions that produce the desired outcome. Although using forward models for inverse problems is not new and has been the subject of study in the robotics literature for a long time, it has proven challenging to find a general-purpose solution that is flexible and powerful enough to solve contact-rich and dynamic robotic manipulation tasks. In this thesis, I will show that with an appropriate choice of model, we can leverage the structures in the learned dynamics models

and powerful insights gained from stochastic optimization to solve complex physical inference and model-based control problems from raw sensory observations.

Physical inference. In Li et al. [2020b], we considered the task of causal discovery from videos in an end-to-end fashion and without supervision on the ground-truth graph structure (Figure 1-3d), effectively estimating the structure and parameters required by the dynamics model in Equation 1.3. Our experiments show that, without retraining, the causal structure assumed by the model allows it to make counterfactual predictions and extrapolate to systems of unseen interaction graphs or graphs of different sizes from training. Beyond structural reasoning, we have also built a neuro-symbolic framework integrated with learned object-based dynamics models for temporal and causal reasoning about videos, which demonstrates significantly better performance in answering the following four types of questions about physical interaction systems: descriptive (e.g., "What shape?"), explanatory (e.g., "What is responsible for event X?"), predictive (e.g., "What will happen next?"), and counterfactual (e.g., "What would have happened if X disappeared?") [Yi et al., 2019].

Model-based control. In Manuelli et al. [2021] and Li et al. [2022], we leveraged the parallel computing power of GPUs and applied sampling-based trajectory optimization methods to compute the control signals (Equation 1.4). Our model closes the control loop by taking the feedback from the environment and adjusting its planned behavior to compensate for the modeling error. We took a step further in Li et al. [2018, 2019b] by using the gradients from the learned model to achieve more efficient optimization of the action sequences. Through both simulated and real-world experiments, our robots have achieved success that greatly exceeds prior works in complex manipulation tasks, such as manipulating a deformable foam (Figure 1-3a), a pusher-slider system (Figure 1-3b), and a cup of water with floating ice cubes (Figure 1-3c). In Li et al. [2019a], we traded off the expressiveness and efficiency of the model by assuming a linear structure over the latent state space, where we could apply quadratic programming to derive the control signals to manipulate ropes and control soft robots.

1.2.3 Multi-Modal Sensing of Physical Interactions via Vision and Touch

To build structured dynamics models grounded in the underlying physical world, it is essential to think thoroughly about what sensory information y_t we need to obtain, how different sensing modalities complement each other, and what unique value a specific modality

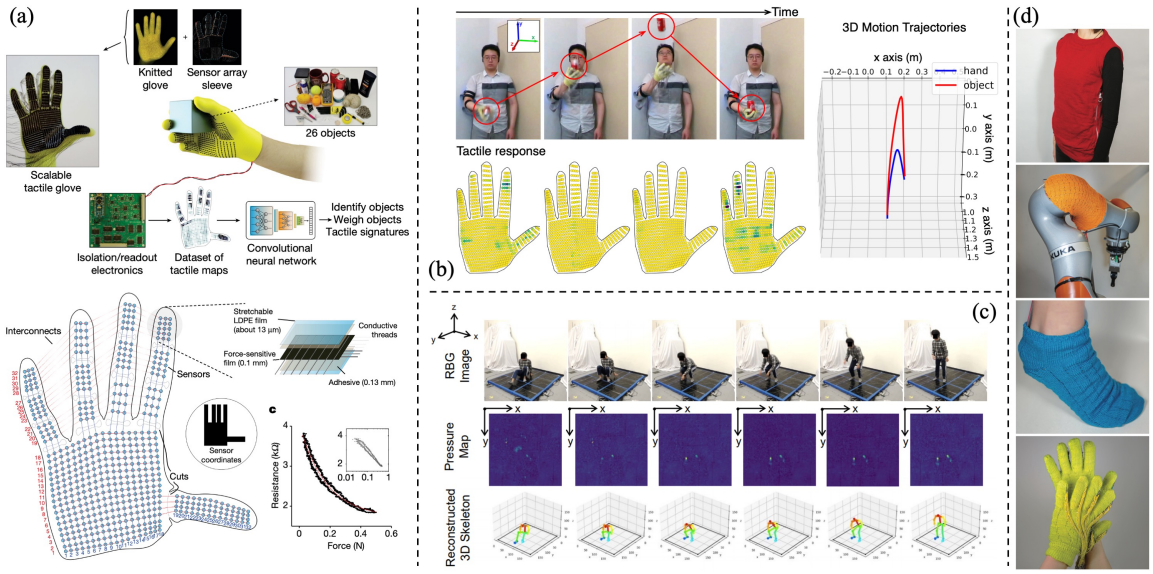


Figure 1-4: **Multi-modal sensing platforms using vision and touch.** To build better structured models of the world, we need expressive multi-modal sensing platforms that include the desired physical priors (e.g., contact and force constraints). Thus, we built scalable, deformable, and flexible tactile sensors in the form of gloves, socks, vests, and robot sleeves to obtain more detailed modeling of physical interactions and study the dynamic interactive behaviors of diverse physical activities. (a) A flexible glove capable of recording tactile information during hand-object interaction. (b) Tactile image and vision-enabled hand-object dynamics modeling. (c) Vision-supervised tactile learning for estimating 3D human poses. (d) 3D conformal tactile sensing garments for learning human-environment interactions.

brings. Specifically in physical interactions, besides vision, touch also plays a critical role in obtaining detailed contact information and injecting physical constraints, such as contacting modes, force, and local contact patterns. Therefore, in papers accepted to *Nature* and *Nature Electronics*, we built multiple low-cost multi-modal sensing platforms to obtain synchronized data from vision and touch that constrain the models used for describing the physical interactions. The systems enable efficient learning frameworks of human-environment interactions, laying the foundation for more expressive structured dynamics modeling.

Specifically, we have developed a set of scalable, high-resolution, conformal tactile sensor arrays that can be automatically manufactured with inexpensive materials [Sundaram et al., 2019, Luo et al., 2021b]; for example, we have designed scalable tactile gloves (Figure 1-4a), which, when coupled with tools from the deep learning community, can learn to discover the signatures of human grasps [Sundaram et al., 2019]. Based on the same working mechanism, we have successfully developed various tactile sensing systems, including a large-scale intelligent carpet [Luo et al., 2021a] (Figure 1-4c) and other wearable tactile sensor arrays (Figure 1-4d). The developed prototypes allowed us to record synchronized vision

and tactile data from various human activities and human-object interactions [Zhang et al., 2021], which can significantly facilitate the development of self-supervised learning systems and enable the building of models that capture the correlation between different sensory modalities and incorporate the constraints in the underlying physical world. For example, using the developed tactile glove, we have been able to demonstrate the ability to (i) identify objects, (ii) learn grasping patterns, (iii) identify weights of grasped objects [Sundaram et al., 2019], and (iv) model the dynamics of hand-object interactions by predicting the 3D locations of both the hand and the object purely from the tactile data (Figure 1-4b) [Zhang et al., 2021]. The wearable garments further allow us to collect tactile information through human-environment interactions, enabling learning systems to identify 3D human poses, activities, and postures.

Insights from the multi-modal sensing platforms—through the lens of automatically manufactured dense tactile arrays—can aid in the future design of new prosthetics and robot grasping/interaction tools. It can also enable more effective robotic manipulation and human-robot interactive capabilities by learning physically grounded predictive models for more expressive and structured modeling of the interactive dynamics.

1.3 Dissertation Structure

This dissertation consists of three parts, circling around the key questions of what representation we should use to describe the environment and what functional class we should use to model the scene dynamics (Equation 1.2 and 1.3). Specifically, I want to learn structured world models from the observation of the environment and show how the models can be learned from robots' interactions with their surroundings and help the robots to make better physical interactions, especially with objects with complicated physical properties.

Part I of this thesis will discuss the use of structured world models in **model-based control** tasks. I will first discuss the problem of a pusher-slider system in the face of external disturbances in Chapter 2. Being robust to external perturbations puts a strong emphasis on the efficiency of the model and requires the system to take real-time feedback from the environment. We thus propose to use keypoints as representations to learn efficient models capable of performing closed-loop real-time model-predictive control. This chapter was previously published as Manuelli et al. [2021]. Then, Chapter 3 will present my work on

using particles as the representations to learn the dynamics at a finer level of granularity for objects of different materials (e.g., rigid bodies, deformable objects, and fluids) and show how we can formulate the model-based control problem to manipulate these objects both in the simulation and the real world. This chapter was previously published as Li et al. [2018].

Moving forward, for objects with extremely high degrees of freedom like fluids, estimating the particle set from raw visual observations is hard, and keypoints won't be able to capture the variability of the object. I thus propose to learn latent-space dynamics models augmented with differentiable volume rendering in Chapter 4 to model the fluid dynamics purely from visual observations. This chapter is primarily based on Li et al. [2022]. On top of this, I will discuss our initial attempts to learn a linear approximation of the latent-space dynamics model with the aim to improve the system identification and model-based optimization performance by drawing insights from Koopman operator theory in Chapter 5 with materials previously published in Li et al. [2019a].

Part II of this thesis will discuss the use of structured world models for the **physical inference** tasks that aim to identify the latent causal structure and perform state and parameter estimation from observation data. Specifically, Chapter 6 (originally published as Li et al. [2020b]) will discuss how I perform structural inference of the underlying system and understand the Granger causal relationship between the constituting components by representing the environment using unsupervised keypoints. Then, Chapter 7 takes a step forward by focusing on more complicated systems involving the interactions between deformable objects, fluids, and rigid bodies. Based on the work published as Li et al. [2020a], the proposed method can perform state and parameter estimation that estimates the object particle positions, object segmentations, and the physical parameters that, when coupled with a learned particle-based dynamics model, can best explain the visual observations and make future predictions.

Part III of this thesis aims to bridge the sensing gap between humans and robots by introducing **multi-model sensing and learning platforms**. Chapter 8 will discuss the work initially published in *Nature* by Sundaram et al. [2019] that proposes a way of manufacturing dense tactile gloves capable of capturing the detailed tactile information between hand-object interactions. Based on similar technology that leverages piezoresistive force-sensitive materials, Chapter 9 will discuss how we construct tactile textiles in the form of gloves, socks, vests, and robot sleeves to capture the interactions between humans, robots,

and the environments. This chapter was previously published as Luo et al. [2021b]. I will show a range of applications made possible by the tactile sensors and the corresponding multi-modal sensing platforms. I will also demonstrate how such platforms can lead to better physically-grounded modeling of the physical interaction process.

Part I

Learning Structured World Models for Model-Based Control

Chapter 2

Learning Keypoint Dynamics via Self-Supervised Dense Correspondence

As discussed in the previous chapter, predictive models have been at the core of many robotic systems, showing promising results ranging from quadrotors to walking robots. However, developing and applying such models to practical robotic manipulation tasks has been challenging due to high-dimensional sensory observations such as images. In this chapter, we introduce model-based prediction with self-supervised keypoints learned from dense visual correspondence. We show that not only is this indeed possible, but we demonstrate that these types of predictive models are extremely efficient and show compelling performance improvements over alternative methods for vision-based reinforcement learning (RL) with autoencoder-type vision training that does not incorporate explicit 3D structures. Through simulation experiments, we demonstrate that our models provide better generalization precision, particularly in 3D scenes, scenes involving occlusion, and in category-generalization. Additionally, we validate that our method effectively transfers to the real world through hardware experiments. Video illustrations can be found on our project website: `https://sites.google.com/view/keypointsintothefuture`.

This chapter includes materials previously published as Manuelli et al. [2021] in the Conference on Robot Learning (CoRL) 2020 with co-authors Lucas Manuelli, Pete Florence, and Russ Tedrake, in which Lucas Manuelli has made major contributions to the content of this chapter.

2.1 Introduction

This chapter aims to build models that can approximately predict how an object will move if we interact with it, and then use the models for planning purposes, with a focus on pusher-slider systems involving external disturbances. Traditional model-based robotics has successfully leveraged such predictive models, oftentimes derived from first principles, to solve challenging planning and control problems [Moore et al., 2014, Mellinger et al., 2012]. In the area of practical vision-based robotic manipulation, however, it has been particularly hard to leverage such predictive models, due to varied and novel objects and the high-dimensional observation spaces involved (e.g., RGB or RGBD images). Alternative approaches, such as imitation learning or model-free reinforcement learning, sidestep the task of building a predictive model and directly learn a policy. Although this can be appealing, model-based techniques offer several benefits. They can be sample efficient compared to model-free methods and, in contrast to behavior cloning techniques, can leverage off-policy non-expert data. Once a model has been acquired, it can be used together with a planner to achieve a wide variety of tasks and goals. One of the main challenges for model learning applied to robotic manipulation is determining the state representation on which the dynamics model should be learned. Prior work has used approaches ranging from full image space dynamics [Finn et al., 2016, Ebert et al., 2017, Yen-Chen et al., 2020, Suh and Tedrake, 2020] to a variety of autoencoder formulations [Agrawal et al., 2016, Hafner et al., 2019b].

In this chapter, we propose to use object keypoints, which are tracked over time, as the latent state in which to learn the dynamics. These keypoints anchor our model-based predictions, and provide various advantages over alternatives: (i) the output is interpretable, which enables the ability to analyze the performance of the visual model separately from the predictive model. (ii) The representation is 3D and hence can naturally handle changing general off-axis 3D camera positions. (iii) As demonstrated in Florence et al. [2018, 2019], the visual models we use, *Dense Object Nets*, have demonstrated reliable performance in a variety of real-world settings and are able to generalize at the category level. We found that autoencoder approaches particularly struggled with category-level generalization. And as discussed in prior works [Florence et al., 2018, Manuelli et al., 2019], keypoints and dense correspondences provide advantages over using 6D object poses: they can apply to deformable objects and represent category-level generalization.

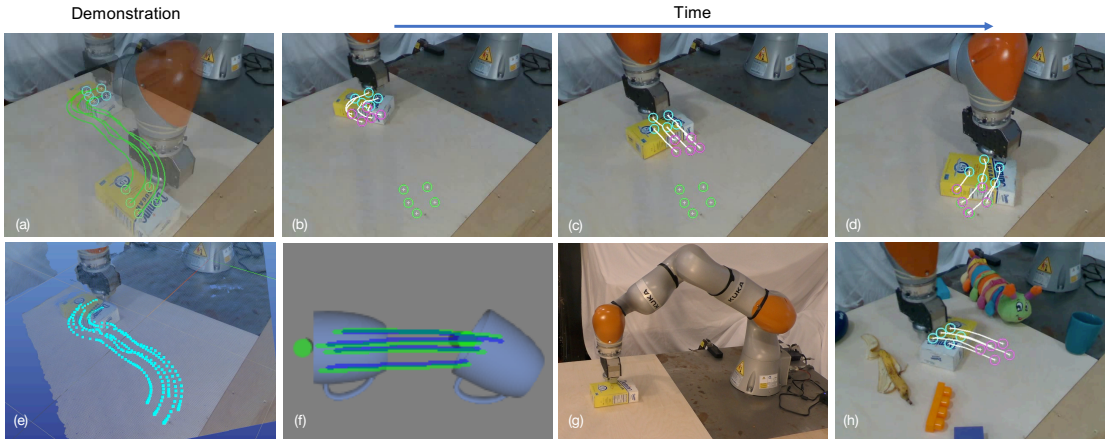


Figure 2-1: **Learning keypoint dynamics for model-predictive control.** (a) Shows the initial pose (blue keypoints) and goal pose (green keypoints) along with the demonstration trajectory. (b) - (d) show the MPC at different points along the episode. Current keypoints are in blue, white lines and purple keypoints show the optimized trajectory from the MPC algorithm, using the learned dynamics model. Goal keypoints are still shown in green. (e) shows the demonstration trajectory in a 3D visualizer. (f) Illustrates a dynamics model on a category level task. The actual keypoint trajectory is shown in green; the predicted trajectory using the learned model shown in blue. (g) Overview of our hardware setup. (h) Example of visual clutter.

We show that this formulation enables reliable, sample-efficient learning capable of precise visual-feedback-based manipulation in the real world – and is trained with nothing other than a small amount of interaction data (10 minutes) and a single demonstration for goal specification. In our approach to acquiring keypoints, we extract them as descriptors, which are tracked from a dense descriptor model – while multiple approaches could be used to acquire keypoints, this route can be entirely self-supervised. As opposed to Florence et al. [2019] that uses keypoints from a dense-correspondence model in an imitation learning framework, the use of keypoints as input to a model-based RL system presents several unique challenges. In particular, the keypoints need to be both informative for the task at hand and be able to be tracked accurately. In this chapter, we explore these challenges and propose solutions.

Contributions: Our primary contribution is (i) a novel formulation of predictive model-learning using learned dense visual descriptors as the state representation. We use this approach to perform closed-loop visual feedback control via model-predictive control (MPC). (ii) Using simulated manipulation experiments we demonstrate that this approach offers performance benefits over a variety of baselines, and (iii) we validate our approach in

real-world robot experiments.

2.2 Related Work

We focus our related work on methods that target robotic manipulation with learned predictive models. The Introduction (Section 2.1) discussed alternative approaches to synthesizing closed-loop feedback controllers without predictive models, via imitation learning or model-free reinforcement learning.

Model-based RL in robotic manipulation. These methods can be classified by whether they use first-principles-based or data-driven models, and whether they consume raw visual inputs (such as RGBD images) or they consume ground truth state information (from a simulator or an external perception system). First-principles-based models (e.g., Hogan and Rodriguez [2016], Zhou et al. [2019]) rely on known object models and thus don't generalize to novel or unknown objects, and also rely on external vision systems. Given this, the tasks we consider are out of scope for these approaches. In the area of methods that use external vision systems but learn data-driven models, Hogan et al. [2018] learns a dynamics model for a closed-loop-controlled planar pushing task, however, the approach is tailored to the specifics of the pusher-slider task and doesn't readily generalize to other tasks, or to novel objects within the same task. Nagabandi et al. [2020] learns deep dynamics models for a variety of different simulated dexterous manipulation tasks, using ground truth object states, and one task on real hardware, using an external camera-based 3D object position tracker. Although Hogan et al. [2018] and Nagabandi et al. [2020] achieve impressive results, their reliance on ground truth object state and/or specialized visual trackers limits their general applicability in more diversified real-world manipulation tasks.

There also exists a large literature on model-based RL for robotic manipulation that operates in the more challenging problem class of directly consuming image observations. Methods can be broadly categorized into whether they predict the image-space dynamics of the entire image [Finn et al., 2016, Ebert et al., 2018, 2017, Yen-Chen et al., 2020, Suh and Tedrake, 2020], or predict the dynamics of a low-dimensional latent state [Yan et al., 2021, Watter et al., 2015, Agrawal et al., 2016, Hafner et al., 2019b]. Although image-space dynamics approaches are general, they require large training datasets. For latent-space dynamics approaches, to avoid a trivial solution where all observations get mapped to a

constant vector, a regularization strategy is needed for the latent state $\boldsymbol{z}$. Watter et al. [2015] and Hafner et al. [2019b] use an autoencoder architecture and regularize the latent state $\boldsymbol{z}$ using a reconstruction loss. Agrawal et al. [2016] regularize the latent space by simultaneously training both forward and inverse dynamics models while Yan et al. [2021] uses a contrastive loss on the latent state. In contrast to these approaches we use visual-correspondence pretraining to produce a latent state which is physically grounded and interpretable as the 3D locations of keypoints on the object(s).

Visual representation learning. For more related-work in self-supervised visual learning for robotics, we refer the reader to Florence et al. [2018]. Approaches that use autoencoders [Watter et al., 2015, Hafner et al., 2019b] or full image-space dynamics [Finn et al., 2016, Ebert et al., 2018, 2017, Yen-Chen et al., 2020] rely on image reconstruction as their source of visual supervision. Kulkarni et al. [2019] is perhaps most related to ours, in that they first learn a visual model which is then used for a downstream task, and they show that freezing the visual model and using a keypoint-type representation as an input to model-free RL algorithms improves performance on Atari ALE [Bellemare et al., 2013]. Our approach is distinct in that (i) we use a predictive model-based method rather than a model-free method, (ii) we use a correspondence-based training loss, while Kulkarni et al. [2019] uses an image-reconstruction-based loss with a specialized pixel-space transport mechanism, and (iii) we demonstrate results with real-world hardware. As a baseline, we try using their vision model as an input to the same model-based RL algorithm used by our own method.

2.3 Self-Supervised Correspondence in Model-Based RL

This section describes our approach to model-based RL using visual observations. The goal is to learn a dynamics model that can then be used to perform online planning for closed-loop control.

2.3.1 Model-Based Reinforcement Learning

Our setting consists of an environment with states $\boldsymbol{x} \in \mathcal{X}$, observations $\boldsymbol{o} \in \mathcal{O}$, actions $\boldsymbol{a} \in \mathcal{A}$ and transition dynamics $\boldsymbol{x}' = f_{\text{state}}(\boldsymbol{x}, \boldsymbol{a})$. The task is specified by a reward function $r(x, a)$ and the goal is to choose actions to maximize the expected reward over a trajectory.

We approach this problem by first learning an approximate dynamics model $\hat{f}_{\theta_{\text{dyn}}}$, which is trained to minize the dynamics prediction error on the observed data $\mathcal{D}$. The learned model is then used to perform online planning to obtain a feedback controller.

Typical example environments are depicted in Figures 2-1 and 2-3. The state $\boldsymbol{x}$ contains information about the underlying pose and physical properties of the object, but we only have access to the observation $\boldsymbol{o}$. The observation typically consists of both the robot's proprioceptive information $\mathcal{O}_{\text{robot}}$ (such as joint angles, end-effector poses, etc.) and high-dimensional images $\mathcal{O}_{\text{image}} \in \mathbb{R}^{W \times H \times C}$ for a C-channel image of height H and width W. Hence the full observation space is $\mathcal{O} = \mathcal{O}_{\text{robot}} \times \mathcal{O}_{\text{image}}$. For the purposes of our approach we will assume that $\boldsymbol{x}$ is fully observable from $\boldsymbol{o}$ (or a short history of $\boldsymbol{o}$ in order to infer velocity information). Note that this still allows for the object to undergo significant partial occlusion as long as it is not completely occluded, see Figure 2-3c for an example. While our work focuses on addressing partial occlusion, future work may address full occlusion via models with longer time horizons or higher-level planning.

Rather than learn the dynamics directly in the observation space, as in Finn et al. [2016] and Ebert et al. [2018], we instead learn a mapping $g : \mathcal{O} \to \mathcal{Z}$ from the high-dimensional observation space $\mathcal{O}$ to a low-dimensional latent space $\mathcal{Z}$ together with a dynamics model $\hat{\boldsymbol{z}}_{t+1} = \hat{f}_{\theta_{dyn}}(\boldsymbol{z}_{t-l:t}, \boldsymbol{a}_t)$ in this latent space, where $\boldsymbol{z}_{t-l:t} = (\boldsymbol{z}_t, \boldsymbol{z}_{t-1}, \ldots, \boldsymbol{z}_{t-l})$ encodes a short history (we use $l = 1$ in all experiments). This latent state should capture sufficient information about the true state $\boldsymbol{x}$ such that driving $\boldsymbol{z} \to \boldsymbol{z}^*$ sufficiently well achieves the goal of driving $\boldsymbol{x}$ to $\boldsymbol{x}^*$ (where $\boldsymbol{z}^*$ is the latent state corresponding to $\boldsymbol{x}^*$). Given the current latent state and a sequence of actions $\{\boldsymbol{a}_t, \boldsymbol{a}_{t+1}, \ldots\}$ we can predict future latent states $\boldsymbol{z}$ by repeatedly applying our learned dynamics model. The forward model is then trained to minimize the dynamics prediction error (also know as *simulation error*) over a horizon H

$$\mathcal{L}_{\text{dynamics}} = \sum_{h=1}^{H} ||\hat{\boldsymbol{z}}_{t+h} - \boldsymbol{z}_{t+h}||_2^2, \quad \hat{\boldsymbol{z}}_{t+h+1} = \hat{f}_{\theta_{\text{dyn}}}(\hat{\boldsymbol{z}}_{t-l:t}, \boldsymbol{a}_{t+h}), \quad \hat{\boldsymbol{z}}_t = \boldsymbol{z}_t \tag{2.1}$$

2.3.2 Learning a Visual Representation

The objective of the visual model $g : \mathcal{O} \to \mathcal{Z}$ is to produce a low-dimensional feature vector $\boldsymbol{z}$ which serves as a suitable latent state in which to learn the dynamics. For the types of tasks and environments that we are interested in, spatial information about object

locations is a critical piece of information. Pose estimation has played a critical role in classical manipulation pipelines, and was also used in dynamics learning approaches such as Hogan et al. [2018] and Nagabandi et al. [2020]. In general, producing pose information from high-dimensional observations (such as RGBD images) requires a dedicated perception system. Although pose can be a powerful state representation when dealing with a single known object, as noted in Florence et al. [2018], Manuelli et al. [2019] and Florence et al. [2019], it has several drawbacks that limit its usefulness in more general manipulation scenarios. In particular it doesn't readily (i) extend to the case of deformable objects, (ii) generalize to novel objects or (iii) extend to category-level tasks.

Our strategy is to leverage visual-correspondence pre-training to track points on the object of interest. The locations of these tracked points can then serve as the latent state on which we learn the dynamics. Similar to the approach taken in Florence et al. [2018, 2019], we use visual-correspondence learning, which is trained in a completely self-supervised fashion, to train a visual model which that can be used to find correspondences across RGB images. We then propose several approaches to produce a low-dimensional latent $\boldsymbol{z}$ using the pre-trained dense-correspondence model.

First we give a bit of background on dense correspondence models [Florence, 2019, Florence et al., 2018]. Given an image observation $\boldsymbol{o}_{\text{image}} \in \mathbb{R}^{W \times H \times C}$ (where C denotes the number of channels), the dense-correspondence model $g_{\theta_{\text{dc}}}$ outputs a full-resolution descriptor image $\mathcal{I}_D \in \mathbb{R}^{W \times H \times D}$. Since we want to learn a dynamics model on a low-dimensional state, we need a way to construct $\boldsymbol{z}$ from the descriptor image $\mathcal{I}_D$. The idea, similar to Florence et al. [2019], is for $\boldsymbol{z}$ to be a set of points on the object(s) that are localized in either image-space or 3D space. These points are represented as a set $\{d_i\}_{i=1}^K$ of K descriptors, where each $d_i \in \mathbb{R}^D$ is a vector in the underlying descriptor space. A parameterless *correspondence function* $g_c(\mathcal{I}_D, d_i)$* extracts the location of the keypoint $y_i \in \mathbb{R}^B$ from the current observation. Combining our learned correspondences together with the reference descriptors, we have a function that maps image observations $\boldsymbol{o}_{\text{image}}$ to keypoint locations $\boldsymbol{y} = \{y_i\}_{i=1}^K$. We propose four methods for constructing the latent state $\boldsymbol{z} = g_{\theta_z}(\boldsymbol{y})$ from $\boldsymbol{y}$, where θ_z denotes the (potentially empty) set of trainable parameters in this mapping.

Descriptor Set (DS): In our simplest variant the latent state $\boldsymbol{z}$ is simply made up of keypoint locations y_i for a set of descriptor keypoints randomly sampled from the object. Specifically,

*see Appendix A.1.2 for more details on the visual-correspondence model

we sample K (we use $K = 50$ in all experiments) descriptors $\{d_i\}_{i=1}^K$ corresponding to pixels from a masked reference descriptor image in our training set. Thus

$$\boldsymbol{z} = (\boldsymbol{z}_{\text{object}}, \boldsymbol{o}_{\text{robot}}) = (\boldsymbol{y}, \boldsymbol{o}_{\text{robot}}) = (\{y_i\}_{i=1}^K, \boldsymbol{o}_{\text{robot}}) \tag{2.2}$$

Spatial Descriptor Set (SDS): Rather than randomly sampling descriptors, as in **(DS)**, this method attempts to choose descriptors $\boldsymbol{d} = \{d_i\}_{i=1}^K$ having specific properties. In particular we would like the descriptors $\{d_i\}_{i=1}^K$ to be (i) *reliable*, and (ii) *spatially separated*. By *reliable* we mean that they can be localized with high accuracy and don't become occluded during the typical operating conditions, see Figure 2-2 for an example. *Spatially separated* means that the chosen descriptors aren't all clustered around the same physical location on the object(s) of interest, but rather are sufficiently spread out (either in 3D space or pixel space) to provide meaningful information about both object position and orientation. Our dense descriptor model can provide a confidence score associated with descriptors and their associated correspondences.[†] Figure 2-2 shows a clear example of high confidence for a valid match and low-confidence when no valid correspondence exists due to an occlusion. We use this feature of our visual model to compute a confidence score c_i for each each descriptor d_i according to what fraction of images in the training set $\mathcal{D}$ contain a high probability correspondence for d_i. The intuition is that descriptors d_i corresponding to points on the object that are easy to localize and remain unoccluded will have a high confidence score. As in the **DS** method we initially select a large number of descriptors ($K = 100$) corresponding to points on the object. We then select the K^* descriptors with the highest average confidence on the training set and which additionally satisfy a threshold on minimum separation distance (typically 25 pixels in a 640×480 image). In the experiments we use $K^* = 4$ or $K^* = 5$.

Weighted Descriptor Set (WDS): One problem that can arise when learning the dynamics of a latent state $\boldsymbol{z}$ is that some component of $\boldsymbol{z}$ may be noisy or unreliable, which can make it difficult or impossible to learn a dynamics model $\boldsymbol{z}' = f(\boldsymbol{z}, \boldsymbol{a})$. This problem doesn't arise in the imitation learning setting of Florence et al. [2019] which performs supervised learning from $\boldsymbol{z}_t \rightarrow \boldsymbol{a}_t^{\text{expert}}$, and thus can learn to ignore components of the latent state $\boldsymbol{z}_t$ that aren't useful for predicting the action $\boldsymbol{a}_t^{\text{expert}}$. The fundamental difference of our dynamics learning formulation compared to the imitation learning setup of Florence et al. [2019] is that the

[†]see Appendix for more details

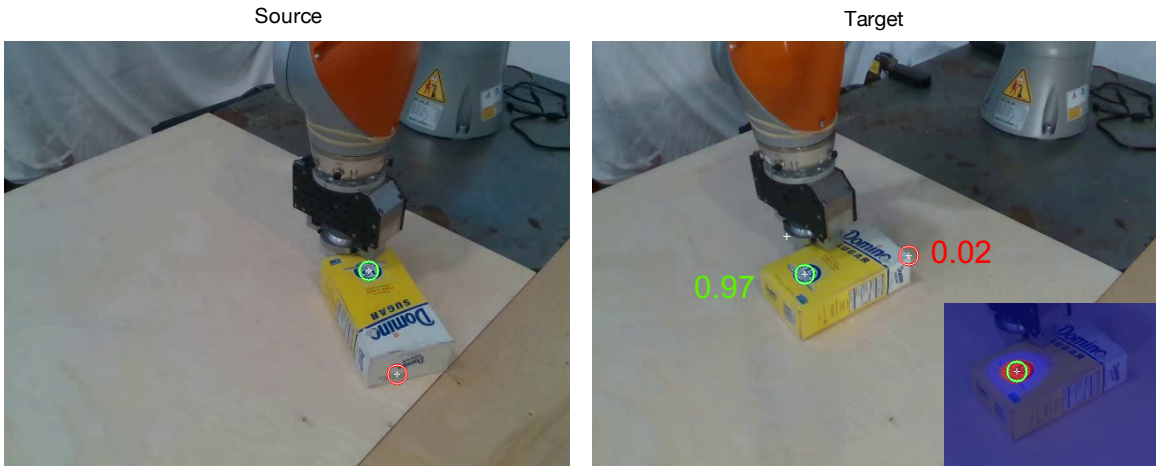


Figure 2-2: **Visualization of the learned visual-correspondence model** on a reference image (left) and target image (right). Colored numbers in the target image represent the probability that the detected correspondence is valid. Green reticle shows a valid correspondence with a high confidence score, red reticle shows a case where no correspondence exists in the target image due to occlusion, hence the low confidence probability. Confidence heatmap shown in bottom right.

latent $\boldsymbol{z}_t$ appears directly in the cost function (Equation 2.1). There are a variety of reasons that one or more of the tracked descriptors could be unreliable (e.g., occlusions, regions of the object where the correspondence model has less accurate) and thus we would like our dynamics learning framework to be robust to this. We achieve this by defining a learnable mapping $\phi : \boldsymbol{y} \to \boldsymbol{z}$ which maps the keypoint locations $\boldsymbol{y}$ to the latent state $\boldsymbol{z}$, where the keypoints *bmy* are as in our **DS** method. A regularization strategy is needed to ensure that ϕ doesn't collapse to the trivial solution $\phi \equiv 0$. We introduce learnable weights $\alpha \in \mathbb{R}^{K \times K}$ and define $w_{k,i} = \frac{\exp(\alpha_{k,i})}{\sum_{j=1}^{K} \exp(\alpha_{k,j})}$. Let $W \in \mathbb{R}^{K \times K}$ be the matrix with entries $w_{k,i}$, where the parameterization guarantees that $w_{k,i} \geq 0$ and $\sum_{i=1}^{K} w_{k,i} = 1$. ϕ is defined as

$$\tilde{y}_k = \sum_{i=1}^{K} w_{k,i} y_i, \quad \phi(\boldsymbol{y}; \alpha) = \tilde{\boldsymbol{y}} = \{\tilde{y}_k\}_{k=1}^{K} \tag{2.3}$$

Note that each y_i is a keypoint location in $\mathbb{R}^B$. Thus $\tilde{\boldsymbol{y}}$ is simply a convex combination of the keypoints in $\boldsymbol{y}$. The latent state $\boldsymbol{z}$ is then defined as

$$\boldsymbol{z} = (\boldsymbol{z}_{\text{object}}, \boldsymbol{o}_{\text{robot}}) = (\tilde{\boldsymbol{y}}, \boldsymbol{o}_{\text{robot}}) = (\phi(\boldsymbol{y}, \alpha), \boldsymbol{o}_{\text{robot}}) \tag{2.4}$$

The learnable weights α are trained jointly with the parameters θ_{dyn} of the dynamics model, and are fixed at test time. The fact that $\boldsymbol{z}_{\text{object}}$ is gotten from $\boldsymbol{y}$ by taking a weighted linear combination preserves the interpretation of $\boldsymbol{z}$ as tracking keypoints on the object, while allowing some flexibility to ignore unreliable keypoints.

Weighted Spatial Descriptor Set (WSDS): This method is simply the combination of **(SDS)** and **(WDS)**.

2.3.3 Learning the Dynamics

We adopt a standard dynamics learning framework where we aim to predict the evolution of the latent state $\boldsymbol{z}$. We train our dynamics model to minimize the prediction error in Equation 2.1 where $\boldsymbol{z}_t = g_{\theta_z}(\boldsymbol{o}_t)$. Our proposed methods differ in the structure of the mapping $g : \mathcal{O} \to \mathcal{Z}$ and the set of trainable parameters Θ.[‡] For all of our methods we keep the weights of the visual-correspondence network, θ_{dc}, fixed.

2.3.4 Online Planning for Closed-Loop Control

Once we have learned a dynamics model $\boldsymbol{z}' = f(\boldsymbol{z}, \boldsymbol{a})$, we use online planning with MPC to select an action. Given a goal latent state $\boldsymbol{z}^*$, we want to find an action sequence $\{\boldsymbol{a}_{t'}\}_{t'=t}^{t+H-1}$ that maximizes the reward $R = \sum_{t'=t}^{t+H-1} r(\boldsymbol{z}_{t'}, \boldsymbol{a}_{t'})$. Our dynamics learning approach is agnostic to the type of optimizer used to solve the MPC problem and the focus of our work is on the visual and dynamics learning, rather than the specifics of the MPC. Many model-based RL approaches [Yen-Chen et al., 2020, Ebert et al., 2017, Nagabandi et al., 2020, Hafner et al., 2019b, Finn et al., 2016] use a random-sampling based planner (e.g., cross-entropy method) to solve the underlying MPC problem. We experimented with random shooting, gradient based shooting, cross-entropy and model-predictive path integral (MPPI) planners and found that MPPI worked best in our scenarios.[§]

2.4 Results

We perform experiments aimed at answering the following questions: (1) Is it possible to successfully use self-supervised descriptors as the latent state for a model-based RL

[‡]see Appendix A.2.2 for details

[§]See Appendix A.3 for more details.

system? (2) What is the effect of various design decisions in our algorithm? (3) How does visual-correspondence learning compare to several benchmark methods in terms of enabling effective model-based RL policies? (4) Can we apply the method on real hardware?

Our main contribution is on the formulation of the visual model and dynamics learning problem rather than the specifics of the MPC. However, to accurately compare our approach to various baselines, we need to perform experiments in which the dynamics model is used in closed-loop to solve a manipulation task. Our formulation of dynamics learning is very general, and in principle can handle a wide variety of manipulation scenarios. However, even with an accurate model (whether it comes from first principles or is learned), using this model to perform closed-loop feedback control remains a challenging problem. Thus, for our closed-loop experiments, we limit ourselves to pushing tasks that can adequately be solved by the planners outlined in Section 2.3.4.

Tasks: Extended experimental details are provided in the Appendix but we provide a brief overview of the tasks here. We perform experiments with four simulated tasks (depicted in Figure 2-3) and one hardware task. All tasks involve pushing an object to a desired goal state. The first three simulation tasks, denoted as *top-down, angled, occlusions* involve pushing a single object. *top-down* has cameras in a top-down orientation while they are angled at 45 degrees in *angled*. Task *occlusions* keeps the angled camera positions but the box is resting on a different face, resulting in significant self-occlusions. Task *mugs* involves pushing many different objects from a category, in this case mugs with different size, shape and textures. The hardware task is essentially identical to the *angled* sim task.

2.4.1 Visual-Correspondence Performance

Figures 2-1 and 2-2 show the performance of our dense visual-correspondence model. In particular, Figure 2-1 shows the localization performance on real data along a trajectory while Figure 2-2 shows an example of the confidence scores used as in the **SDS** method. Figure 2-3 shows the descriptors used in the **SDS** method for each of our simulation tasks. In particular Figure 2-3d-e shows the ability of the descriptors to accurately find correspondences across different object instances within a category, despite differences in color and shape.

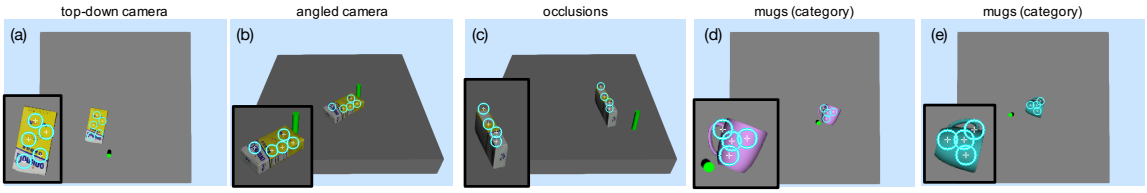


Figure 2-3: This figure shows reference descriptors $\{d_i\}_{i=1}^K$ of the **Spatial Descriptor Set** variant for our different simulation environments. In particular in (c) the sides of the object undergo occlusions as the box rotates about the vertical axis. (d)-(e) show two different mugs from the category-level *mugs (category)* task.

Task	*top-down camera*		*angled camera*		*occlusions*		*mugs (category)*	
Method / task data	pos, cm	angle, °	pos, cm	angle, °	pos, cm	angle, °	pos, cm	angle, °
Avg. trajectory	11.32	32.05	11.32	32.05	10.36	31.81	11.28	56.25
GT 3D	1.14	4.01	1.14	4.01	1.29	5.45	–	–
SDS	1.19	3.88	1.10	3.97	1.46	6.77	1.20	15.03
WDS	1.16	4.43	1.30	5.12	1.49	8.64	1.00	13.29
DS	1.12	4.07	1.45	5.50	3.66	12.27	1.19	11.73
WSDS	1.19	4.00	1.18	4.86	1.28	5.59	1.39	18.18

Table 2.1: **Ablations and comparison with ground-truth** – quantitative results for various ablations of our method on four simulated tasks. Each method was evaluated on the same set of 200 different initial and goal states. The *pos* and *angle* columns denote the translational (in cm) and rotational (in degrees) deviations of the object from the goal position, averaged across all trials for a specific method and task. *Avg. trajectory* denotes the average translation and rotation between the initial and goal poses for each task.

2.4.2 Ablations on Visual-Correspondence for Dynamics Learning

Ablation studies show that the choice of *what to track* can have a substantial impact on model performance, especially in the case of partial occlusions. Quantitative results are detailed in Table 2.1. On tasks *top-down* and *angled camera* all methods perform reasonably well, almost matching the performance of the **GT 3D** baseline that uses ground truth state information. Intuitively this is because there are minimal occlusions in these settings, and so almost all keypoints can be tracked reliably using dense-visual-correspondence. In contrast the *occlusions* introduces the potential for significant occlusions. Given the camera angle as shown in Figure 2-3c and the fact that the object rotates through the full 360 degrees of yaw during the task, only the top face of the box remains unoccluded while the 4 side faces are alternately occluded and visible. This task exposes significant differences in performance among our various ablations. In particular **SDS**, **WDS** and **WSDS** perform significantly better than **DS**. We believe that this is due to the fact that some of the descriptors $\{d_i\}_{i=1}^K$ that are tracked in **DS** correspond to locations on the object that become occluded during

Task	*top-down camera*		*angled camera*		*occlusions*		*mugs (category)*	
Method	pos, cm	angle, °	pos, cm	angle, °	pos, cm	angle, °	pos, cm	angle, °
SDS (ours)	1.19	**3.88**	**1.10**	**3.97**	**1.46**	**6.77**	1.20	15.03
WDS (ours)	**1.16**	4.43	1.30	5.12	1.49	8.64	**1.00**	**13.29**
Transporter 3D	2.01	15.95	4.36	25.14	3.72	20.65	2.81	61.22
Transporter 2D	2.08	13.59	3.60	23.60	2.74	18.86	2.33	60.18
Autoencoder	2.18	14.79	2.85	13.79	3.08	14.20	9.05	56.84

Table 2.2: **Comparisons with baselines** – quantitative results of our method compared to various baselines on our four simulated tasks. Each method was evaluated on the same set of 200 different initial and goal states. *pos* and *angle* denote the translational (in cm) and rotational (in degrees) deviations of the object from the goal position, averaged across all trials for a specific method and task.

an episode. The dense visual-correspondence model is not able to track keypoints through occlusions, and when trying to localize an occluded point our dense-correspondence model maps it to the closest point in descriptor space, which is not the location of the true correpondence. Hence the keypoint locations that makeup the latent state $\boldsymbol{z}_{\text{object}}$ for **DS** suffer reduced accuracy, leading to a less accurate dynamics model and ultimately lower performance when used for closed-loop MPC.

On the category-level task *mugs* the camera is in a top-down position and thus occlusions are no longer an issue. However because the task involves different objects from a category there is shape variation among the different objects. Thus the methods that use $K = 50$ keypoints, namely **DS** and **WDS**, perform better than the sparser variants **SDS**, **WSDS** that use only $K = 4, 5$ keypoints. We believe that this is due to the fact that having a larger number of keypoints better captures the shape variation across object instances and allows for a more accurate dynamics model.

2.4.3 Comparison of Visual-Correspondence Pretraining With Baselines

In our comparisons against baselines, our model outperforms alternatives on all experimental tasks. The largest differences are apparent in tasks *occlusions* and *mugs* which involve partial occlusions and category-level generalization, respectively. The **transporter** baseline uses the keypoint locations from Kulkarni et al. [2019] as the latent state z , while the **autoencoder** baseline jointly trains an autoencoder with a forward dynamics model. For a detailed discussion of the baselines see Appendix A.4.3. Quantitative results are detailed in Table 2.2.

On task *top-down camera*, the **transporter** [Kulkarni et al., 2019] model was able

to achieve performance that was only slightly worse than **WDS** and **SDS**, while the **autoencoder** performed significantly worse. The top-down setting is ideally suited for the feature transport approach of the transporter model.

On tasks *angled camera* and *occlusions*, which have angled camera positions as opposed to the top-down task, the performance of our methods remained consistent while **transporter** suffered. This is potentially due to the fact that the feature transport mechanism of **transporter** is not well-suited to off-axis camera positions in 3D worlds. The performance of the **autoencoder** baseline remainded consistent, but worse than our approach, across tasks without category-level generalization.

Task *mugs* contains a variety of different mug shapes with varied visual appearances and hence tests category-level generalization of both the perception and dynamics models. As discussed in Section 2.4.1, our dense-correspondence model is able to find correspondences across these variations in appearance using only self-supervision, which allows us to learn a dynamics model that is effective for completing the task. This task is significantly harder than the other tasks not only because of the presence of novel objects, but because the goal states involve much larger rotations, as detailed in the first row of Table 2.1. Both the **transporter** and **autoencoder** baselines perform poorly in this task. We hypothesize that this is due to the fact that there is much more variance in the visual appearance of the objects as compared to the other tasks and thus the latent state $\boldsymbol{z}$ produced by these baselines is not amenable to dynamics learning.

2.4.4 Hardware

Experimental Setup: We used a Kuka IIWA LBR robot with a custom cylindrical pusher attached to the end-effector to perform our hardware experiments, see Figure 2-1. RGBD sensing was provided by two RealSense D415 cameras rigidly mounted offboard the robot and calibrated to the robot's coordinate frame. To enable effective correspondence learning between views, it is ideal to have views with *some* overlap such that correspondences exist, but still maintain different-enough views. At test time only a single camera is used to localize the dense-descriptor keypoints. The robot is controlled by commanding end-effector velocity in the xy plane at 5Hz.

Hardware Results: For visual learning we collected a small dataset of the object in different positions to provide a diverse set of views for training the dense-correspondence

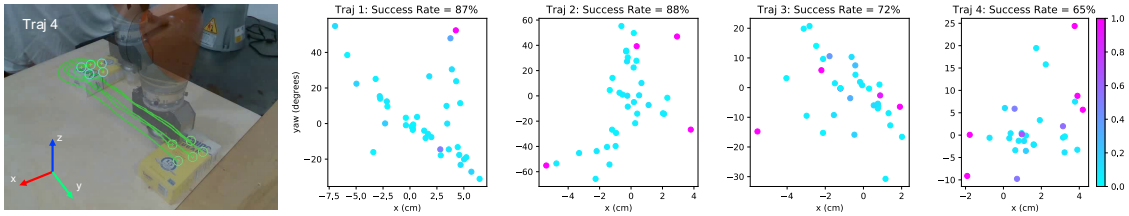


Figure 2-4: **Hardware results.** Left image shows demonstration for trajectory 4. Scatter plots show results of our approach on the four different reference trajectory tracking tasks. The axes of the plots show the deviation of the object starting pose from the initial pose of the demonstration. Color indicates the distance between final and goal poses, lower cost is better. The various reference trajectories are of different difficulties, as reflected by the different regions of attraction of the MPC controller. More details can be found in Appendix A.5 and videos are on our project page.

model. For dynamics learning, we collected a dataset of the robot randomly pushing the object around. This amounted to approximately 10 minutes of interaction time and was used to train the dynamics model. All hardware experiments used the **SDS** method. To enable our MPC controller to accomplish long-horizon tasks, we supplied the controller with a reference trajectory for the object keypoints that came from a single demonstration, see Figure 2-1a-d. The MPC controller then tracked this reference trajectory using a 2 second MPC horizon, which corresponds to $H = 10$ since we are commanding actions at 5 Hz. We collected 4 different reference trajectories[¶] and ran multiple rollouts for each trajectory, varying the initial condition of the object pose during each run to test the region of attraction of our controller. In all cases our controller showed the ability to stabilize the system to the reference trajectory in spite of perturbations to the initial condition. Quantitative results are detailed in Figure 2-4. In particular, we see that the ability of the controller to stabilize the trajectory in the face of disturbances in the initial condition depends on the trajectory. For trajectory (1) the controller is able to stabilize disturbances of up to 60 degrees, while trajectory (4) has a much smaller region of attraction. As can be seen in Appendix A.5, trajectory (1) is a relatively simple trajectory with minimal orientation change between start and goal, while trajectory (4) involves a challenging 180-degree orientation change and requires the robot to operate at the edge of its kinematic workspace, reducing its control authority. Overall, our system exhibits impressive feedback and is able to track a trajectory in the keypoint latent space, enabling one-shot imitation learning. We encourage the reader to watch the videos on our project page to see the system in action.

[¶]see Appendix A.5 for details on the reference trajectories

2.5 Discussion

We presented a method for using self-supervised visual-correspondence learning as input to a predictive dynamics model. Our approach produces interpretable latent states that outperform competing baselines on a variety of simulated manipulation tasks. Additionally, we demonstrated how the category-level generalization of our visual-correspondence model enables learning of a category-level dynamics model, resulting in large performance gains over baselines. Finally, we demonstrated our approach on a real hardware system, and showed its ability to stabilize complex long-horizon plans by tracking the latent state trajectory from a single demonstration.

Chapter 3

Learning Particle Dynamics for Manipulating Rigid Bodies, Deformable Objects, and Fluids

The previous chapter discussed the use of keypoint representations for pusher-slider systems, focusing on the manipulation of primarily rigid objects. However, real-life control tasks involve matters of various substances—soft bodies, granular materials, liquid, gas—each with distinct physical behaviors and having high degrees of freedom, making keypoints insufficient to capture the variations of the objects. Particle-based simulators have been developed to model the dynamics of these complex scenes; however, relying on approximation techniques, their simulation often deviates from real-world physics, especially in the long term. In this chapter, we propose to take an alternative approach by learning a particle-based simulator directly from data for complex control tasks. Combining learning with particle-based systems brings in two major benefits: first, the learned simulator, just like other particle-based systems, acts widely on objects of different materials; second, the particle-based representation poses a strong inductive bias for learning: particles of the same type have the same dynamics within. This enables the model to learn models of complicated interacting systems from a relatively small amount of data and generalize better than baselines that do not exploit the underlying structure. We demonstrate robots achieving complex manipulation tasks using the learned simulator, such as manipulating fluids and deformable foam, with experiments both in simulation and in the real world. Our study helps lay the foundation for

robot learning of dynamic scenes with particle-based representations. Please see our project page for video illustrations: http://dpi.csail.mit.edu.

This chapter was previously published as Li et al. [2018] in the International Conference on Learning Representations (ICLR) 2019 with Jiajun Wu, Russ Tedrake, Joshua B. Tenenbaum, and Antonio Torralba as co-authors.

3.1 Introduction

Objects have distinct dynamics. Under the same push, a rigid box will slide, modeling clay will deform, and a cup full of water will fall with water spilling out. The diverse behavior of different objects poses challenges to traditional rigid-body simulators used in robotics [Todorov et al., 2012, Tedrake and the Drake Development Team, 2019]. Particle-based simulators aim to model the dynamics of these complex scenes [Macklin et al., 2014]; however, relying on approximation techniques for the sake of perceptual realism, their simulation often deviates from real world physics, especially in the long term. Developing generalizable and accurate forward dynamics models directly from observation data is of critical importance for robot manipulation of distinct real-life objects.

We propose to learn a differentiable, particle-based simulator for complex control tasks based on neural networks, drawing inspiration from recent development in learning-based physical engines [Battaglia et al., 2016, Chang et al., 2017]. In robotics, the use of simulators capable of producing gradients, together with continuous and symbolic optimization algorithms, has enabled planning for increasingly complex whole-body motions with multi-contact and multi-object interactions [Toussaint et al., 2018]. Yet most of the work in this direction has only tackled interactions within rigid body systems. We use particles as the representation for the environment and develop dynamic particle interaction networks (DPI-Nets) for learning particle dynamics, focusing on capturing the dynamic, hierarchical, and long-range interactions of particles (Figure 3-1a-c). DPI-Nets can then be combined with classic perception and gradient-based optimization algorithms for robot manipulation of deformable objects (Figure 3-1d).

Learning a particle-based simulator brings in two major benefits. First, the learned simulator, just like other particle-based systems, acts widely on objects of different states. DPI-Nets have successfully captured the complex behaviors of deformable objects, fluids, and

rigid-bodies. With learned DPI-Nets, our robots have achieved success in manipulation tasks that involve deformable objects of complex physical properties, such as molding plasticine to a target shape.

Second, the particle-based representation poses a strong inductive bias for learning: particles of the same type have the same dynamics within. This enables the model to share the parameters of neural modules that are used to compute the interactions between particles. As a result, the model can learn from a relatively small amount of data and extrapolate to testing environments that are larger or smaller than the training distribution. Experiments suggest that DPI-Nets quickly learn to characterize a novel object of unknown physical properties. The learned model also helps the robot to successfully manipulate deformable objects in the real world.

DPI-Nets combine three key features for effective particle-based simulation and control: multi-step spatial propagation, hierarchical particle structure, and dynamic interaction graphs. In particular, it employs dynamic interaction graphs, built on the fly throughout manipulation, to capture the meaningful interactions among particles of deformable objects and fluids. The use of dynamic graphs allows neural models to focus on learning meaningful interactions among particles, and is crucial for obtaining good simulation accuracy and high success rates in manipulation. As objects deform when robots interact with them, a fixed interaction graph over particles is insufficient for robot manipulating non-rigid objects.

Experiments demonstrate that DPI-Nets significantly outperform interaction networks [Battaglia et al., 2016], HRN [Mrowca et al., 2018], and a few other baselines. More importantly, unlike previous papers that focused only on forward simulation, we have applied our model to downstream control tasks. Our DPI-Nets enable complex manipulation tasks for deformable objects and fluids and adapt to scenarios with unknown physical parameters that need to be identified online. We have also performed real-world experiments to demonstrate our model's generalization ability.

3.2 Related Work

Physics simulators with analytical gradients. Researchers have developed techniques to differentiate through physics-based simulators and sought to provide analytical gradients [Ehrhardt et al., 2017, Degrave et al., 2019, Todorov et al., 2012, Tedrake and the Drake

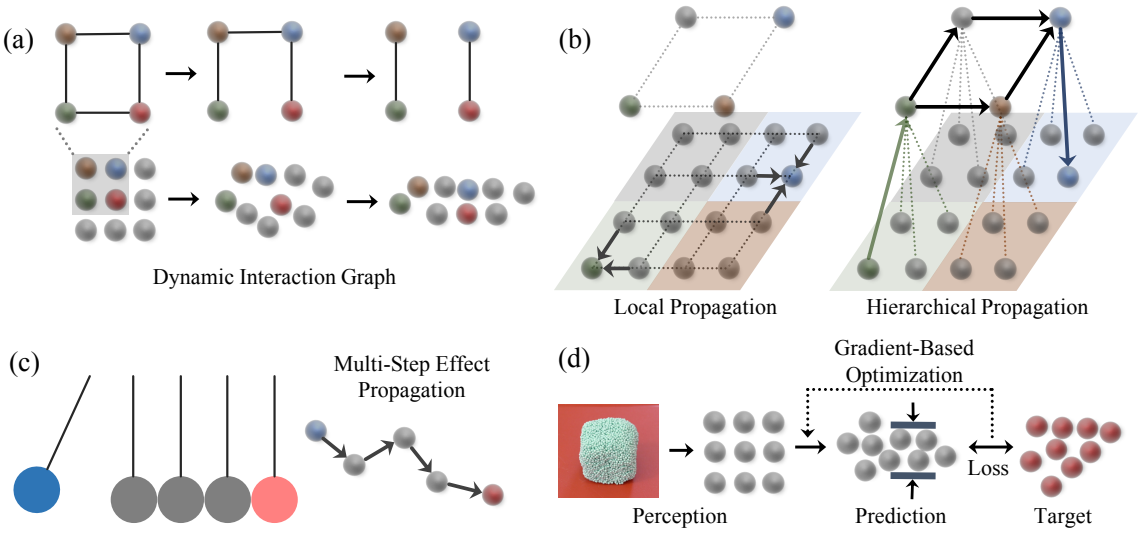


Figure 3-1: **Learning particle dynamics for control.** (a) DPI-Nets learn particle interaction while dynamically building the interaction graph over time. (b) Build hierarchical graph for multi-scale effect propagation. (c) Multi-step message passing for handling instantaneous force propagation. (d) Perception and control with the learned model. Our system first reconstructs the particle-based shape from visual observation. It then uses gradient-based trajectory optimization to search for the actions that produce the most desired output.

Development Team, 2019]. In particular, Battaglia et al. [2016] and Chang et al. [2017] have explored learning a simulator from data by approximating object interactions with neural networks. Li et al. [2019b] proposed learning to propagate signals along the interaction graph and extended to partially observable scenarios. These methods mostly focus on modeling rigid body dynamics.

Simulators capable of providing analytical gradients for deformable objects have been less studied. Recently, Schenck and Fox [2018b] proposed SPNets to extract gradients from position-based simulation of fluids [Macklin and Müller, 2013]. An inspiring concurrent work from Mrowca et al. [2018] explored learning to approximate particle dynamics of deformable shapes with the Hierarchical Relation Network (HRN). Compared with these papers, we introduce state-specific modeling and dynamic graphs for accurate forward prediction for different states of matter (rigid bodies, deformable shapes, fluids). We also demonstrate how the learned dynamics model can be used for control in both simulation and real world.

Our approach is also complementary to some recent work on learning to discover the interaction graphs [van Steenkiste et al., 2018, Kipf et al., 2018]. Our model can also be naturally augmented with a perception module to handle raw visual input, as suggested by Watters et al. [2017], Wu et al. [2017], and Fragkiadaki et al. [2016].

Model-predictive control with gradient-based optimization. Many recent papers have studied model-predictive control with deep networks [Lenz et al., 2015, Gu et al., 2016, Nagabandi et al., 2018, Farquhar et al., 2018, Srinivas et al., 2018]. They often learn an abstract state transition function, instead of an explicit account of the environment [Silver et al., 2017, Oh et al., 2017], and then use the learned function to facilitate training of a policy network. A few recent papers have employed the analytical gradients from the simulators [de Avila Belbute-Peres et al., 2018, Schenck and Fox, 2018b] for control problems, such as tool manipulation and tool-use planning [Toussaint et al., 2018]. Our model builds on and extends these approaches by learning a general physics simulator that takes raw object observations (e.g., positions, velocities) of each particle as input. We then integrate it into classic trajectory optimization algorithms for control. Compared with pure analytical simulators, our learned simulator can better generalize to novel testing scenarios where the governing equations and object/environment parameters are unknown.

A few papers have explored using interaction networks for planning and control. They often learn a policy based on interaction networks' rollouts [Racanière et al., 2017, Hamrick et al., 2017, Pascanu et al., 2017]. In contrast, our model learns a dynamics simulator and directly optimizes trajectories for continuous control. Recently, Sanchez-Gonzalez et al. [2018] have applied interaction networks for control, and Li et al. [2019b] have further extended interaction nets to handle instance signal propagation for controlling multiple rigid bodies under partial observations. Compared with them, our dynamic particle interaction networks simulate and control deformable, particle-based objects, using dynamic graphs to tackle scenes with complex object interactions.

3.3 Approach

3.3.1 Preliminaries

We first describe how interaction networks [Battaglia et al., 2016] represent the physical system; we then extend them for particle-based dynamics.

The interactions within a physical system are represented as a directed graph, $G = \langle O, R \rangle$, where vertices $O = \{o_i\}$ represent objects and edges $R = \{r_k\}$ represent relations. Specifically, $o_i = \langle x_i, a_i^o \rangle$, where $x_i = \langle q_i, \dot{q}_i \rangle$ is the state of object i, containing its position q_i and velocity $\dot{q}_i$. a_i^o denotes its attributes (e.g., mass, radius). For relation, we have

$r_k = \langle u_k, v_k, a_k^r \rangle, \quad 1 \leq u_k, v_k \leq |O|$, where u_k is the receiver, v_k is the sender, and both are integers. a_k^r is the type and attributes of relation k (e.g., collision, spring connection).

The goal is to build a learnable physical engine to capture the underlying physical interactions using function approximators ϕ. The learned model can then be used to infer the system dynamics and predict the future from the current interaction graph as $G_{t+1} = \phi(G_t)$, where G_t denotes the scene state at time t.

Interaction networks. Battaglia et al. [2016] proposed interaction networks (IN), a general-purpose, learnable physics engine that performs object- and relation-centric reasoning about physics. INs define an object function f_O and a relation function f_R to model objects and their relations in a compositional way. The future state at time $t+1$ is predicted as $e_{k,t} = f_R(o_{u_k,t}, o_{v_k,t}, a_k^r)_{k=1\ldots|R|}, \hat{o}_{i,t+1} = f_O(o_{i,t}, \sum_{k \in \mathcal{N}_i} e_{k,t})_{i=1\ldots|O|}$, where $o_{i,t} = \langle x_{i,t}, a_i^o \rangle$ denotes object i at time t, u_k and v_k are the receiver and sender of relation r_k respectively, and $\mathcal{N}_i$ denotes the relations where object i is the receiver.

Propagation networks. A limitation of INs is that at every time step t, it only considers local information in the graph G and cannot handle instantaneous changes in forces/accelerations in other bodies, which however is a common phenomenon in rigid-body dynamics. Li et al. [2019b] proposed propagation networks to approximate the instantaneous force changes by doing multi-step message passing. Specifically, they first employed the ideas on fast training of RNNs [Lei and Zhang, 2017, Bradbury et al., 2017] to encode the shared information beforehand and reuse them along the propagation steps. The encoders for objects are denoted as f_O^{enc} and the encoder for relations as f_R^{enc}, where we denote $c_{i,t}^o = f_O^{\text{enc}}(o_{i,t}), c_{k,t}^r = f_R^{\text{enc}}(o_{u_k,t}, o_{v_k,t}, a_k^r)$.

At time t, denote the propagating influence from relation k at propagation step l as $e_{k,t}^l$, and the propagating influence from object i as $h_{i,t}^l$. For step $1 \leq l \leq L$, propagation can be described as

$$\text{Step 0:} \qquad h_{i,t}^0 = \mathbf{0}, \quad i = 1 \ldots |O|, \tag{3.1}$$

$$\text{Step } l = 1, \ldots, L: \qquad e_{k,t}^l = f_R(c_{k,t}^r, h_{u_k,t}^{l-1}, h_{v_k,t}^{l-1}), k = 1 \ldots |R|, \tag{3.2}$$

$$h_{i,t}^l = f_O(c_{i,t}^o, \sum_{k \in \mathcal{N}_i} e_{k,t}^l, h_{i,t}^{l-1}), i = 1 \ldots |O|, \tag{3.3}$$

$$\text{Output:} \qquad \hat{o}_{i,t+1} = f_O^{\text{output}}(h_{i,t}^L), \quad i = 1 \ldots |O|, \tag{3.4}$$

where f_O denotes the object propagator, and f_R denotes the relation propagator.

3.3.2 Dynamic Particle Interaction Networks

Particle-based system is widely used in physical simulation due to its flexibility in modeling various types of objects [Macklin et al., 2014]. We extend existing systems that model object-level interactions to allow particle-level deformation. Consider object set $\{\mathbf{o}_i\}$, where each object $\mathbf{o}_i = \{o_i^k\}_{k=1...|\mathbf{o}_i|}$ is represented as a set of particles. We now define the graph on the particles and the rules for influence propagation.

Dynamic graph building. The vertices of the graph are the union of particles for all objects $O = \{o_i^k\}_{i=1...|O|,k=1...|\mathbf{o}_i|}$. The edges R between these vertices are dynamically generated over time to ensure efficiency and effectiveness. The construction of the relations is specific to environment and task, which we'll elaborate in Section 3.4. A common choice is to consider the neighbors within a predefined distance.

An alternative is to build a static, complete interaction graph, but it has two major drawbacks. First, it is not efficient. In many common physical systems, each particle is only interacting with a limited set of other particles (e.g., those within its neighborhood). Second, a static interaction graph implies a universal, continuous neural function approximator; however, many physical interactions involve discontinuous functions (e.g., contact). In contrast, using dynamic graphs empowers the model to tackle such discontinuity.

Hierarchical modeling for long-range dependence. Propagation networks [Li et al., 2019b] require a large L to handle long-range dependence, which is both inefficient and hard to train. Hence, we add one level of hierarchy to efficiently propagate the long-range influence among particles [Mrowca et al., 2018]. For each object that requires modeling of the long-range dependence (e.g., rigid-body), we cluster the particles into several non-overlapping clusters. For each cluster, we add a new particle as the cluster's root. Specifically, for each object $\mathbf{o}_i$ that requires hierarchical modeling, the corresponding roots are denoted as $\tilde{\mathbf{o}}_i = \{\tilde{o}_i^k\}_{k=1...|\tilde{\mathbf{o}}_i|}$, and the particle set containing all the roots is denoted as $\tilde{O} = \{\tilde{o}_i^k\}_{i=1...|O|,k=1...|\tilde{\mathbf{o}}_i|}$. We then construct an edge set $R_{\text{LeafToRoot}}$ that contains directed edges from each particle to its root, and an edge set $R_{\text{RootToLeaf}}$ containing directed edges from each root to its leaf particles. For each object that need hierarchical modeling, we add pairwise directed edges between all its

roots, and denote this edge set as $R_{\text{RootToRoot}}$.

We employ a multi-stage propagation paradigm: first, propagation among leaf nodes, $\phi_{\text{LeafToLeaf}}(\langle O, R\rangle)$; second, propagation from leaf nodes to root nodes, $\phi_{\text{LeafToRoot}}(\langle O \cup \tilde{O}, R_{\text{LeafToRoot}}\rangle)$; third, propagation between roots, $\phi_{\text{RootToRoot}}(\langle \tilde{O}, R_{\text{RootToRoot}}\rangle)$; fourth, propagation from root to leaf, $\phi_{\text{RootToLeaf}}(\langle O \cup \tilde{O}, R_{\text{RootToLeaf}}\rangle)$. The signals on the leaves are used to do the final prediction.

Applying to objects of various materials. We define the interaction graph and the propagation rules on particles for different types of objects as follows:

- *Rigid bodies.* All the particles in a rigid body are globally coupled; hence for each rigid object, we define a hierarchical model to propagate the effects. After the multi-stage propagation, we average the signals on the particles to predict a rigid transformation (rotation and translation) for the object. The motion of each particle is calculated accordingly. For each particle, we also include its offset to the center-of-mass to help determine the torque.

- *Elastic/Plastic objects.* For elastically deforming particles, only using the current position and velocity as the state is not sufficient, as it is not clear where the particle will be restored after the deformation. Hence, we include the particle state with the resting position to indicate the place where the particle should be restored. When coupled with plastic deformation, the resting position might change during an interaction. Thus, we also infer the motion of the resting position as a part of the state prediction. We use hierarchical modeling for this category but predict next state for each particles individually.

- *Fluids.* For fluid simulation, one has to enforce density and incompressibility, which can be effectively achieved by only considering a small neighborhood for each particle [Macklin and Müller, 2013]. Therefore, we do not need hierarchical modeling for fluids. We build edges dynamically, connecting a fluid particle to its neighboring particles. The strong inductive bias leveraged in the fluid particles allows good performance even when tested on data outside training distributions.

For the interaction between different materials, two directed edges are generated for any pairs of particles that are closer than a certain distance.

3.3.3 Control on the Learned Dynamics

Model-based methods offer many advantages when comparing with their model-free counterparts, such as generalization and sample efficiency. However, for cases where an accurate model is hard to specify or computationally prohibitive, a data-driven approach that learns to approximate the underlying dynamics becomes useful.

Function approximators such as neural networks are naturally differentiable, and off-the-shelf toolboxes for neural networks have emphasized the speed and ease of use of analytical gradients. We can thus rollout using the learned dynamics and optimize the control inputs by minimizing a loss between the simulated results and a target configuration via gradient descent. In cases where certain physical parameters are unknown, we can perform online system identification by minimizing the difference between the model's prediction and the reality. An outline of our algorithm can be found in Section B.1.

Model predictive control using shooting methods. Let's denote $\mathcal{G}_g$ as the goal and $\hat{u}_{1:T}$ be the control inputs, where T is the time horizon. The control inputs are part of the interaction graph, such as the velocities or the initial positions of a particular set of particles. We denote the resulting trajectory after applying $\hat{u}$ as $\mathcal{G} = \{G_i\}_{i=1:T}$. The task here is to determine the control inputs as to minimize the distance between the actual outcome and the specified goal $\mathcal{L}_{\text{goal}}(\mathcal{G}, \mathcal{G}_g)$.

Our dynamic particle interaction network does forward simulation by taking the dynamics graph at time t as input, and produces the graph at next time step, $\hat{G}_{t+1} = \Phi(G_t)$, where Φ is implemented as DPI-Net as described in the previous section. Let's denote the the history until time t as $\bar{\mathcal{G}} = \{G_i\}_{i=1\ldots t}$, and the forward simulation from time step t as $\hat{\mathcal{G}} = \{\hat{G}_i\}_{i=t+1\ldots T}$. The loss $\mathcal{L}_{\text{goal}}(\bar{\mathcal{G}} \cup \hat{\mathcal{G}}, \mathcal{G}_g)$ can be used to update the control inputs by doing stochastic gradient descent (SGD). This is known as the shooting method in trajectory optimization [Tedrake, 2009].

The learned model might deviate from the reality due to accumulated prediction errors. We use Model-Predictive Control (MPC) [Camacho and Alba, 2013] to stabilize the trajectory by doing forward simulation and updating the control inputs at every time step to compensate the simulation error.

Online adaptation. In many real-world cases, without actually interacting with the environment, inherent attributes such as mass, stiffness, and viscosity are not directly observable. DPI-Nets can estimate these attributes on the fly with SGD updates by minimizing the distance between the predicted future states and the actual future states $\mathcal{L}_{\text{state}}(\hat{G}_t, G_t)$.

3.4 Experiments

We evaluate our method on four different environments containing different types of objects and interactions. We will first describe the environments and show simulation results. We then present how the learned dynamics helps to complete control tasks in both simulation and the real world.

3.4.1 Environments

FluidFall (Figure 3-2a). Two drops of fluids are falling down, colliding, and merging. We vary the initial position and viscosity for training and evaluation.

BoxBath (Figure 3-2b). A block of fluids are flushing a rigid cube. In this environment, we have to model two different materials and the interactions between them. We randomize the initial position of the fluids and the cube to test the model's generalization ability.

FluidShake (Figure 3-2c). We have a box of fluids and the box is moving horizontally, The speed of the box is randomly selected at each time step. We vary the size of the box and volume of the fluids to test generalization.

RiceGrip (Figure 3-2d). We manipulate an object with both elastic and plastic deformation (e.g., sticky rice). We use two cuboids to mimic the fingers of a parallel gripper, where the gripper is initialized at a random position and orientation. During the simulation of one grip, the fingers will move closer to each other and then restore to its original positions. The model has to learn the interactions between the gripper and the "sticky rice", as well as the interactions within the "rice" itself.

We use all four environments in evaluating our model's performance in simulation. We use the rollout MSE as our metric. We further use the latter two for control, because they involve fully actuated external shapes that can be used for object manipulation. In FluidShake, the control task requires determining the speed of the box at each time step, in order to make the fluid match a target configuration within a time window; in RiceGrip, the control

task corresponds to select a sequence of grip configurations (position, orientation, closing distance) to manipulate the deformable object as to match a target shape. The metric for performance in control is the Chamfer distance between the manipulation results and the target configuration.

3.4.2 Physical Simulation

We present implementation details for dynamics learning in the four environment and perform ablation studies to evaluate the effectiveness of the introduced techniques.

Implementation details. For FluidFall, we dynamically build the graph by connecting each particle to its neighbors within a certain distance d. No hierarchical modeling is used.

For BoxBath, we model the rigid cube as in Section 3.3.2, using multi-stage hierarchical propagation. Two directed edges will be constructed between two fluid particles if the distance between them is smaller than d. Similarly, we also add two directed edge between rigid particles and fluid particles when their distance is smaller than d.

For FluidShake, we model fluid as in Section 3.3.2. We also add five external particles to represent the walls of the box. We add a directed edge from the wall particle to the fluid particle when they are closer than d. The model is a single propagation network, where the edges are dynamically constructed over time.

For RiceGrip, we build a hierarchical model for rice and use four propagation networks for multi-stage effect propagation (Section 3.3.2). The edges between the "rice" particles are dynamically generated if two particles are closer than d. Similar to FluidShake, we add two external particles to represent the two "fingers" and add an edge from the "finger" to the "rice" particle if they are closer than the distance d. As "rice" can deform both elastically and plastically, we maintain a resting position that helps the model restore a deformed particle. The output for each particle is a 6-dim vector for the velocity of the current observed position and the resting position. More training details for each environment can be found in Section B.4. Details for data generation are in Section B.3.

Results. Qualitative and quantitative results are in Figure 3-2 and Table 3.1. We compare our method (DPI-Net) with three baselines, Interaction Networks [Battaglia et al., 2016], HRN [Mrowca et al., 2018], and DPI-Net without hierarchy. Note that we use the same set of hyperparameters in our model for all four testing environments.

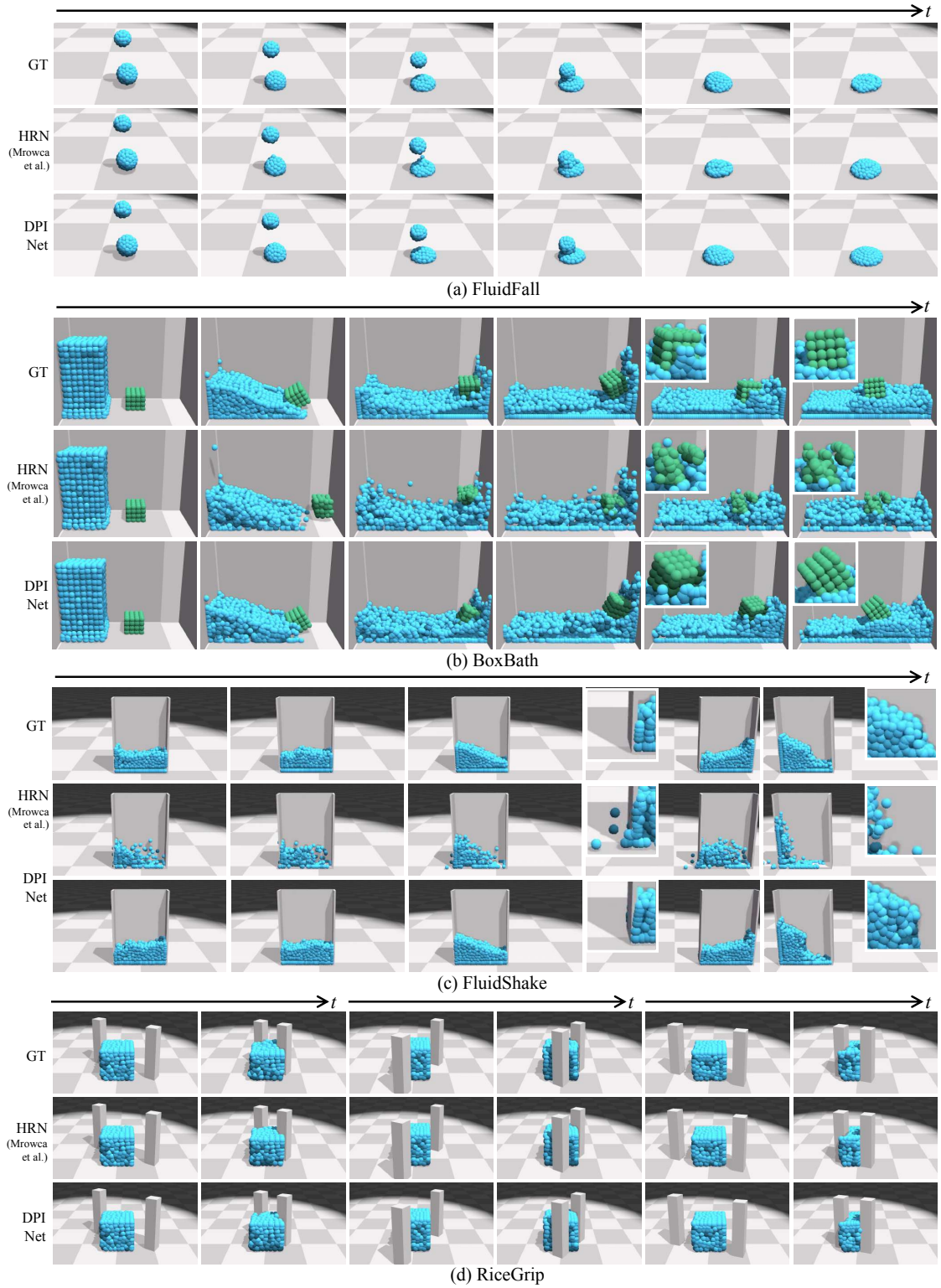


Figure 3-2: **Qualitative results on forward simulation.** We compare the ground truth (GT) and the rollouts from HRN [Mrowca et al., 2018] and our model (DPI-Net) in four environments (FluidFall, BoxBath, FluidShake, and RiceGrip). The simulations from our DPI-Net are significantly better. We provide zoom-in views for a few frames to show details. Please see our video for more empirical results.

Methods	FuildFall	BoxBath	FluidShake	RiceGrip
IN [Battaglia et al., 2016]	2.74 ± 0.56	N/A	N/A	N/A
HRN [Mrowca et al., 2018]	0.21 ± 0.04	3.62 ± 0.40	3.58 ± 0.77	0.17 ± 0.11
DPI-Net w/o hierarchy	**0.15 ± 0.03**	2.64 ± 0.69	**1.89 ± 0.36**	0.29 ± 0.13
DPI-Net	**0.15 ± 0.03**	**2.03 ± 0.41**	**1.89 ± 0.36**	**0.13 ± 0.07**

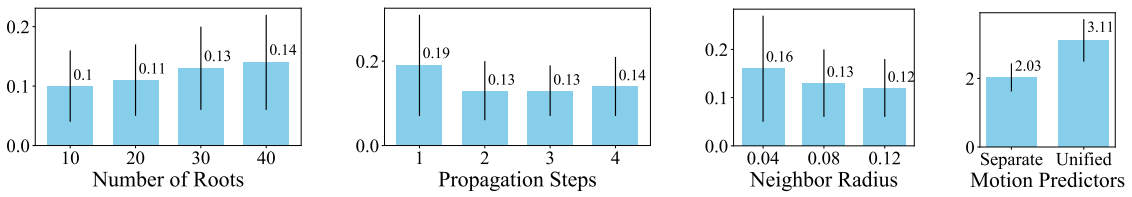
Table 3.1: **Quantitative results on forward simulation.** MSE ($\times 10^{-2}$) between the ground truth and model rollouts. The hyperparameters used in our model are fixed for all four environments. FluidFall and FluidShake involve no hierarchy, so DPI-Net performs the same as the variant without hierarchy. DPI-Net significantly outperforms HRN [Mrowca et al., 2018] in modeling fluids (BoxBath and FluidShake) due to the use of dynamic graphs.

Specifically, Interaction Networks (IN) consider a complete graph of the particle system. Thus, it can only operate on small environments such as FluidFall; it runs out of memory (12GB) for the other three environments. IN does not perform well, because its use of a complete graph makes training difficult and inefficient, and because it ignores influence propagation and long-range dependence.

Without a dynamic graph, modeling fluids becomes hard, because the neighbors of a fluid particle changes constantly. Table 3.1 shows that for environments that involve fluids (BoxBath and FluidShake), DPI-Net performs better than those with a static interaction graph. Our model also performs better in scenarios that involve objects of multiple states (BoxBath, Figure 3-2b), because it uses state-specific modeling. Models such as HRN [Mrowca et al., 2018] aim to learn a universal dynamics model for all states of matter. It is therefore solving a harder problem and, for this particular scenario, expected to perform not as well. When augmented with state-specific modeling, HRN's performance is likely to improve, too. Without hierarchy, it is hard to capture long-range dependence, leading to performance drop in environments that involve hierarchical object modeling (BoxBath and RiceGrip).

Appendix B.2 includes results on scenarios outside the training distribution (e.g., more particles). DPI-Net performs well on these out-of-sample cases, successfully leveraging the inductive bias.

Ablation studies. We also test our model's sensitivity to hyperparameters. We consider three of them: the number of roots for building hierarchy, the number of propagation steps L, and the size of the neighborhood d. We test them in RiceGrip. As can be seen from the results shown in Figure 3-3a, DPI-Nets can better capture the motion of the "rice" by using fewer roots, on which the information might be easier to propagate. Longer propagation



(a) RiceGrip (b) BoxBath

Figure 3-3: **Ablation studies.** We perform ablation studies to test our model's robustness to hyperparameters. The performance is evaluated by the mean squared distance ($\times 10^{-2}$) between the ground truth and model rollouts. (a) We vary the number of roots when building hierarchy, the propagation step L during message passing, and the size of the neighborhood d. (b) In BoxBath, DPI-Nets use separate motion predictors for fluids and rigid bodies. Here we compared with a unified motion predictor.

steps do not necessarily lead to better performance, as they increases training difficulty. Using larger neighborhood achieves better results, but makes computation slower. Using one TITAN Xp, each forward step in RiceGrip takes 30ms for $d = 0.04$, 33ms for $d = 0.08$, and 40ms for $d = 0.12$.

We also perform experiments to justify our use of different motion predictors for objects of different states. Figure 3-3b shows the results of our model *vs*. a unified dynamics predictor for all objects in BoxBath. As there are only a few states of interest (solids, liquids, and soft bodies), and their physical behaviors are drastically different, it is not surprising that DPI-Nets, with state-specific motion predictors, perform better, and are equally efficient as the unified model (time difference smaller than 3ms per forward step).

3.4.3 Control

We leverage dynamic particle interaction network for control tasks in both simulation and real world. Because trajectory optimization using shooting method can easily stuck to a local minimum, we first randomly sample N_{sample} control sequences, and select the best performing one according to the rollouts of our learned model. We then optimize it via shooting method using our model's gradients. We also use online system identification to further improve the model's performance. Figure 3-4 and Figure 3-5 show qualitative and quantitative results, respectively. More details of the control algorithm can be found in Section B.5.

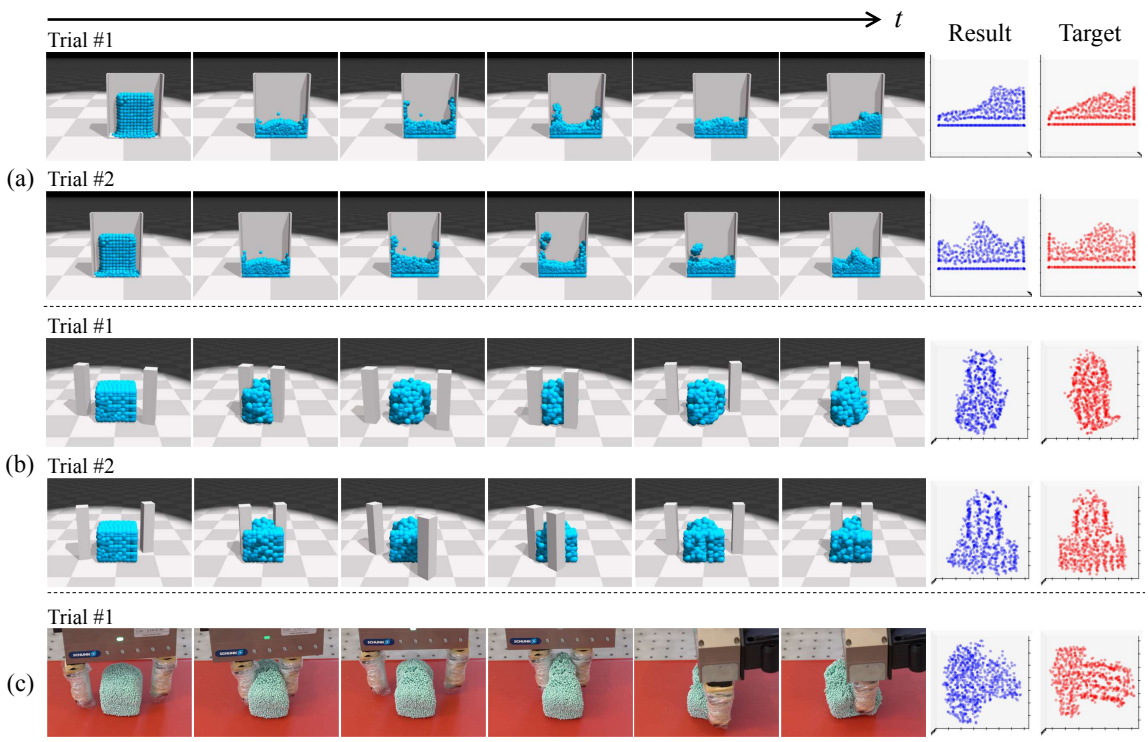
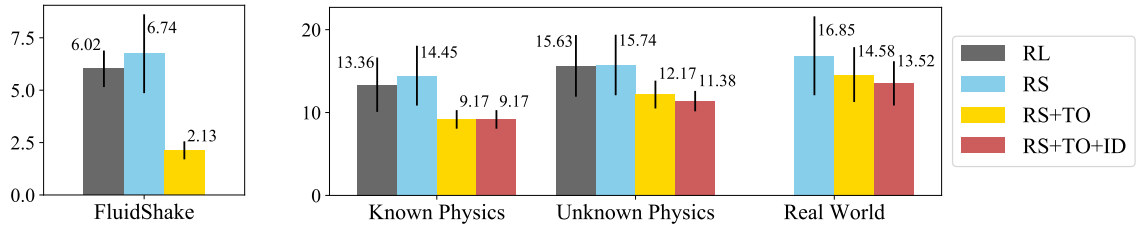


Figure 3-4: **Qualitative results on control.** (a) FluidShake - Manipulating a box of fluids to match a target shape. The Result and Target indicate the fluid shape when viewed from the cutaway view. (b) RiceGrip - Gripping a deformable object and molding it to a target shape. (c) RiceGrip in Real World - Generalize the learned dynamics and the control algorithms to the real world by doing online adaptation. The last two columns indicate the final shape viewed from the top.

FluidShake. We aim to control the speed of the box to match the fluid particles to a target configuration. We compare our method (RS+TO) with random search over the learned model (without trajectory optimization - RS) and Model-free Deep Reinforcement Learning (Actor-Critic method optimized with PPO [Schulman et al., 2017] (RL). Figure 3-5a suggests that our model-based control algorithm outperforms both baselines with a large margin. Also RL is not sample-efficient, requiring more than 10 million time steps to converge while ours requires 600K time steps.

RiceGrip. We aim to select a sequence of gripping configurations (position, orientation, and closing distance) to mold the "sticky rice" to a target shape. We also consider cases where the stiffness of the rice is unknown and need to be identified. Figure 3-5b shows that our dynamic particle interaction network with system identification performs the best, and is much more efficient than RL (150K *vs*. 10M time steps).



(a) (b) RiceGrip

Figure 3-5: **Quantitative results on control.** We show the results on control (as evaluated by the Chamfer distance ($\times 10^{-2}$) between the manipulated result and the target) for (a) FluidShake and (b) RiceGrip by comparing with four baselines. **RL**: Model-free deep reinforcement learning optimized with PPO; **RS**: Random search the actions from the learned model and select the best one to execute; RS + **TO**: Trajectory optimization augmented with model predictive control; RS + TO + **ID**: Online system identification by estimating uncertain physical parameters during run time.

RiceGrip in the real world. We generalize the learned model and control algorithm for RiceGrip to the real world. We first reconstruct object geometry using a depth camera mounted on our Kuka robot using TSDF volumetric fusion [Curless and Levoy, 1996]. We then randomly sampled N_{fill} particles within the object mesh as the initial configuration for manipulation. Figure 3-4c and Figure 3-5b shows that, using DPI-Nets, the robot successfully adapts to the real world environment of unknown physical parameters and manipulates a deformable foam into various target shapes. The learned policy in RiceGrip does not generalize to the real world due to domain discrepancy, and outputs invalid gripping configurations.

3.5 Discussion

We have demonstrated that a learned particle dynamics model can approximate the interaction of diverse objects, and can help to solve complex manipulation tasks of deformable objects. Our system requires standard open-source robotics and deep learning toolkits, and can be potentially deployed in household and manufacturing environments. Robot learning of dynamic scenes with particle-based representations shows profound potential due to the generalizability and expressiveness of the representation. Our study helps lay the foundation for it.

Chapter 4

Learning Latent-Space Dynamics with Neural Radiance Field

Now we move forward by considering the modeling and manipulation of dynamical systems with extremely high degrees of freedom (DoF) purely from visual observations, like manipulating a box of fluid with ice cubes floating on it (i.e., fluid-body interactions). In this case, it is hard to estimate the full state information of the particle set when the only inputs are the RGB images. We also do not know how to use keypoints (typically a sparse set of points attached to semantically-meaningful parts of an object) in this case, as the fluid is of extremely high DoF and constantly changes its shape during interactions.

This chapter takes on this challenge and aims to learn models for highly dynamic 3D scenes purely from 2D visual observations. Our model combines Neural Radiance Fields (NeRF) and time contrastive learning with an autoencoding framework, which learns viewpoint-invariant 3D-aware scene representations (z_t in Equation 1.2, but in this chapter, we overwrite its notation as $\boldsymbol{s}_t$). We show that a dynamics model, constructed over the learned representation space, enables visuomotor control for challenging manipulation tasks involving both rigid bodies and fluids, where the target can be specified in a viewpoint different from what the robot operates on. Moreover, when coupled with an auto-decoding framework, it can even support goal specification from camera viewpoints that are *outside the training distribution*. We further demonstrate the richness of the learned 3D dynamics model by performing future prediction and novel view synthesis. Finally, we provide detailed ablation studies regarding different system designs and qualitative analysis of the learned representations. Video

illustrations can be found on our project page: https://3d-representation-learning.github.io/nerf-dy/.

This chapter was originally published as Li et al. [2022] in the Conference on Robot Learning (CoRL) 2021 as **Oral Presentation**, in which Shuang Li has also made significant contributions along with co-authors Vincent Sitzmann, Pulkit Agrawal, and Antonio Torralba.

4.1 Introduction

Existing state-of-the-art model-based systems operating from vision typically treat images as 2D grids of pixels [Ebert et al., 2018, Hafner et al., 2019a, Schrittwieser et al., 2020]. The world, however, is three-dimensional. Modeling the environment from 3D enables amodal completion and allows the agents to operate from different views. Therefore, it is desirable to obtain good 3D-aware representations of the environment from 2D observations to achieve better task performance when an accurate inference of 3D information is essential, which can further make it easier to specify tasks, learn from third-person videos, etc.

One of the core questions of model learning in robotic manipulation is how to determine the state representation for learning the dynamics model. The desired representation should make it easy to capture the environment dynamics, exhibit a good 3D understanding of the objects in the scene, and be applicable to diverse object sets such as rigid or deformable objects and fluids. One line of prior work learns the dynamics model directly in the image pixel space [Finn et al., 2016, Ebert et al., 2017, Yen-Chen et al., 2020, Suh and Tedrake, 2020]. However, modeling dynamics in such a high-dimensional space is challenging, and these methods typically generate blurry images when performing the long-horizon future predictions. As has been exemplified in Chapter 2, another line of work focused on only predicting task-relevant features identified as keypoints [Manuelli et al., 2019, 2021, Wang et al., 2020, Gao and Tedrake, 2021, Li et al., 2020b]. Such models perform well in terms of category-level generalization, i.e., the same set of keypoints can represent different instances within the same category, but are not sufficient to model objects with large variations like fluids and granular materials. Other methods learn dynamics in the latent space [Watter et al., 2015, Agrawal et al., 2016, Hafner et al., 2019b,a, Schrittwieser et al., 2020]. However, the majority of these methods learn dynamics models using 2D convolutional neural networks

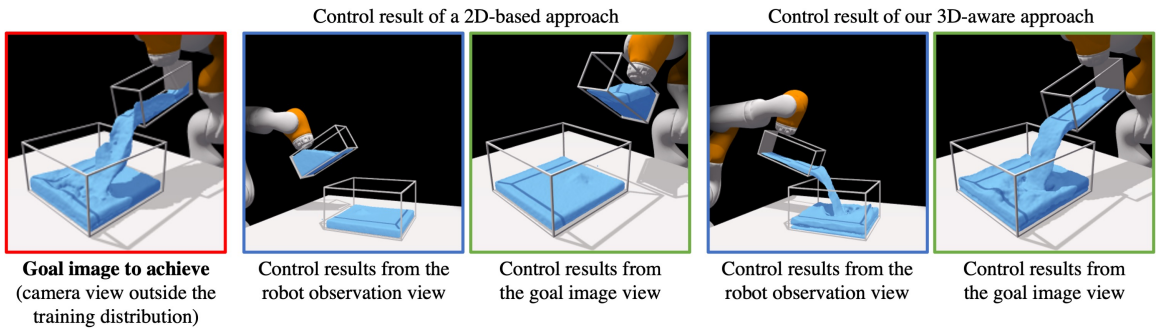


Figure 4-1: **Comparison of the control results between a 2D-based baseline and our 3D-aware approach.** The task here is to achieve the configuration shown on the left, observed from a viewpoint that is outside the training distribution. The agent only takes a single-view visual observation as input (images with blue frames) from a viewpoint that is vastly different from the goal. Our method generalizes well in this scenario and outperforms the 2D-based baseline, demonstrating the benefits of the learned 3D-aware scene representations.

and reconstruction loss – which has the same problem as predicting dynamics in the image space, i.e., their learned representations lack equivariance to 3D transformations. Time contrastive networks [Sermanet et al., 2018], on the other hand, aim to learn viewpoint-invariant representations from multi-view inputs, but do not require detailed modeling of 3D contents. As a result, previously unseen scene configurations and camera poses are out-of-distribution samples for the state estimator. As we will see, this leads to wrong state estimates and results in faulty control trajectories.

Meanwhile, recent work in computer vision has made impressive progress in the learning of 3D-structured neural scene representations. These approaches allow inference of 3D structure and appearance, trained only given 2D observations, either by overfitting on a single scene [Sitzmann et al., 2019a, Lombardi et al., 2019, Mildenhall et al., 2020] or by generalizing across scenes [Sitzmann et al., 2019b, Tung et al., 2019, 2020]. Through their 3D inductive bias, the scene representations inferred by these models encode the scene contents with better accuracy and are invariant to changes in camera perspectives. It is desirable to push these ideas further to obtain a deeper understanding of how these methods, which directly reason over 3D, can bring in new characteristics and how they can be beneficial for dynamics modeling and complicated control tasks.

In this chapter, we aim to leverage recently proposed 3D-structure-aware implicit neural scene representations for visuomotor control tasks. We thus propose to embed neural radiance fields [Mildenhall et al., 2020] in an auto-encoder framework, enabling tractable inference of

the 3D-structure-aware scene state for dynamic environments. By additionally enforcing a time contrastive loss on the estimated states, we ensure that the learned state representations are viewpoint-invariant. We then train a dynamics model that predicts the evolution of the state space conditioned on the input action, enabling us to perform control in the learned state space. Though the representation itself is grounded in the 3D implicit field, the convolutional encoder is not. At test time, we overcome this limitation by performing inference-via-optimization [Park et al., 2019, Sitzmann et al., 2019b], enabling accurate state estimation even for out-of-distribution camera poses and, therefore, control of tasks where the goal view is specified in an entirely unseen camera perspective. These contributions enable us to perform model-based visuomotor control of complex scenes, modeling 3D dynamics of both rigid objects and fluids. Through comparison with various baselines, the learned representation from our model is more precise at describing the contents of 3D scenes, which allows it to accomplish control tasks involving rigid objects and fluids with significantly better accuracy (Figure 4-1). Please see our supplementary video for better visualization.

We summarize our contributions as follows: (i) We extend an autoencoding framework with a neural radiance field rendering module and time contrastive learning that allows us to learn 3D-aware scene representations for dynamics modeling and control purely from visual observations. (ii) By incorporating the auto-decoder mechanism at test time, our framework can adjust the learned representation and accomplish the control tasks with the goal specified from camera viewpoints outside the training distribution. (iii) We are the first to augment neural radiance fields using a time-invariant dynamics model, supporting future prediction and novel view synthesis across a wide range of environments with different types of objects.

4.2 Related Work

3D scene representation learning. Prior work leverages the latent spaces of autoencoder-like models as learned representations of the underlying 3D scene to enable novel view synthesis from a single image [Worrall et al., 2017, Tatarchenko et al., 2015]. Eslami et al. [2018] embed this approach in a probabilistic framework. To endow models with 3D structure, voxelgrids can be leveraged as neural scene representations [Rezende et al., 2016, Nguyen-Phuoc et al., 2018, Sitzmann et al., 2019a, Tung et al., 2019, Zhu et al., 2018], while others have tried to predict particle sets from images [Li et al., 2020a] (which will be discussed in

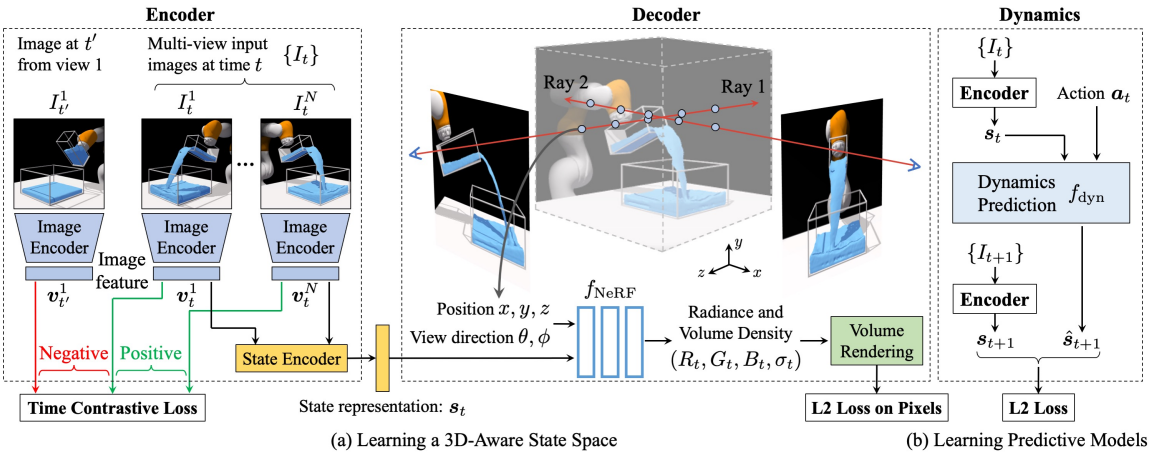


Figure 4-2: **Overview of the training procedure. Left:** an encoder that maps the input images into a latent scene representation. The images are first sent to an image encoder to generate the image feature representations $\boldsymbol{v}$. Then we combine the image features from the same time step using a state encoder to obtain the state representation $\boldsymbol{s}_t$. A time contrastive loss is applied to enable our model to be invariant to camera viewpoints. **Middle:** a decoder that takes the scene representation as input and generates the visual observation conditioned on a given viewpoint. We use an L2 loss to ensure the reconstructed image to be similar to the ground truth image. **Right:** a dynamics model that predicts the future scene representations $\hat{\boldsymbol{s}}_{t+1}$ by taking in the current representation $\boldsymbol{s}_t$ and action $\boldsymbol{a}_t$. We use an L2 loss to enforce the predicted latent representation to be similar to the scene representation $\boldsymbol{s}_{t+1}$ extracted from the true visual observation I_{t+1}.

more detail in Chapter 7) or embed an explicit 3D representation to enable inference from never-before-seen viewpoints [Saponaro et al., 2019]. Sitzmann et al. [2019b] propose to learn neural implicit representations of 3D shape and appearance supervised with posed 2D images via a differentiable renderer. Generalizing across neural implicit representations can also be realized by local conditioning on CNN features [Saito et al., 2019, Trevithick and Yang, 2020, Yu et al., 2020], but this does not learn a global representation of the scene state. Alternatively, gradient-based meta-learning has been proposed for faster inference of implicit neural representations [Sitzmann et al., 2020]. Deformable scenes can be modeled by transporting input coordinates to neural implicit representations with an implicitly represented flow field or time-variant latent code [Niemeyer et al., 2019, Park et al., 2020, Pumarola et al., 2020, Tretschk et al., 2020, Li et al., 2020c, Du et al., 2020, Xian et al., 2021, Ost et al., 2021, Li et al., 2021]; however, they typically fit one trajectory and cannot handle different initial conditions and external action inputs, limiting their use in control.

Model-based RL in robotic manipulation. We can categorize model-based RL methods by whether they use physics-based or data-driven models, and whether they assume full state

access or only visual observation. Methods that rely on physics-based models typically assume full-state information of the environment [Hogan and Rodriguez, 2016, Zhou et al., 2019] and require the knowledge of the object models, making them hard to generalize to novel objects or partially observable scenarios. For data-driven models, people have attempted to learn a dynamics model for closed-loop planar pushing [Bauza et al., 2018] or dexterous manipulation [Nagabandi et al., 2020]. Schenck and Fox [2017, 2018a] tackle a similar fluid pouring task via closed-loop simulation. Although they have achieved impressive results, they rely on state estimators customized for specific tasks, limiting their applicability to more general and diversified manipulation tasks.

Various model-based RL methods have been proposed to learn state representations from visual observations, such as image-space dynamics [Finn et al., 2016, Ebert et al., 2017, 2018, Suh and Tedrake, 2020], keypoint representation [Kulkarni et al., 2019, Manuelli et al., 2021, Li et al., 2020b], and low-dimensional latent space [Watter et al., 2015, Hafner et al., 2019b,a, Schrittwieser et al., 2020]. Some works learn a meaningful representation space using reconstruction loss [Hafner et al., 2019b,a]. Others jointly train the forward and inverse dynamics models [Agrawal et al., 2016], or use time contrastive loss to regularize the latent embedding [Sermanet et al., 2018]. We differ from the previous methods by explicitly incorporating a 3D volumetric rendering process during training.

4.3 3D-Aware Representation Learning for Dynamics Modeling

Inspired by Neural Radiance Fields (NeRF) [Mildenhall et al., 2020], we propose a framework that learns a viewpoint-invariant model for dynamic environments. As shown in Figure 4-2, our framework has three parts: (1) an encoder that maps the input images into a latent state representation, (2) a decoder that generates an observation image under a certain viewpoint based on the state representation, and (3) a dynamics model that predicts the future state representations based on the current state and the input action.

4.3.1 3D-Aware Scene Representation Learning

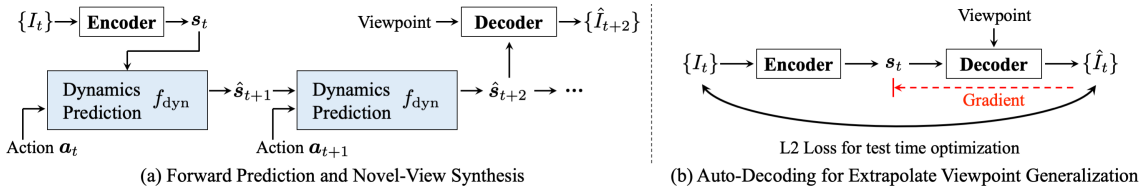
Neural radiance field. Given a 3D point $\boldsymbol{x} \in \mathbb{R}^3$ in a scene and a viewing direction unit vector $\boldsymbol{d} \in \mathbb{R}^3$ from a camera, NeRF learns to predict a volumetric radiance field. This is represented using a differentiable rendering function f_{NeRF} that predicts the corresponding

density σ and RGB color $\boldsymbol{c}$ using $f_{\text{NeRF}}(\boldsymbol{x}, \boldsymbol{d}) = (\sigma, \boldsymbol{c})$. To render the color of an image pixel, NeRF integrates the information along the camera ray using $\hat{\boldsymbol{C}}(\boldsymbol{r}) = \int_{h_{\text{near}}}^{h_{\text{far}}} T(h)\sigma(h)\boldsymbol{c}(h)dh$, where $\boldsymbol{r}(h) = \boldsymbol{o} + h\boldsymbol{d}$ is the camera ray with its origin $\boldsymbol{o} \in \mathbb{R}^3$ and unit direction vector $\boldsymbol{d} \in \mathbb{R}^3$, and $T(h) = \exp(-\int_{h_{\text{near}}}^{h} \sigma(s)ds)$ is the accumulated transparency between the pre-defined near depth h_{near} and far depth h_{far} along that camera ray. The mean squared error between the reconstructed color $\hat{\boldsymbol{C}}$ and the ground truth $\boldsymbol{C}$ is:

$$\mathcal{L}_{\text{rec}} = \sum_{\boldsymbol{r}} \|\hat{\boldsymbol{C}}(\boldsymbol{r}) - \boldsymbol{C}(\boldsymbol{r})\|_2^2. \tag{4.1}$$

Neural radiance field for dynamic scenes. One key limitation of NeRF is that it assumes the scene is static. For a dynamic scene, it must learn a separate radiance field f_{NeRF} for each time step. This severely limits the ability of NeRF to be used in planning and control, as it is unable to handle dynamic scenes with different initial configurations or input action sequences. While other models have shown generalization across scenes [Sitzmann et al., 2019b, Niemeyer et al., 2020], it's unclear how they can be used in visuomotor control. To enable f_{NeRF} to model dynamic scenes, we learn an encoding function f_{enc} that maps the visual observations to a feature representation $\boldsymbol{s}$ for each time step and learn the volumetric radiance field decoding function based on $\boldsymbol{s}$. Let $\{I_t\}$ denotes the set of 2D images that capture a 3D scene at time t from one or more camera viewpoints. The image taken from the i^{th} viewpoint is represented as I_t^i. We use ResNet-18 [He et al., 2016] to extract a feature vector for each image. We take the output of ResNet-18 before the pooling layer and send it to a fully-connected layer, resulting in a 256 dimension image feature $\boldsymbol{v}_t^i$. This image feature is concatenated with the corresponding camera viewpoint information (a 16-D vector obtained by flattening the camera view matrix) and processed using a small multilayer perceptron (MLP) to generate the final image feature. The scene representation $\boldsymbol{s}_t$ at time t is generated by first averaging the image features across multiple camera viewpoints, then being encoded using another small MLP and normalized to have a unit L2 norm.

Given a 3D point $\boldsymbol{x}$, a viewing direction unit vector $\boldsymbol{d}$, and a scene representation $\boldsymbol{s}_t$, we learn a function $f_{\text{dec}}(\boldsymbol{x}, \boldsymbol{d}, \boldsymbol{s}_t) = (\sigma_t, \boldsymbol{c}_t)$ to predict the radiance field represented by the density σ_t and RGB color $\boldsymbol{c}_t$. Similar to NeRF, we use the integrated information along the camera ray to render the color of image pixels from an input viewpoint and then compute the image reconstruction loss using Equation 4.1. During each training iteration, we render two images from different viewpoints to calculate more accurate gradient updates. f_{dec} depends



(a) Forward Prediction and Novel-View Synthesis

(b) Auto-Decoding for Extrapolate Viewpoint Generalization

Figure 4-3: **Forward prediction and viewpoint extrapolation.** (a) We first feed the input image(s) at time t to the encoder to derive the scene representation $\boldsymbol{s}_t$. The dynamics model then takes $\boldsymbol{s}_t$ and the corresponding action sequence as input to iteratively predict the future. The decoder synthesizes the visual observation conditioned on the predicted state representation and an input viewpoint. (b) We propose an auto-decoding inference-via-optimization framework to enable extrapolated viewpoint generalization. Given an input image I_t taken from a viewpoint outside the training distribution, the encoder first predicts the scene representation $\boldsymbol{s}_t$. Then the decoder reconstructs the observation $\hat{I}_t$ from $\boldsymbol{s}_t$ and the camera viewpoint from I_t. We calculate the L2 distance between I_t and $\hat{I}_t$ and backpropagate the gradient through the decoder to update $\boldsymbol{s}_t$. The updating process is repeated for K iterations, resulting in a more accurate $\boldsymbol{s}_t$ of the underlying 3D scene.

on the scene representation $\boldsymbol{s}_t$, forcing it to encode the 3D contents of the scene to support rendering from different camera poses.

Time contrastive learning. To enable the image encoder to be viewpoint invariant, we regularize the feature representation of each image $\boldsymbol{v}_t^i$ using multi-view time contrastive loss (TCN) [Sermanet et al., 2018] (see Figure 4-2a). The TCN loss encourages features of images from different viewpoints at the same time step to be similar, while repulsing features of images from different time steps to be dissimilar. More specifically, given a time step t, we randomly select one image I_t^i as the anchor and extract its image feature $\boldsymbol{v}_t^i$ using the image encoder. Then we randomly select one positive image from the same time step but different camera viewpoint $I_t^{i'}$ and one negative image from a different time step but the same viewpoint $I_{t'}^i$. We use the same image encoder to extract their image features $\boldsymbol{v}_t^{i'}$ and $\boldsymbol{v}_{t'}^i$. Similar to Sermanet et al. [2018], we minimize the following time contrastive loss:

$$\mathcal{L}_{\text{tc}} = \max\left(\|\boldsymbol{v}_t^i - \boldsymbol{v}_t^{i'}\|_2^2 - \|\boldsymbol{v}_t^i - \boldsymbol{v}_{t'}^i\|_2^2 + \alpha, 0\right), \tag{4.2}$$

where α is a hyper-parameter denoting the *margin* between the positive and negative pairs.

4.3.2 Learning the Predictive Model

After we have obtained the latent state representation $\boldsymbol{s}$, we use supervised learning to estimate the forward dynamics model, $\hat{\boldsymbol{s}}_{t+1} = f_{\text{dyn}}(\boldsymbol{s}_t, \boldsymbol{a}_t)$. Given $\boldsymbol{s}_t$ and a sequence of actions $\{\boldsymbol{a}_t, \boldsymbol{a}_{t+1}, \dots\}$, we predict H steps in the future by iteratively feeding in actions into

the one-step forward model. We implement f_{dyn} as an MLP network which is trained by optimizing the following loss function:

$$\mathcal{L}_{\text{dyn}} = \sum_{h=1}^{H} \|\hat{\boldsymbol{s}}_{t+h} - \boldsymbol{s}_{t+h}\|_2^2, \quad \text{where} \quad \hat{\boldsymbol{s}}_{t+h} = f_{\text{dyn}}(\hat{\boldsymbol{s}}_{t+h-1}, \boldsymbol{a}_{t+h-1}), \quad \hat{\boldsymbol{s}}_t = \boldsymbol{s}_t. \tag{4.3}$$

We define the final loss as a combination of the image reconstruction loss, the time contrastive loss, and the dynamics prediction loss: $\mathcal{L} = \mathcal{L}_{\text{rec}} + \mathcal{L}_{\text{tc}} + \mathcal{L}_{\text{dyn}}$. We first train the encoder f_{enc} and decoder f_{dec} together using stochastic gradient descent (SGD) by minimizing $\mathcal{L}_{\text{rec}}$ and $\mathcal{L}_{\text{tc}}$, which makes sure that the learned scene representation $\boldsymbol{s}$ encodes the 3D contents and is viewpoint-invariant. We then fix the encoder parameters, and train the dynamics model f_{dyn} by minimizing $\mathcal{L}_{\text{dyn}}$ using SGD. Please see the supplementary materials for the network architecture and training details.

4.4 Visuomotor Control

4.4.1 Online Planning for Closed-Loop Control

When given the goal image I^{goal} and its associated camera pose, we can feed them through the encoder f_{enc} to get the state representation for the goal configuration $\boldsymbol{s}^{\text{goal}}$. We use the same method to compute the state representation for the current scene configuration $\boldsymbol{s}_1$. The goal of the online planning problem is to find an action sequence $\boldsymbol{a}_1, \ldots, \boldsymbol{a}_{T-1}$ that minimizes the distance between the predicted future representation and the goal representation at time T. As shown in Figure 4-3a, given a sequence of actions, our model can iteratively predict a sequence of latent state representations. The latent-space dynamics model can then be used for downstream closed-loop control tasks via online planning with model-predictive control (MPC). We formally define the online planning problem as follows:

$$\min_{\boldsymbol{a}_1, \ldots, \boldsymbol{a}_{T-1}} \quad \|\boldsymbol{s}^{\text{goal}} - \hat{\boldsymbol{s}}_T\|_2^2, \qquad \text{s.t.} \quad \hat{\boldsymbol{s}}_1 = \boldsymbol{s}_1, \hat{\boldsymbol{s}}_{t+1} = f_{\text{dyn}}(\hat{\boldsymbol{s}}_t, \boldsymbol{a}_t). \tag{4.4}$$

Many existing off-the-shelf model-based RL methods can be used to solve the MPC problem [Ebert et al., 2017, Nagabandi et al., 2020, Hafner et al., 2019b, Finn et al., 2016, Manuelli et al., 2021, Li et al., 2019a,b]. We experimented with random shooting, gradient-based trajectory optimization, cross-entropy method, and model-predictive path integral (MPPI) planners [Williams et al., 2015a] and found that MPPI performed the best. In

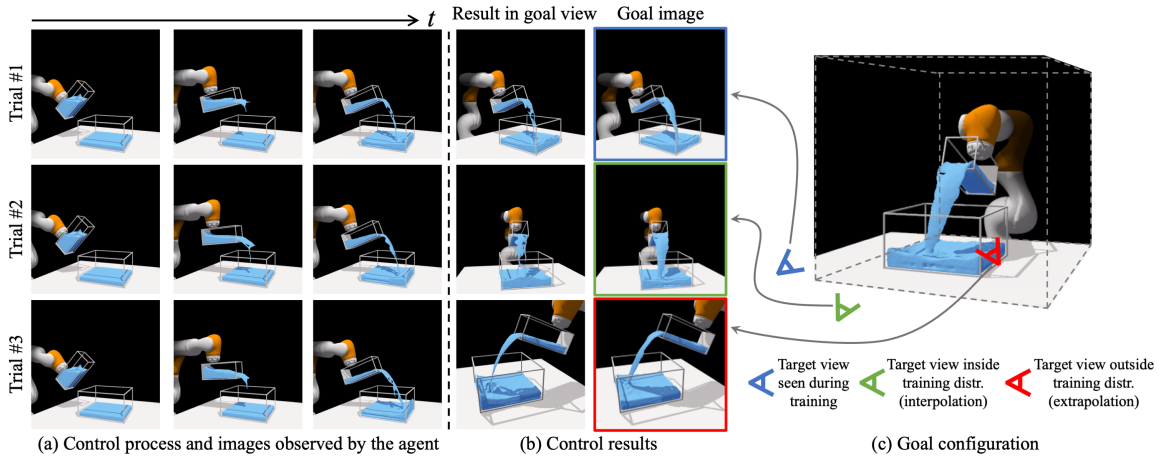


Figure 4-4: **Qualitative control results of our method on three types of testing scenarios.** The image on the right shows the target configuration we aim to achieve. The left three columns show the control processes, which are also the input images to the agent. The fourth column is the control results from the same viewpoint as the goal image. Trial #1 specifies the goal using a different viewpoint from the agent's but has been encountered during training. Trial #2 uses a goal view that is an interpolation of training viewpoints. Trial #3 uses an extrapolated viewpoint that is outside the training distribution. Our method performs well in all settings.

our experiments, we specify the action space as the position and orientation of the arm's end-effector. Then, the joint angle of the arm is calculated via inverse kinematics. Please check our supplementary materials for more details on how we solve the planning task.

4.4.2 Auto-Decoder for Viewpoint Extrapolation

End-to-end visuomotor agents can undergo significant performance drop when the test-time visual observations are captured from camera poses outside the training distribution. The convolutional image encoder suffers from the same problem as it is not equivariant to changes in the camera pose, meaning it has a hard time generalizing to out-of-distribution camera views. As shown in Figure 4-3b, when encountered an image from a viewpoint outside training distribution, with a pixel distribution vastly different from what the model is trained on, passing it through the encoder f_{enc} will give us an amortized estimation of the scene representation $\boldsymbol{s}_t$. It has a high chance that the decoded image is different from the ground truth as the viewpoint has never been encountered during training.

We fix this problem at test time by applying the inference-by-optimization (also named as an auto-decoding) framework that backpropagates through the volumetric renderer and the neural implicit representation into the state estimate [Park et al., 2019, Sitzmann et al., 2019b].

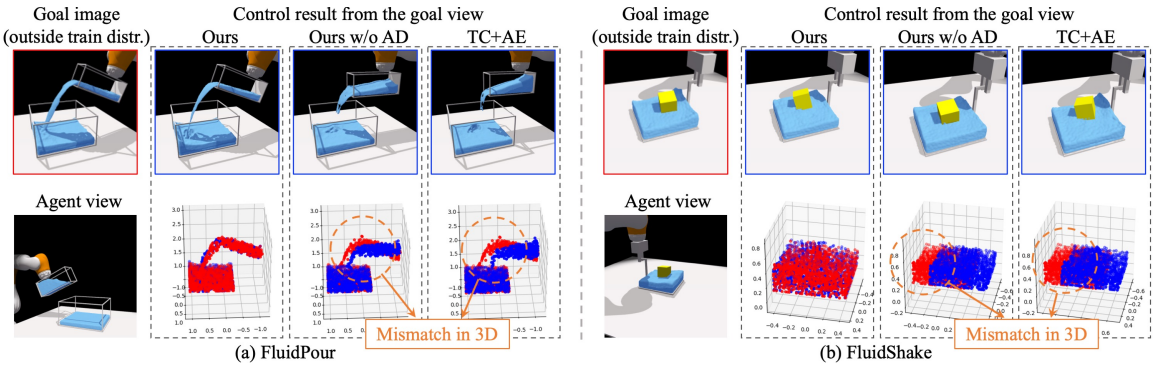


Figure 4-5: **Qualitative comparisons between our method and baseline approaches on control tasks.** We show the closed-loop control results on the FluidPour and FluidShake environments. The goal image viewpoint (top-left image of each block) is outside the training distribution and is different from the viewpoint observed by the agent (bottom-left image of each block). Our final control results are much better than a variant that does not perform auto-decoding test-time optimization (Ours w/o AD) and the best-performing baseline (TC+AE), both of which fail to accomplish the task and their control results (blue points) exhibit an apparent deviation from the target configuration (red points) when measured in the 3D points space of the fluids and floating cube.

This is inspired by the fact that the rendering function, $f_{\text{dec}}(\boldsymbol{x}, \boldsymbol{d}, \boldsymbol{s}_t) = (\sigma_t, \boldsymbol{c}_t)$ is viewpoint equivariant, where the output only depends on the state representation $\boldsymbol{s}_t$, the location $\boldsymbol{x}$, and the ray direction $\boldsymbol{d}$, meaning that the output is invariant to the camera position along the camera ray, i.e., even we move the camera closer or farther away along the camera ray, f_{dec} still tends to generate the same results. We leverage this property and calculate the L2 distance between the input image and the reconstructed image $\mathcal{L}_{\text{ad}} = \|I_t - \hat{I}_t\|_2^2$, and then update the scene representation $\boldsymbol{s}_t$ using stochastic gradient descent. We repeat this updating process K times to derive the state representation of the underlying 3D scene. Note that this update only changes the scene representation while keeping the parameters in the decoder fixed. The resulting representation is used as $\boldsymbol{s}^{\text{goal}}$ in Equation 4.4 to solve the online planning problem.

4.5 Experiments

Environments. We consider the following four environments involving both fluid and rigid objects to evaluate the proposed model and baseline approaches. The environments are simulated using NVIDIA FleX [Macklin et al., 2014]. (1) FluidPour (Figure 4-7a): This environment contains a fully-actuated cup that pours fluids into a container at the

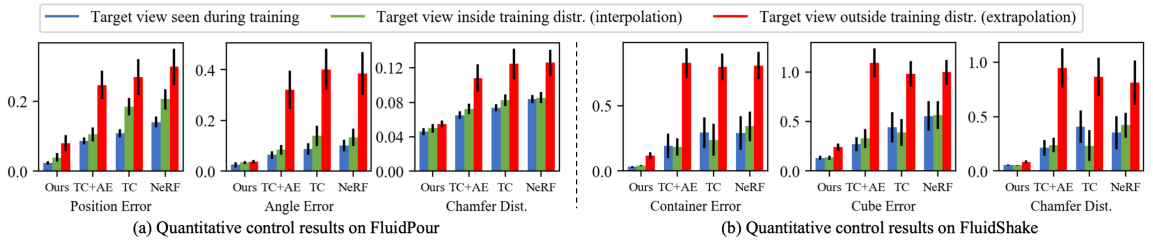


Figure 4-6: **Quantitative comparisons between our method and baselines on the FluidPour and FluidShake.** In each environment, we compare the results using three different evaluation metrics under three settings, i.e., (1) the target image view seen during training, (2) the target image view is inside the training distribution but not seen during training (interpolation), and (3) the target image view is outside the training distribution (extrapolation). The height of the bars indicates the mean, and the error bar denotes the standard error. Our model significantly outperforms all baselines under all testing settings.

bottom. (2) FluidShake (Figure 4-7b): A fully-actuated container moves on a 2D plane. Inside the container are fluids and a rigid cube floating on the surface. (3) RigidStack (Figure 4-7c): Three rigid cubes form a vertical stack and are released from a certain height but in different horizontal positions. They fall down and collide with each other and the ground. (4) RigidDrop (Figure 4-7d): A cube falls down from a certain height. There is a container fixed at a random position on the ground. The cube either falls into the container or bounces out. We use 20 camera views for all environments and generated $1,000$ trajectories of 300 time steps for both FluidPour and FluidShake as an offline dataset to train the model, 800 trajectories of 80 time steps for RigidStack, and $1,000$ trajectories of 50 time steps for RigidDrop.

Evaluation metrics. We use the first two environments, i.e., FluidPour and FluidShake, to measure the control performance, where we specify the target configuration of the control task using images from (1) one of the viewpoints encountered during training, (2) an interpolated viewpoint between training viewpoints, and (3) an extrapolated viewpoint outside the training distribution (Figure 4-4).

We provide quantitative evaluations on the control performance in FluidPour and FluidShake by extracting the particle set from the simulator, and measuring the Chamfer distance between the result and the goal, which we denote as "Chamfer Dist.". In FluidPour, we provide additional measurements on the L2 distance of the position/orientation of the cup towards the goal, denoting as "Position Error" and "Angle Error" respectively. In FluidShake, we calculate the L2 distance of the container and cube's position towards the goal and denote

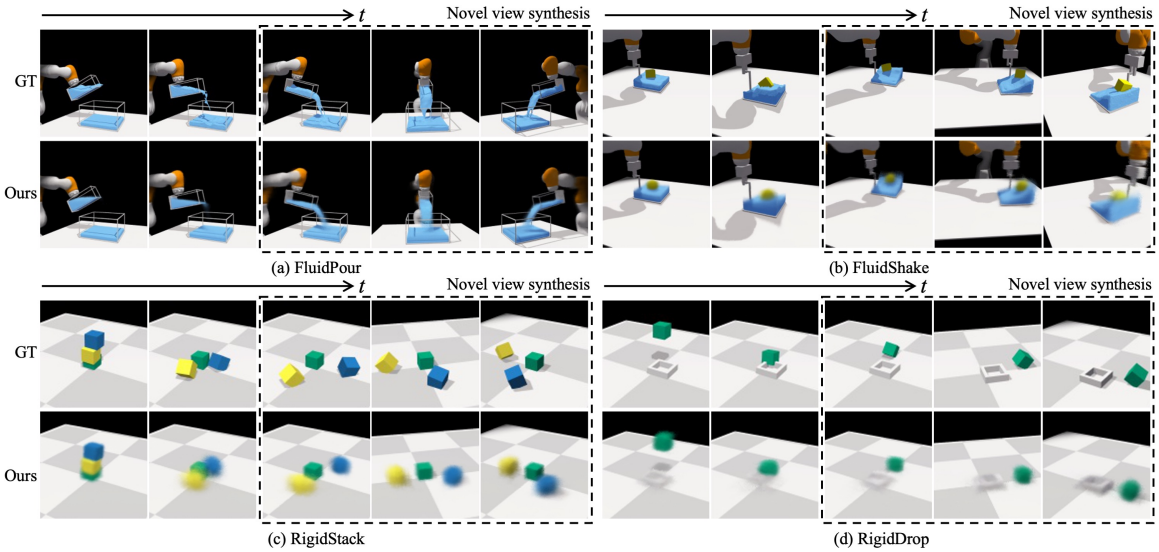


Figure 4-7: **Forward prediction and novel view synthesis on four environments.** Given a scene representation and an input action sequence, our dynamics model predicts the subsequent latent scene representation, which is used as the input of our decoder model to reconstruct the corresponding visual observation based on different viewpoints. In each block, we render images based on the open-loop future dynamic prediction from left to right. Images in the dotted box compare our model's novel view synthesis results in the last time step with the ground truth from three different viewpoints.

them as "Container Error" and "Cube Error" respectively.

4.5.1 Baseline Methods

For comparison, we consider the following three baselines: **TC**: Similar to TCN [Sermanet et al., 2018], it only uses time contrastive loss for learning the image feature without having to reconstruct the scene. We learn a dynamics model directly on the image features for control. **TC+AE**: Instead of using Neural Radiance Fields to reconstruct the image, this method uses the standard convolutional decoder to reconstruct the target image when given a new viewpoint. This would then be similar to GQN [Eslami et al., 2018] augmented with a time contrastive loss. **NeRF**: This method is a direct adaptation from the original NeRF paper [Mildenhall et al., 2020] and is the same as ours except that it does not include the time contrastive loss during training and the auto-decoding test-time optimization. We use the same dynamics model shown in Figure 4-2b and train the model for each baseline respectively for dynamic prediction. We use the same feedback control method, i.e., MPPI [Williams et al., 2015a], for our model and all the baselines.

4.5.2 Control Results

Goal specification from novel viewpoints. Figure 4-4c shows the goal configuration, and we ask the learned model to perform three control trials where the goal is specified from different types of viewpoints. The left three columns show the MPC control process from the agent's viewpoint. The fourth column visualizes the final configuration the agent achieves from the same viewpoint as the goal image. Trial #1 specifies the goal using a different viewpoint from the agent's but has been encountered during training. Trial #2 uses a goal view that is an interpolation of training viewpoints. Our agent can achieve the target configuration with decent accuracy. For trial #3, we specify the goal view by moving the camera closer, higher, and more downwards with respect to the container. Note this goal image view is outside the distribution of training viewpoints. With the help of test-time auto-decoding optimization introduced in Section 4.4.2, our method can successfully achieve the target configuration, as shown in the figure.

Baseline comparisons. We benchmark our model with the baselines by assessing their performance on the downstream control tasks. Figure 4-5 shows the qualitative comparison between our model (Ours), a variant of our model that does not perform the auto-decoding test-time optimization (Ours w/o AD), and the best-performing baseline (TC+AE) introduced in Section 4.5.1. We find that when the target view is outside the training distribution and vastly different from the agent view, our full method shows a much better performance in achieving the target configuration. The variant without auto-decoding optimization and TC+AE fail to accomplish the task and exhibit an apparent deviation from the ground truth in the 3D points space of the fluids and floating cube. We also provide quantitative comparisons on the control results. Figure 4-6 shows the performance in the FluidPour and FluidShake environments. We find our full model significantly outperforms the baseline approaches in both environments under all scenarios and evaluation metrics. The results effectively demonstrate the advantages of the learned 3D-aware scene representations, which contain a more precise encoding of the contents in the 3D environments and are capable of extrapolation viewpoint generalization.

4.5.3 Dynamic Prediction and Novel View Synthesis

Conditioned on a scene representation and an input action sequence, our dynamics model f_{dyn} can iteratively predict the evolution of the scene representations. Our decoder can then take the predicted state representation and reconstruct the corresponding visual observation from a query viewpoint. Figure 4-7 shows that our model can accurately predict the future and perform novel view synthesis on four environments involving both fluid and rigid objects, suggesting its usefulness in trajectory optimization. Please check our video results in the supplementary material for better visualization.

4.6 Discussion

In this chapter, we proposed to learn viewpoint-invariant 3D-aware scene representations from visual observations using an autoencoding framework augmented with a neural radiance field rendering module and time contrastive learning. We show that the learned 3D representations perform well on the model-based visuomotor control tasks. When coupled with an auto-decoding test-time optimization mechanism, our method allows goal specification from a viewpoint outside the training distribution. We demonstrate the applicability of the proposed framework in a range of complicated physics environments involving rigid objects and fluids, which we hope can facilitate future works on visuomotor control for complex and dynamic 3D manipulation tasks.

Chapter 5

Learning Compositional Linear Dynamics with Koopman Operators

Previous chapters consider dynamics models represented using neural networks, where the dynamics transition goes through highly nonlinear computations. Despite their strong generalization power, they are not as efficient in system identification and control, because deep nets are heavily over-parameterized, making optimization time-consuming and sample-inefficient. On the other hand, finding an embedding space for a linear approximation of a nonlinear dynamical system will enable much more efficient system identification and control synthesis. The Koopman operator theory lays the foundation for identifying the nonlinear-to-linear coordinate transformations with data-driven methods. Recently, researchers have proposed to use deep neural networks as a more expressive class of basis functions for calculating the Koopman operators. These approaches, however, assume a fixed dimensional state space; they are therefore not applicable to scenarios with a variable number of objects.

In this chapter, we continue the momentum of incorporating structures into the learning systems and propose to learn compositional Koopman operators, using graph neural networks to encode the state into object-centric embeddings and using a block-wise linear transition matrix to regularize the shared structure across objects. The learned dynamics can quickly adapt to new environments of unknown physical parameters and produce control signals to achieve a specified goal. Our experiments on manipulating ropes and controlling soft robots show that the proposed method has better efficiency and generalization ability than existing baselines.

This chapter contains materials previously published by Li et al. [2019a] in the International Conference on Learning Representations (ICLR) 2020 as **Spotlight Presentation**, in which Hao He has also made significant contributions along with co-authors Jiajun Wu, Dina Katabi, and Antonio Torralba.

5.1 Introduction

Simulating and controlling complex dynamical systems, such as ropes or soft robots, relies on two key features of the dynamics model: first, it needs to be *efficient* for system identification and motor control; second, it needs to be *generalizable* to a complex, constantly evolving environments.

In practice, computational models for complex, nonlinear dynamical systems are often not efficient enough for real-time control [Mayne, 2000]. The Koopman operator theory suggests that identifying nonlinear-to-linear coordinate transformations allows efficient linear approximation of nonlinear systems [Williams et al., 2015b, Mauroy and Goncalves, 2016]. Fast as they are, however, existing papers on Koopman operators focus on a single dynamical system, making it hard to generalize to cases where there are a variable number of components.

In this chapter, we propose compositional Koopman operators, integrating Koopman operators with graph networks discussed in detail in Chapter 3 for generalizable and efficient dynamics modeling. We build on the idea of encoding states into object-centric embeddings with graph neural networks, which ensures generalization power. But instead of using over-parameterized neural nets to model state transition, we identify the Koopman matrix and control matrix from data as a linear approximation of the nonlinear dynamical system. The linear approximation allows efficient system identification and control synthesis.

The main challenge of extending Koopman theory to multi-object systems is scalability. The number of parameters in the Koopman matrix scales quadratically with the number of objects, which harms the learning efficiency and leads to overfitting. To tackle this issue, we exploit the structure of the underlying system and use the same block-wise Koopman sub-matrix for object pairs of the same relation. This significantly reduces the number of parameters that need to be identified by making it independent of the size of the system.

Our experiments include simulating and controlling ropes of variable lengths and soft robots of different shapes. The compositional Koopman operators are significantly more

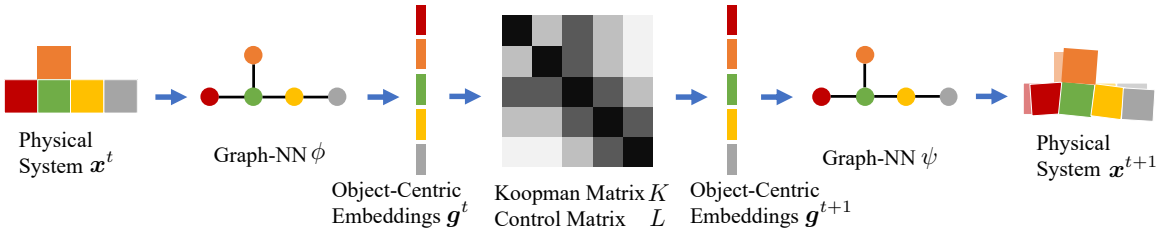


Figure 5-1: **Overview of compositional Koopman operators.** A graph neural network ϕ takes in the current state of the physical system $\boldsymbol{x}^t$, and generates object-centric representations in the Koopman space $\boldsymbol{g}^t$. We then use the block-wise Koopman matrix K and control matrix L identified from Equation 5.6 or Equation 5.8 to predict the Koopman embeddings in the next time step $\boldsymbol{g}^{t+1}$. Note that in K and L, object pairs of the same relation share the same sub-matrix. Another graph neural network ψ maps $\boldsymbol{g}^{t+1}$ back to the original state space, i.e., $\boldsymbol{x}^{t+1}$. The mapping between $\boldsymbol{g}^t$ and $\boldsymbol{g}^{t+1}$ is linear and is shared across all time steps, where we can iteratively apply K and L to the Koopman embeddings and roll multiple steps into the future. The formulation enables efficient system identification and control synthesis.

accurate than the state-of-the-art learned physics engines [Battaglia et al., 2016, Li et al., 2019b], and faster when adapting to new environments of unknown physical parameters. Our method also outperforms vanilla deep Koopman methods [Lusch et al., 2018, Morton et al., 2018] and Koopman models with manually-designed basis functions, which shows the advantages of using a structured Koopman matrix and graph neural networks. Please see our project page for demonstrating videos: `http://koopman.csail.mit.edu/`.

5.2 Related Work

Koopman operators. The Koopman operator formalism of dynamical systems is rooted in the seminal works of Koopman and Von Neumann in the early 1930s [Koopman, 1931, Koopman and Neumann, 1932]. The core idea is to map the state of a nonlinear dynamical system to an embedding space, over which we can linearly propagate into the future. Researchers have proposed various algorithms to explore the Koopman spectral properties from data. A large portion of them are in the class of dynamic mode decomposition (DMD) [Rowley et al., 2009, Schmid, 2010, Tu et al., 2014, Williams et al., 2015b, Arbabi and Mezic, 2017]. The linear representation will enable efficient prediction, estimation, and control using tools from linear dynamical systems [Williams et al., 2016, Proctor et al., 2018, Mauroy and Goncalves, 2019, Korda and Mezić, 2018]. People have been using hand-designed Koopman observables for various modeling and control tasks [Brunton et al., 2016, Kaiser et al., 2017, Abraham et al., 2017, Bruder et al., 2019b, Arbabi et al., 2018]. Some recent

works have applied the method to the real world and successfully control soft robots with great precision [Bruder et al., 2019a, Mamakoukas et al., 2019].

However, hand-crafted basis functions sometimes fail to generalize to more complex environments. Learning these functions from data using neural nets turns out to generate a more expressive invariant subspace [Lusch et al., 2018, Takeishi et al., 2017] and has achieved successes in fluid control [Morton et al., 2018]. Morton et al. [2019] has also extended the framework to account for uncertainty in the system by inferring a distribution over observations. Our model differs by explicitly modeling the compositionality of the underlying system with graph networks. It generalizes better to environments of a variable number of objects or soft robots of different shapes.

Learning-based physical simulators. Battaglia et al. [2016] and Chang et al. [2017] first explored learning a simulator from data by approximating object interactions with neural networks. These models are no longer bounded to hard-coded physical rules, and can adapt to scenarios where the underlying physics is unknown. Please refer to Battaglia et al. [2018] for a full review. As discussed in detail in Chapter 3, Mrowca et al. [2018] and Li et al. [2019a] extended these models to approximate particle dynamics of deformable shapes and fluids. Flexible as they are, these models become less efficient during model adaptation in complex scenarios, because the optimization of neural networks usually needs a lot of samples and compute, which limits its use in an online setting. Nagabandi et al. [2019a,b] proposed to use meta-learning for online adaptation, and have shown to be effective in simulated robots and a real legged millirobot. However, it is not clear whether their methods can generalize to systems with variable numbers of instances. The use of graph nets and Koopman operators in our model allows better generalization ability and enables efficient system identification as we only need to identify the transition matrices, which is essentially a least-square problem and can be solved very efficiently.

The previous chapters have shown examples of using the learned physics engines, together with model-predictive control (MPC), for various manipulation tasks. Some more examples including Sanchez-Gonzalez et al. [2018], Li et al. [2019b], and Janner et al. [2019]. Many previous papers in this direction also jointly learn a latent dynamics model and a policy in a model-based reinforcement learning setup [Racanière et al., 2017, Hamrick et al., 2017, Pascanu et al., 2017, Hafner et al., 2019b]. In this chapter, we leverage the fact that the

embeddings in the Koopman space are propagating linearly through time, which allows us to formulate the control problem as quadratic programming and optimize the control signals much more efficiently.

5.3 Approach

We first present the basics of Koopman operators: for a nonlinear dynamical system, the Koopman observation functions can map the state space to an embedding space where the dynamics become linear. We then discuss the compositional nature of physical systems and show how graph networks can be used to capture the compositionality.

5.3.1 The Koopman Operators

Let $\boldsymbol{x}^t \in \mathcal{X} \subset \mathbb{R}^n$ be the state vector for the system at time step t. We consider a non-linear discrete-time dynamical system described by $\boldsymbol{x}^{t+1} = F(\boldsymbol{x}^t)$. The Koopman operator [Koopman, 1931], denoted as $\mathcal{K} : \mathcal{F} \to \mathcal{F}$, is a linear transformation defined by $\mathcal{K}g \triangleq g \circ F$, where $\mathcal{F}$ is the collection of all functions (also referred to as *observables*) that form an infinite-dimensional Hilbert space. For every function $g : \mathcal{X} \to \mathbb{R}$ belonging to $\mathcal{F}$, we have

$$(\mathcal{K}g)(\boldsymbol{x}^t) = g(F(\boldsymbol{x}^t)) = g(\boldsymbol{x}^{t+1}), \tag{5.1}$$

making the function space $\mathcal{F}$ invariant under the action of the Koopman operator.

Although the theory guarantees the existence of the Koopman operator, its use in practice is limited by its infinite dimensionality. Most often, we assume there is an invariant subspace $\mathcal{G}$ of the Koopman operator. It spans by a set of base observation functions $\{g_1, \cdots, g_m\}$ and satisfies that $\mathcal{K}g \in \mathcal{G}$ for any $g \in \mathcal{G}$. With a slightly abuse of the notation, we now use $g(\boldsymbol{x}^t) : \mathbb{R}^n \to \mathbb{R}^m$ to represent $[g_1(\boldsymbol{x}^t), \cdots, g_m(\boldsymbol{x}^t)]^T$. By constraining the Koopman operator on this invariant subspace, we get a finite-dimensional linear operator $K \in \mathbb{R}^{m \times m}$ that we refer as the *Koopman matrix*.

Traditionally, people hand-craft base observation functions from the knowledge of underlying physics. The system identification problem is then reduced to finding the Koopman matrix K, which can be solved by linear regression given historical data of the system. Recently, researchers have also explored data-driven methods that automatically find the

Koopman invariant subspace via representing the base observation functions $g(\boldsymbol{x})$ via deep neural networks.

Although the original Koopman theory does not consider the system with external control inputs, researchers have found that linearly injecting control signals to the Koopman observation space can give us good numerical performance [Brunton et al., 2016, Bruder et al., 2019a]. Mathematically, considering a dynamical system, $\boldsymbol{x}^{t+1} = F(\boldsymbol{x}^t, \boldsymbol{u}^t)$, with an external control input $\boldsymbol{u}^t$, we aim to find the Koopman observation functions and the linear dynamics model in the form of

$$g(\boldsymbol{x}^{t+1}) = Kg(\boldsymbol{x}^t) + L\boldsymbol{u}^t, \tag{5.2}$$

where the coefficient matrix L is referred to as the *control matrix*.

5.3.2 Compositional Koopman Operators

The dynamics of a physical system are governed by physical rules, which are usually shared across different subcomponents in the system. Explicitly modeling such compositionality enables more efficient system identification and control synthesis and provides better generalization ability.

Motivating example. Consider a system with N balls moving in a 2D plane, each pair connected by a linear spring. Assume all balls have mass 1 and all springs share the same stiffness coefficient k. We denote the i's ball's position as (x_i, y_i) and its velocity as $(\dot{x}_i, \dot{y}_i)$. For ball i, Equation 5.3 describes its dynamics, where $\boldsymbol{x}_i \triangleq [x_i, y_i, \dot{x}_i, \dot{y}_i]^T$ denotes ball i's state:

$$\dot{\boldsymbol{x}}_i = \begin{bmatrix} \dot{x}_i \\ \dot{y}_i \\ \ddot{x}_i \\ \ddot{y}_i \end{bmatrix} = \begin{bmatrix} \dot{x}_i \\ \dot{y}_i \\ \sum_{j=1}^N k(x_j - x_i) \\ \sum_{j=1}^N k(y_j - y_i) \end{bmatrix} = \underbrace{\begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ k - Nk & 0 & 0 & 0 \\ 0 & k - Nk & 0 & 0 \end{bmatrix}}_{\triangleq A} \begin{bmatrix} x_i \\ y_i \\ \dot{x}_i \\ \dot{y}_i \end{bmatrix} + \sum_{j \neq i} \underbrace{\begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ k & 0 & 0 & 0 \\ 0 & k & 0 & 0 \end{bmatrix}}_{\triangleq B} \begin{bmatrix} x_j \\ y_j \\ \dot{x}_j \\ \dot{y}_j \end{bmatrix}. \tag{5.3}$$

We can represent the state of the whole system using the union of every ball's state, where $\boldsymbol{x} = [\boldsymbol{x}_1, \cdots, \boldsymbol{x}_N]^T$. Then the transition matrix is essentially a block matrix, where the matrix

parameters are shared among the diagonal or off-diagonal blocks as shown in Equation 5.4:

$$\dot{\boldsymbol{x}} = \begin{bmatrix} \dot{\boldsymbol{x}}_1 \\ \dot{\boldsymbol{x}}_2 \\ \vdots \\ \dot{\boldsymbol{x}}_N \end{bmatrix} = \begin{bmatrix} A & B & \cdots & B \\ B & A & \cdots & B \\ \vdots & \vdots & \ddots & \vdots \\ B & B & \cdots & A \end{bmatrix} \begin{bmatrix} \boldsymbol{x}_1 \\ \boldsymbol{x}_2 \\ \vdots \\ \boldsymbol{x}_N \end{bmatrix}. \tag{5.4}$$

Based on the linear spring system, we make three observations for multi-object systems.

- **The system state is composed of the state of each individual object.** The dimension of the whole system scales linearly with the number of objects. We formulate the system state by concatenating the state of every object, corresponding to an object-centric state representation.
- **The transition matrix has a block-wise substructure.** After assuming an object-centric state representation, the transition matrix naturally has a block-wise structure as shown in Equation 5.4.
- **The same physical interactions share the same transition block.** The blocks in the transition matrix encode actual interactions and generalize across systems. A and B govern the dynamics of the linear spring system, and are shared by systems with a different number of objects.

These observations inspire us to exploit the structure of multi-object systems, instead of learning separate models for systems that contains different numbers of balls.

Compositional Koopman operators. Motivated by the linear spring system, we want to inject a good inductive bias to incorporate compositionality when applying the Koopman theory. This allows better generalization ability and more efficient system identification and better controller design. Figure 5-1 shows an overview of our model.

Considering a system with N objects, we denote $\boldsymbol{x}^t$ as the system state at time t and $\boldsymbol{x}_i^t$ is the state of the i'th object. We further denote $\boldsymbol{g}^t \triangleq g(\boldsymbol{x}^t)$ as the embedding of the state in the Koopman invariant space. In the rest of the paper, we call $\boldsymbol{g}^t$ the *Koopman embedding*. Based on the observation we made in the case of linear spring system, we propose the following assumptions on the compositional structure of the Koopman embedding and the Koopman matrix.

- **The Koopman embedding of the system is composed of the Koopman embedding of every objects.** Similar to the decomposition in the state space, we assume the Koopman embedding can be divided into object-centric sub-embeddings, i.e. $\boldsymbol{g}^t \in \mathbb{R}^{Nm}$ denoting the concatenation of $\boldsymbol{g}_1^t, \cdots, \boldsymbol{g}_N^t$, where we use $\boldsymbol{g}_i^t = g_i(\boldsymbol{x}^t) \in \mathbb{R}^m$ as the Koopman embedding for the i'th object.

- **The Koopman matrix has a block-wise structure.** It is natural to think the Koopman matrix is composed of block matrices after assuming an object-centric Koopman embeddings. In Equation 5.5, $K_{ij} \in \mathbb{R}^{m\times m}$ and $L_{ij} \in \mathbb{R}^{m\times l}$ are blocks of the Koopman matrix and the control matrix, where l is the dimension of the action for each object and $\boldsymbol{u}^t \in \mathbb{R}^{Nl}$ is the concatenation of $\boldsymbol{u}_1^t, \cdots, \boldsymbol{u}_N^t$ denoting the total control signal at time t:

$$\begin{bmatrix} \boldsymbol{g}_1^{t+1} \\ \vdots \\ \boldsymbol{g}_N^{t+1} \end{bmatrix} = \begin{bmatrix} K_{11} & \cdots & K_{1N} \\ \vdots & \ddots & \vdots \\ K_{N1} & \cdots & K_{NN} \end{bmatrix} \begin{bmatrix} \boldsymbol{g}_1^{t} \\ \vdots \\ \boldsymbol{g}_N^{t} \end{bmatrix} + \begin{bmatrix} L_{11} & \cdots & L_{1N} \\ \vdots & \ddots & \vdots \\ L_{N1} & \cdots & L_{NN} \end{bmatrix} \begin{bmatrix} \boldsymbol{u}_1^{t} \\ \vdots \\ \boldsymbol{u}_N^{t} \end{bmatrix}. \tag{5.5}$$

  As we have seen in the case of linear spring system, those matrix blocks are not independent, but some of them share the same set of values.

- **The same physical interactions shall share the same transition block.** The equivalence between the blocks should reflect the equivalence of the interactions, where we use the same transition sub-matrix for object pairs of the same relation. For example, if the system is composed of N identical objects interacting with the same relation, then, by symmetry, all the diagonal blocks should be the same, while all the off-diagonal blocks should also be the same. The repetitive structure allows us to efficiently identify the values using least squares regression.

5.3.3 Learning the Koopman Embeddings Using Graph Neural Networks

For a physical system that contains N objects, we represent the system at time t using a directed graph $G^t = (O^t, R)$, where vertices $O^t = \{\boldsymbol{o}_i^t\}_{i=1}^N$ represent objects and edges $R = \{\boldsymbol{r}_k\}_{k=1}^{N^2}$ represent pair-wise relations. Specifically, $\boldsymbol{o}_i^t = (\boldsymbol{x}_i^t, \boldsymbol{a}_i^o)$, where $\boldsymbol{x}_i^t$ is the state of object i and $\boldsymbol{a}_i^o$ is a one-hot vector indicating the object type, e.g., fixed or movable. For relation, we have $\boldsymbol{r}_k = (u_k, v_k, \boldsymbol{a}_k^r), 1 \le u_k, v_k \le N$, where u_k and v_k are integers denoting the end points of this directed edge, and $\boldsymbol{a}_k^r$ is a one-hot vector denoting the type of the

relation k.

We use a graph neural network similar to Interaction Networks (IN) [Battaglia et al., 2016] to generate object-centric Koopman embeddings. IN defines an object function f_O and a relation function f_R to model objects and their relations in a compositional way. Similar to a message passing procedure, we calculate the edge effect $\boldsymbol{e}_k^t = f_R(\boldsymbol{o}_{u_k}^t, \boldsymbol{o}_{v_k}^t, \boldsymbol{a}_k^r)_{k=1\ldots N^2}$, and node effect $\boldsymbol{g}_i^t = f_O(\boldsymbol{o}_i^t, \sum_{k\in\mathcal{N}_i} \boldsymbol{e}_k^t)_{i=1\ldots N}$, where $\mathcal{N}_i$ denotes the relations that point to the object i and $\{\boldsymbol{g}_i^t\}$ are the derived Koopman embeddings. We use this graph neural network, denoted as ϕ, to represent our Koopman observation function.

System identification. For a sequence of observations $\widetilde{\boldsymbol{x}} = [\boldsymbol{x}^1, \cdots, \boldsymbol{x}^T]$ from time 1 to time T, we first map them to the Koopman space as $\widetilde{\boldsymbol{g}} = [\boldsymbol{g}^1, \cdots, \boldsymbol{g}^T]$ using the graph encoder ϕ, where $\boldsymbol{g}^t = \phi(\boldsymbol{x}^t)$. We use $\boldsymbol{g}^{i:j}$ to denote the sub-sequence $[\boldsymbol{g}^i, \cdots, \boldsymbol{g}^j]$. To identify the Koopman matrix, we solve the linear regression $\min_K \|K\boldsymbol{g}^{1:T-1} - \boldsymbol{g}^{2:T}\|_2$. As a result, $K = \boldsymbol{g}^{2:T}(\boldsymbol{g}^{1:T-1})^\dagger$ will asymptotically approach the Koopman operator $\mathcal{K}$ with an increasing T. For cases where there are control inputs $\widetilde{\boldsymbol{u}} = [\boldsymbol{u}^1, \cdots, \boldsymbol{u}^{T-1}]$, the calculation of the Koopman matrix and the control matrix is essentially solving a least squares problem w.r.t. the objective

$$\min_{K,L} \|K\boldsymbol{g}^{1:T-1} + L\widetilde{\boldsymbol{u}} - \boldsymbol{g}^{2:T}\|_2. \tag{5.6}$$

As we mentioned in the Section 5.3.2, the dimension of the Koopman space is linear to the number of objects in the system, i.e., $\widetilde{\boldsymbol{g}} \in \mathbb{R}^{Nm\times T}$ and $K \in \mathbb{R}^{Nm\times Nm}$. If we do not enforce any structure on the Koopman matrix K, we will have to identify N^2m^2 parameters. Instead, we can significantly reduce the number by leveraging the assumption on the structure of K. Assume we know some blocks ($\{K_{ij}\}$) of the matrix K are shared and in total there are h different kinds of blocks, which we denote as $\hat{K} \in \mathbb{R}^{h\times m\times m}$. Then, the number of parameter to be identified reduce to hm^2. Usually, h does not depend on N, and is much smaller than N^2. Now, for each block K_{ij}, we have a one-hot vector $\sigma_{ij} \in \{0,1\}^h$ indicating its type, i.e., $K_{ij} = \sigma_{ij}\hat{K} \in \mathbb{R}^{m\times m}$. Finally, as shown in Equation 5.7, we represent the Koopman matrix as the product of the index tensor σ and the parameter tensor $\hat{K}$:

$$K = \sigma \otimes \hat{K} = \begin{bmatrix} \sigma_{11}\hat{K} & \cdots & \sigma_{1N}\hat{K} \\ \vdots & \ddots & \vdots \\ \sigma_{N1}\hat{K} & \cdots & \sigma_{NN}\hat{K} \end{bmatrix}, \text{where} \quad \sigma = \begin{bmatrix} \sigma_{11} & \cdots & \sigma_{1N} \\ \vdots & \ddots & \vdots \\ \sigma_{N1} & \cdots & \sigma_{NN} \end{bmatrix} \in \mathbb{R}^{N\times N\times h}. \tag{5.7}$$

Similar to the Koopman matrix, we assume the same block structure in the control matrix L and denote its parameter as $\hat{L} \in \mathbb{R}^{h \times m \times l}$. The least squares problem of identifying $\hat{K}$ and $\hat{L}$ becomes

$$\min_{\hat{K},\hat{L}} \|(\sigma \otimes \hat{K})\boldsymbol{g}^{1:T-1} + (\sigma \otimes \hat{L})\widetilde{\boldsymbol{u}} - \boldsymbol{g}^{2:T}\|_2, \tag{5.8}$$

where $\sigma \otimes \hat{K} \in \mathbb{R}^{Nm \times Nm}$, $\sigma \otimes \hat{L} \in \mathbb{R}^{Nm \times Nl}$, $\boldsymbol{g}^{1:T-1} \in \mathbb{R}^{Nm \times (T-1)}$ and $\widetilde{\boldsymbol{u}} \in \mathbb{R}^{Nl \times (T-1)}$. Since the linear least squares problems described in Equation 5.6 and Equation 5.8 have analytical solutions, performing system identification using our method is very efficient.

Training GNN models. To make predictions on the states, we use a graph decoder ψ to map the Koopman embeddings back to the original state space. In total, we have three losses to train the graph encoder and decoder. The first term is the auto-encoding loss

$$\mathcal{L}_{\text{ae}} = \frac{1}{T}\sum_{i}^{T} \|\psi(\phi(\boldsymbol{x}^i)) - \boldsymbol{x}^i\|. \tag{5.9}$$

The second term is the prediction loss. To calculate it, we rollout in the Koopman space and denote the embeddings as $\hat{\boldsymbol{g}}^1 = \boldsymbol{g}^1$, and $\hat{\boldsymbol{g}}^{t+1} = K\hat{\boldsymbol{g}}^t + L\boldsymbol{u}^t$, for $t = 1, \cdots, T-1$. The prediction loss is defined as the difference between the decoded states and the actual states, i.e.,

$$\mathcal{L}_{\text{pred}} = \frac{1}{T}\sum_{i=1}^{T} \|\psi(\hat{\boldsymbol{g}}^i) - \boldsymbol{x}^i\|. \tag{5.10}$$

Third, we employ a metric loss to encourage the Koopman embeddings preserving the distance in the original state space. The loss is defined as the absolute error between the distances measured in the Koopman space and that in the original space, i.e.,

$$\mathcal{L}_{\text{metric}} = \sum_{ij} \left| \|\boldsymbol{g}^i - \boldsymbol{g}^j\| - \|\boldsymbol{x}^i - \boldsymbol{x}^j\| \right|. \tag{5.11}$$

Having Koopman embeddings that perserves the distance in the state space is important as we are using the distance in the Koopman space to define the cost function for downstream control tasks.

The final training loss is simply the combination of all the terms above: $\mathcal{L} = \mathcal{L}_{\text{ae}} + \lambda_1\mathcal{L}_{\text{pred}} + \lambda_2\mathcal{L}_{\text{metric}}$. We then minimize the loss $\mathcal{L}$ by optimizing the parameters in the graph

encoder ϕ and graph decoder ψ using stochastic gradient descent. Once the model is trained, it can be used for system identification, future prediction, and control synthesis.

5.3.4 Control

For a control task, the goal is to synthesize a sequence of control inputs $\boldsymbol{u}^{1:T}$ that minimize $C = \sum_{t=1}^{T} c_t(\boldsymbol{x}^t, \boldsymbol{u}^t)$, the total incurred cost, where $c_t(\boldsymbol{x}^t, \boldsymbol{u}^t)$ is the instantaneous cost. For example, considering the control task of reaching a desired state $\boldsymbol{x}^*$ at time T, we can design the following instantaneous cost, $c_t(\boldsymbol{x}^t, \boldsymbol{u}^t) = \mathbb{1}_{[t=T]}\|\boldsymbol{x}^t - \boldsymbol{x}^*\|_2^2 + \lambda\|\boldsymbol{u}^t\|_2^2$. The first term promotes the control sequence that matches the state to the goal, while the second term regularizes the control signals.

Open-loop control via quadratic programming (QP). Our model maps the original nonlinear dynamics to a linear dynamical system. We can then solve the control task by solving a linear control problem. With the assumption that the Koopman embeddings preserve the distance measure, we define the control cost as $c_t(\boldsymbol{g}^t, \boldsymbol{u}^t) = \mathbb{1}_{[t=T]}\|\boldsymbol{g}^t - \boldsymbol{g}^*\|_2^2 + \lambda\|\boldsymbol{u}^t\|_2^2$. As a result, we reduce the problem to minimizing a quadratic cost function $C = \sum_{t=1}^{T} c_t(\boldsymbol{g}^t, \boldsymbol{u}^t)$ over variables $\{\boldsymbol{g}^t, \boldsymbol{u}^t\}_{t=1}^{T}$ under linear constrains $\boldsymbol{g}^{t+1} = K\boldsymbol{g}^t + L\boldsymbol{u}^t$, where $\boldsymbol{g}^1 = \phi(\boldsymbol{x}^1)$ and $\boldsymbol{g}^* = \phi(\boldsymbol{x}^*)$.

Model predictive control (MPC). Solving the QP gives us control signals, which might not be good enough for long-term control as the prediction error accumulates. We can combine it with Model Predictive Control, assuming feedback from the environment every τ steps.

5.4 Experiments

Environments. We evaluate our method by assessing how well it can simulate and control ropes and soft robots. Specifically, we consider three environments. (1) **Rope** (Figure 5-2a): the top mass of a rope is fixed to a specific height. We apply force to the top mass to move it in a horizontal line. The rest of the masses are free to move according to internal force and gravity. (2) **Soft** (Figure 5-2b): we aim to control a soft robot that is consist of soft blocks. Blocks in dark grey are rigid and those in light blue are soft blocks. Each one of the dark blue blocks is soft but have an actuator inside that can contract or expand the block. One of

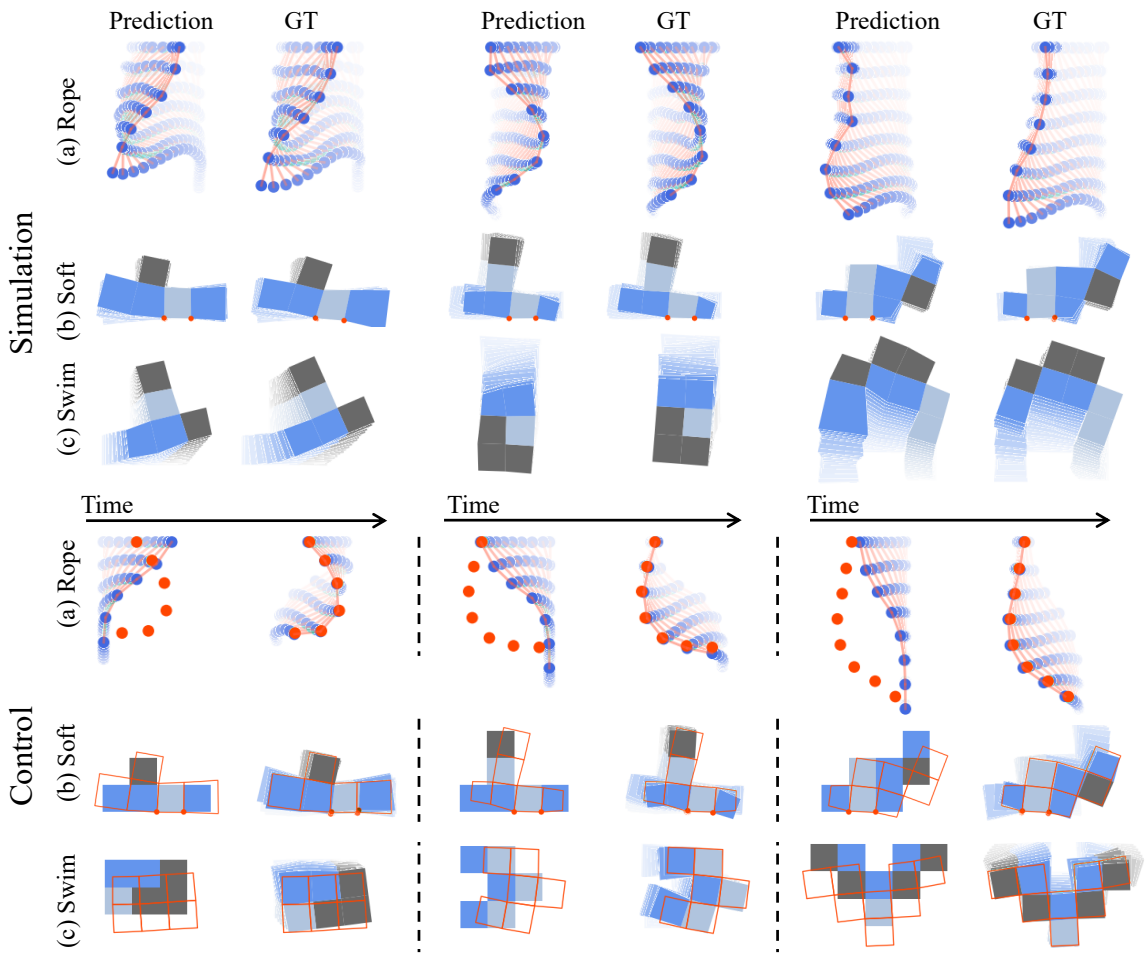


Figure 5-2: **Qualitative results.** Top: our model prediction matches the ground truth over a long period. Bottom: for control, we use red dots or frames to indicate the goal. We apply the control signals generated from our identified model to the original simulator, which allows the agent to achieve the goal accurately. Please refer to our supplementary video in the website for more results.

the blocks is pinned to the ground, as shown using the red dots. (3) **Swim** (Figure 5-2c): instead of pinning the soft robot to the ground, we let the robot swim in fluids. The colors shown in this environment have the same meaning as in **Soft**.

Observation space. In the **Rope** environment, each mass on the rope is considered as an object. The observation of each mass is its position and velocity in the 2D plane, which has a dimension of 4. In total, a rope with N masses has an observation space of dimension $4N$. In both the **Soft** and the **Swim** environments, each quadrilateral is considered as an object. For each quadrilateral, we have access to the positions and velocities of the four corners. Thus for a soft robot containing N quadrilaterals, we have a $4 \times 4 \times N = 16N$ dimensional observation.

Baselines. We compare our model to the following baselines: Interaction Networks [Battaglia et al., 2016] (**IN**), Propagation Networks [Li et al., 2019b] (**PN**), and Koopman method with hand-crafted Koopman base functions (**KPM**). IN and PN are the state-of-the-art learning-based physical simulators, and we evaluate their adaptation ability by finetuning their parameters on a small sequence of observations from the testing environment. Similar to our method, KPM fits a linear dynamics in the Koopman space. Instead of learning Koopman observations from data, KPM uses polynomials of the original states as the basis functions. In our setting, we set the maximum order of the polynomials to be three to make the dimension of the hand-crafted Koopman embeddings match our model's.

Data generation. We generate 10,000 episodes for Rope and 50,000 episodes for Soft and Swim. Among them, 90% are used for training, and the rest for testing. Each episode has 100 time steps. In the dataset, the physical systems have a various number of objects from 5 to 9, i.e., the ropes have 5 to 9 masses while the soft robots in Soft and Swim environments have 5 to 9 quadrilaterals. To evaluate the model's extrapolating generalization ability, for each environment, we generate an extra dataset with the same size as the test set while containing systems consist of 10 to 14 objects.

Training and evaluation protocols. All models are trained using Adam optimizer [Kingma and Ba, 2015] with a learning rate of 10^{-4} and a batch size of 8. λ_1 and λ_2 are 1.0 and 0.3, respectively, for our model. For both our model and the baselines, we apply 400K iterations of gradient steps in the **Rope** environment and 580K iterations in the **Soft** and **Swim** environment. Our model is trained on the sub-sequence of length 64 from the training set, and IN/PN aims at minimizing the L1 distance between their prediction and the ground truth. During test time, the models have to adapt to a new environment of unknown physical parameters, where they have access to a short sequence of observations and the opportunity to adjust their models' parameters. Our model uses 8 episodes to identify the transition matrix via least-square regression. IN/PN update the model's parameters by minimizing the distance between the model's prediction and the actual observation using a gradient step of length 10^{-4} for 5 iterations. For evaluation, we use two metrics: **simulation error** and **control error**. For a given episode, the simulation error at time step t is defined as the mean squared error between the model prediction $\hat{\boldsymbol{x}}^t$ and the ground truth $\boldsymbol{x}^t$. For control,

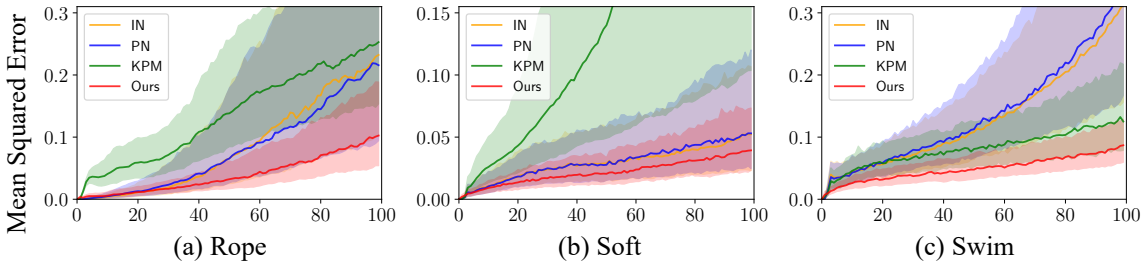


Figure 5-3: **Quantitative results on simulation.** The x axis shows time steps. The solid lines indicate medians and the transparent regions are the interquartile ranges of simulation errors. Our method significantly outperforms the baselines in all testing environments.

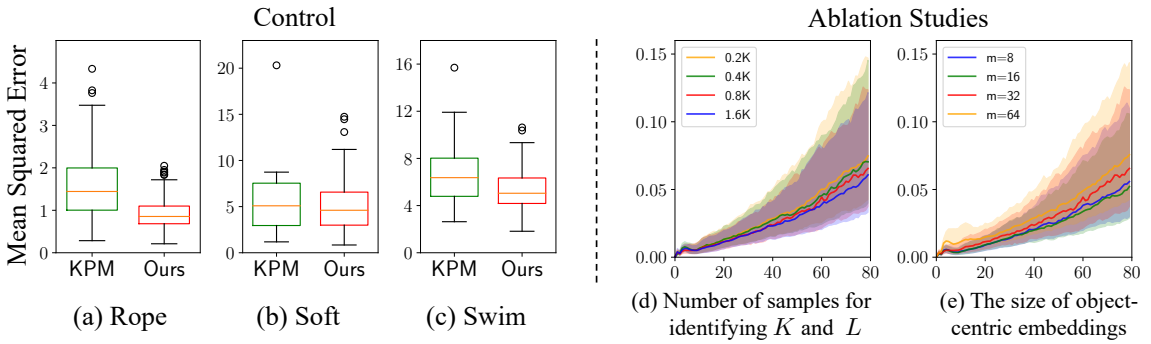


Figure 5-4: **Quantitative results on control and ablation studies on model hyperparameters.** Left: box-plots show the distributions of control errors. The yellow line in the box indicates the median. Our model consistently achieves smaller errors in all environments against KPM. Right: our model's simulation errors with different amount of data for system identification (d) and different dimensions of the Koopman space (e).

we pick the initial frame $\boldsymbol{x}^0$ and the t'th frame $\boldsymbol{x}^t$ from a episode. Then we ask the model to generate a control sequence of length t to transfer the system from the initial state $\boldsymbol{x}^0$ to the target state $\boldsymbol{x}^t$. The control error is defined as the mean squared distance between the target state and the state of the system at time t.

5.4.1 Simulation

Figure 5-2 shows qualitative results on simulation. Our model accurately predicts system dynamics for more than 100 steps. For Rope, the small prediction error comes from the slight delay of the force propagation inside the rope; hence, the tail of the rope usually has a larger error. For Soft, our model captures the interaction between the body parts and generates accurate prediction over the global movements of the robot. The error mainly comes from the misalignment of some local components.

We evaluate the models by predicting 100 steps into the future on 500 trajectories and

Figure 5-3 shows quantitative results. IN and PN do not work well in the Rope and Swim environments due to insufficient system identification ability. The KPM baseline performs poorly in the Rope and Soft environments indicating the limited power of polynomial Koopman base functions. Our model significantly outperforms all the baselines.

5.4.2 Control

We compare our model with KPM, the Koopman baseline using polynomial basis. In Rope, we ask the models to perform open-loop control where it only solves the QP once at the beginning. The length of the control sequence is 40. When it comes to Soft/Swim, each model is asked to generate control signals of 64 steps, and we allow the model to receive feedback after 32 steps. Thus every model has a second chance to correct its control sequence by solving the QP again at the time step 32.

As shown in Figure 5-2, our model leverages the inertia of the rope and matches the target state accurately. As for controlling a soft body swinging on the ground or swimming in the water, our model can move each part (the boxes) of the body to the exact target position. The small control error comes from the slight misalignment of the orientation and the size of the body parts. Figure 5-4 shows that quantitatively our model outperforms KPM, too.

5.4.3 Ablation Study

Structure of the Koopman matrix. We explore three different structures of the Koopman matrix, *Block*, *Diag* and *None*, to understand its effect on the learned dynamics. *None* assumes no structure in the Koopman matrix. *Diag* assumes a diagonal block structure of K: all off-diagonal blocks (K_{ij} where $i \neq j$) are zeros and all diagonal blocks share the same values. *Block* predefines a block-wise structure, decided by the relation between the objects as introduced in Section 5.3.3.

	Simulation	Control
Diag	0.133 (0.174)	2.337 (2.809)
None	0.117 (0.083)	1.522 (1.288)
Block	**0.105 (0.075)**	**0.854 (1.101)**

Table 5.1: **Ablation study results on the Koopman matrix structure** (Rope environment). For simulation, we show the Mean Squared Error between the prediction and the ground truth at $T = 100$. For control, we show the performance with a horizon of length 40. The numbers in parentheses show the performance on extrapolation.

Table 5.1 includes our model's simulation error and control error with different Koopman matrix structures in Rope. All models are trained in the Rope environment with 5 to 9

masses. Besides the result on the test set, we also report models' extrapolation performance in parentheses, where the model is evaluated on systems with more masses than training, i.e., 10 to 14 masses.

Our model with *Block* structure consistently achieves a smaller error in all settings. *Diag* assumes an overly simplified structure, leading to larger errors and failing to make reasonable controls. *None* has comparable simulation errors but larger control errors. Without the structure in the Koopman matrix, it overfits the data and makes the resulting linear dynamics less amiable to the control.

Hyperparameters. In our main experiments, we set the dimension of the Koopman embedding to $m = 32$ per object. Online system identification requires 800 data samples for each training/test case. To understand our model's performance under different hyperparameters, we vary the dimension of the Koopman embedding from 8 to 64 and the number of data samples used for system identification from 200 to 1,600. Figure 5-4d shows that more data for system identification leads to better simulation results. Figure 5-4e shows that dimension 16 gives the best results on simulation. It may suggest that the intrinsic dimension of the Koopman invariant space of the Rope system is around 16 per object.

5.5 Discussion

Compositionality is common in our daily life. Many ordinary objects contain repetitive subcomponents: ropes and soft robots (as shown in this chapter), granular materials such as coffee beans and lego blocks, and deformable objects such as cloth and modeling clay. Although these objects are known to be very challenging to manipulate using traditional methods, this and previous chapters have made steady progress in constructing systems advancing the robots' capability to interact with these objects.

The specific formulation proposed in this chapter opens up a new direction by combining deep Koopman operators with graph neural networks. By leveraging the compositional structure in the Koopman operator via graph neural nets, our model can efficiently manipulate deformable objects such as ropes and soft robots, and generalize to systems with variable numbers of components. We hope this work could encourage more endeavors in modeling larger and more complex systems by integrating the power of the Koopman theory and the expressiveness of neural networks.

Part II

Learning Structured World Models for Physical Inference

Chapter 6

Learned Structured World Models for Causal Discovery From Videos

Starting the first chapter of Part II of this thesis, I will discuss the use of structured world models for physical inference, specifically for the inference of state, parameter, and latent scene structure of the underlying environment. In this chapter, I will focus on the task of causal discovery of physical systems from videos.

Causal discovery is at the core of human cognition. It enables us to reason about the environment and make counterfactual predictions about unseen scenarios that can vastly differ from our previous experiences. We consider the task of causal discovery from videos in an end-to-end fashion without supervision on the ground-truth graph structure. In particular, our goal is to discover the structural dependencies among environmental and object variables: inferring the type and strength of interactions that have a causal effect on the behavior of the dynamical system. Our model consists of (a) a perception module that extracts a semantically meaningful and temporally consistent keypoint representation from images, (b) an inference module for determining the graph distribution induced by the detected keypoints, and (c) a dynamics module that can predict the future by conditioning on the inferred graph. We assume access to different configurations and environmental conditions, i.e., data from unknown interventions on the underlying system; thus, we can hope to discover the correct underlying causal graph without explicit interventions. We evaluate our method in a planar multi-body interaction environment and scenarios involving fabrics of different shapes like shirts and pants. Experiments demonstrate that our model can correctly identify

the interactions from a short sequence of images and make long-term future predictions. The causal structure assumed by the model also allows it to make counterfactual predictions and extrapolate to systems of unseen interaction graphs or graphs of various sizes. Please refer to our project page for video illustrations: https://yunzhuli.github.io/V-CDN/.

This chapter was originally published as Li et al. [2020b] in the Conference on Neural Information Processing Systems (NeurIPS) 2020 along with co-authors Antonio Torralba, Animashree Anandkumar, Dieter Fox, and Animesh Garg.

6.1 Introduction

Causal understanding of the world around us is part of the bedrock of intelligence. This ability enables counterfactual reasoning, which often distinguishes algorithmic models from intelligent behavior in humans. This ability to discover latent causal mechanisms from data poses an important technical question towards building intelligent and interactive systems [Spirtes et al., 2000, Peters et al., 2017, Glymour et al., 2019]. For instance, Figure 6-1 shows an example of a multi-body system. While the images may convey the identity and position of balls, the structural causal mechanism is latent. Each pair of balls is connected to each other through an edge (say a spring, a rigid rod, or be free). Further, each edge may have a set of hidden confounders, like the rest length of a spring or the rigid rod, that causally affect the physical interaction behavior. The underlying causal structure and governing functional mechanism may not be apparent if observations, such as images, are implicit measurements of ground-truth variables [Zhang et al., 2017]. Furthermore, they can also vary across different configurations and scenarios within a domain. Hence, we need few-shot causal discovery algorithms purely from image data.

In a special case, where the entities are all disconnected and the only interactions are of collision-type, there have been a number of models proposed to employ an object-centric formulation in recent literature to directly predict the future from images [Watters et al., 2017, Janner et al., 2019, Santoro et al., 2017]. In such cases, model discovery may not even be necessary given these solutions. However, these *associative* models crumble in the face of more complex stationary underlying generative structures such as different types of latent edges and edge mechanisms [Gong et al., 2017]. Moreover, they are insufficient to capture novel generative structures and make counterfactual predictions at test time.

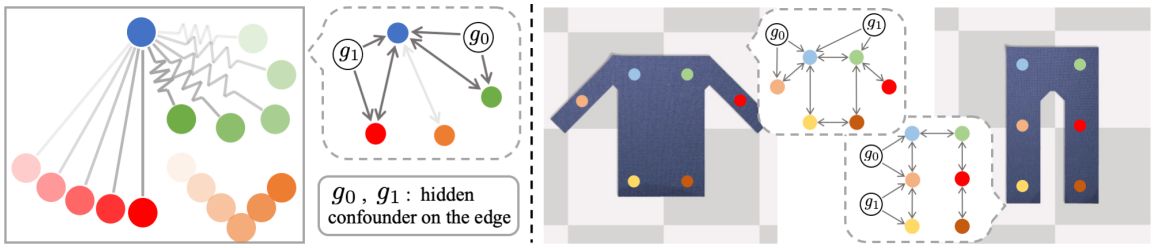


Figure 6-1: **Causal discovery in physical systems from videos.** The left figure shows balls, connected by invisible physical relations (shown in grey), moving around. Hidden confounding variables like edge type and edge parameters have a causal effect on the behavior of the underlying system. We humans can observe balls, infer the existence and variables on the edges between the balls, and predict the future. Similarly, in the cloth environment shown on the right, we can find a reduced-order representation by placing temporally consistent keypoints on the images and determining the causal relationships between them to reflect the cloth's topology.

In this work, we aim to discover the structural causal model (SCM) to predict the future and reason over counterfactuals. To recover an SCM only from images, we need to first learn a compact state representation, infer a causal graph among these variables as well as identify hidden confounders, finally learn the functional mechanism of dynamics. This is a particularly challenging task in that we only have images and do not have explicit knowledge of the node variables. Furthermore, we neither assume access to ground truth causal graph, nor the hidden confounders and the dynamics that characterize the effect of the physical interactions. In order to tackle this end-to-end causal discovery problem in an unsupervised manner, we learn from datasets that contain episodes generated from different causal graphs but with a shared dynamics model.

Summary of results. The main contributions of this work lie in the one-shot discovery of *unseen* causal mechanisms in new environments from partially observed visual data in a continuous state space. This entails jointly performing model class estimation, parameter inference, and thereby building a predictive model for new latent structures at test time in a meta-learning framework.

The proposed Visual Causal Discovery Network (V-CDN) consists of three modules for visual perception, structure inference, and dynamics prediction (Figure 6-2). Specifically, we train the perception module that extracts unsupervised keypoints from the images to enable node discovery, building upon Kulkarni et al. [2019]. The inference module then takes the predicted keypoints and infers the exogenous variables that govern the interactions between

each pair of keypoints using graph neural networks. Conditioned on the inferred graph, the dynamics module learns to predict the future movements of the keypoints. We consider a variety of configurations and scenarios, which gives us different combinations of variables, i.e., data from unknown interventions on the underlying system. Thus, we can hope to discover the correct underlying causal graph without explicit interventions.

Experiments show that our proposed model is robust to input noise and works well on multi-body interactions with varying degrees of complexity. Notably, our method can facilitate *counterfactual predictions* and *extrapolate* to cases with a variable number of objects and scenarios where the underlying interaction graphs are never seen before. Experiments in a fabric environment also demonstrate the generalization ability of our method, where the same model can handle fabrics of different types and shapes, accurately identifying the dependency structure and modeling the underlying dynamics even when state variables are a reduced-order keypoint-based representation of the original system.

6.2 Visual Causal Discovery in Physical Systems: V-CDN

In this section, we present the details of our model, which extracts structured representations from videos, discovers the causal relationships, infers the hidden confounding variables on the directed edges, and then predicts the future. Our model directly learns from raw videos, which recovers the underlying causal graph without any ground truth supervision.

Problem formulation. We consider a dataset of M trajectories observed from a latent generative dynamical system, where each datapoint is generated with unknown interventions on both the underlying causal graph structure and parameters affecting the mechanism. The generative process of each episode follows a *causal summary graph* [Peters et al., 2017], $\mathcal{G}_m = (\mathcal{V}_m^{1:T}, \mathcal{E}_m), m = 1 \dots M$, where $\mathcal{V}_m^{1:T}$ contains the subcomponents underlying the system at different time steps and $\mathcal{E}_m$, which we assume is invariant over time, denotes the causal relationships between the constituting components. Specifically, for each directed edge $(v_{m,i}, v_{m,j}) \in \mathcal{E}_m$, there are both discrete and continuous hidden confounders denoting the type and parameters of the relationship that determines the computation of the underlying structural causal model (SCM) [Pearl, 2009] and affects the behavior of the dynamical system. We further assume that in the dynamical system, there are no instantaneous edges or edges that go back in time. Note that the *causal summary graph* may contain cycles, but when

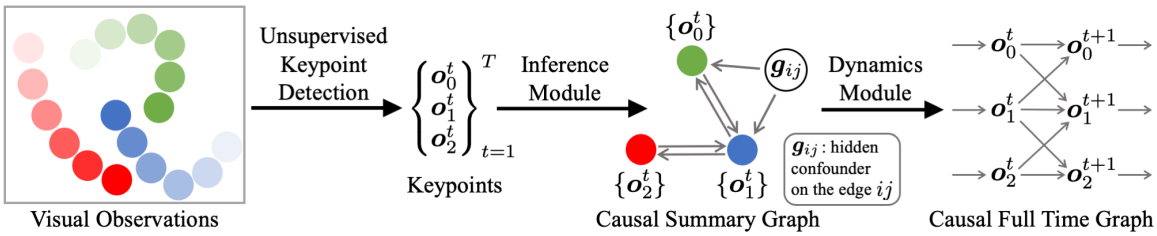


Figure 6-2: **Model overview.** Visual Causal Discovery Network (V-CDN) consists of three components: (a) a perception module to process the images and extract unsupervised keypoints as the state representation, (b) an inference module that observes the movements of the keypoints and determines the existence of the causal relations as well as the associated hidden confounders, and (c) a dynamics module that predicts the future by conditioning on the current state and the inferred *causal summary graph*.

spanning over time, the derived *causal full time graph* is a directed acyclic graph (DAG), as shown in Figure 6-2.

In this work, we consider the case where we only have access to the data in the form of image sequences, $\mathbb{I}_m = \{I_m^{1:T}\}$, without any knowledge of the ground truth causal structure and the intervention being applied, where I_m^t is an image of dimension $H \times W$, denoting the data we received at time t of episode m. The goal is to perform a one-shot recovery of the *causal summary graph* from a short sequence of images and simultaneously learn a shared dynamics model that operates on the identified graph to make counterfactual predictions into the future. This is a particularly challenging task and our method serves as a first step for tackling this problem in an end-to-end fashion using unsupervised intermediate keypoint representations.

Overview of Visual Causal Discovery Network (V-CDN). We aim to find a temporally-consistent (and possibly reduced-order) keypoint-based representation from images using a perception module trained in an unsupervised way,

$$\tilde{\mathcal{V}}_m^t = f_\theta^{\mathcal{V}}(I_m^t), \quad t = 1, \ldots, T, \tag{6.1}$$

where the function $f_\theta^{\mathcal{V}}$, parameterized by θ, takes raw images as input and outputs a set of keypoints in 2-D coordinates, $\tilde{\mathcal{V}}_m^t = \{\boldsymbol{o}_{m,i}^t | \boldsymbol{o}_{m,i}^t \in \mathbb{R}^2\}_{i=1}^N$, that reflect the constituting components in the system. Then, we use an inference module, $f_\phi^{\mathcal{E}}$, parameterized by ϕ, that

takes the sequence of detected keypoints as input and predicts the edge set, $\tilde{\mathcal{E}}_m$,

$$\tilde{\mathcal{E}}_m = f_\phi^{\mathcal{E}}(\tilde{\mathcal{V}}_m^{1:T}), \tag{6.2}$$

where $\tilde{\mathcal{E}}_m = \{(\boldsymbol{o}_{m,i}, \boldsymbol{o}_{m,j}, \boldsymbol{g}_{m,ij})\}$. $\boldsymbol{g}_{m,ij}$ includes $\boldsymbol{g}_{m,ij}^{\mathrm{d}}$ and $\boldsymbol{g}_{m,ij}^{\mathrm{c}}$, denoting the latent discrete and continuous confounders associated with the directed edge from j to i at episode m. $\tilde{\mathcal{V}}_m^{1:T}$ and $\tilde{\mathcal{E}}_m$ together constitute our discovered *causal summary graph*, conditioned on which, a dynamics module, $f_\psi^{\mathcal{D}}$, parameterized by ψ, aims to predict the state of the keypoints at time $T+1$,

$$\hat{\mathcal{V}}_m^{T+1} = f_\psi^{\mathcal{D}}(\tilde{\mathcal{V}}_m^{1:T}, \tilde{\mathcal{E}}_m). \tag{6.3}$$

By iteratively applying $f_\psi^{\mathcal{D}}$, we are able to make long-term future predictions.

The perception module, $f_\theta^{\mathcal{V}}$, the inference module, $f_\phi^{\mathcal{E}}$, and the dynamics modules, $f_\psi^{\mathcal{D}}$, are shared among all episodes in the dataset consisting of various causal graphs with different discrete and continuous hidden confounders, which enables one-shot adaptation to an unseen graph at test time and allows counterfactual predictions by intervening on the identified graph and rolling into the future using the dynamics module.

To train the system, we take an unsupervised keypoint detection algorithms [Kulkarni et al., 2019] as our perception module and train it on the image set, $\mathbb{I}$, for extracting temporally-consistent keypoints. The inference module and the dynamics module are trained together by minimizing the following objective:

$$\min_{\phi,\psi} \sum_m \sum_t \mathcal{L}(\tilde{\mathcal{V}}_m^{t+1}, f_\psi^{\mathcal{D}}(\tilde{\mathcal{V}}_m^{1:t}, \tilde{\mathcal{E}}_m)) + \lambda R(\tilde{\mathcal{E}}_m), \tag{6.4}$$

where $R(\cdot)$ is a regularizer imposed on the identified graph, e.g., to encourage sparsity.

6.2.1 Unsupervised Keypoint Detection From Videos

The perception module's task is to transform the images into a keypoint representation in an unsupervised way. In this work, we leverage the technique developed by Kulkarni et al. [2019]. In particular, we use reconstruction loss over the pixels to encourage the keypoints to disperse over the foreground of the image. During training, it takes in a source image I^{src} and a target image I^{tgt} sampled from the dataset, and passes them through a feature extractor $f_\omega^{\mathcal{V}}$ and a keypoint detector $f_\theta^{\mathcal{V}}$. The method then uses an operation called

transport to construct a new feature map, $\Phi(I^{\text{src}}, I^{\text{tgt}})$, using a set of local features indicated by the detected keypoints. A refiner network takes in the feature map and generates the reconstruction, $\hat{I}^{\text{tgt}}$. The module optimizes the parameters in the feature extractor, keypoint detector and refiner by minimizing a pixel-wise L_2 loss, $\mathcal{L}_{\text{rec}} = \|I^{\text{tgt}} - \hat{I}^{\text{tgt}}\|$, using stochastic gradient descent.

By combining the keypoint-based bottleneck layer and the downstream reconstruction task, the model extracts temporally-consistent keypoints spreading over the foreground of the images. We denote the detected keypoints at time t as $\tilde{\mathcal{V}}_m^t \triangleq f_\theta^{\mathcal{V}}(I_m^t)$, where $\tilde{\mathcal{V}}_m^t = \{\boldsymbol{o}_{m,i}^t | \boldsymbol{o}_{m,i}^t \in \mathbb{R}^2\}_{i=1}^N$.

6.2.2 Graph Neural Networks As the Spatial Encoder

We use graph neural networks as a building block to model the interactions between different keypoints and generate object- and relation-centric embeddings. Both the inference and the dynamics modules will have the graph neural networks as a submodule to capture the underlying inductive bias.

Specifically, for a set of N keypoints, we construct a directed graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where vertices $\mathcal{V} = \{\boldsymbol{o}_i\}$ represent the information on the keypoints and edges $\mathcal{E} = \{(\boldsymbol{o}_i, \boldsymbol{o}_j, \boldsymbol{g}_{ij})\}$ represent the directed relation pointing to i from j, where $\boldsymbol{g}_{ij}$ denotes the associated edge attributes.

We employ a graph neural network with a similar structure as the Interaction Networks (IN) [Battaglia et al., 2016] as our spatial encoder, denoted as ϕ, to generate the embeddings for the objects and the relations: $(\{\boldsymbol{h}_i\}, \{\boldsymbol{h}_{ij}\}) = \phi(\mathcal{V}, \mathcal{E})$.

6.2.3 Inferring the Directed Edge Set of the *Causal Summary Graph*

After we obtain the keypoints from the images, we use an inference module to discover the edge set of the *causal summary graph* and infer the parameters associated with the directed edges. The inference module takes the detected keypoints over a small time window within the same episode as input and outputs a posterior distribution over the structure of the graph. More specifically, we denote the keypoint sequence as $\tilde{\mathcal{V}}_m^{1:T} = \{\boldsymbol{o}_{m,i}^{1:T}\}_{i=1}^N$. Our goal is to predict the distribution of the edge set conditioned on the keypoint sequence using the parameterized inference function, $p_\phi(\tilde{\mathcal{E}}_m | \tilde{\mathcal{V}}_m^{1:T}) \triangleq f_\phi^{\mathcal{E}}(\tilde{\mathcal{V}}_m^{1:T})$.

To achieve our goal, we first use a graph neural network, as discussed in Section 6.2.2, to

propagate information spatially for each frame, which gives us both node and edge embeddings for each keypoint at each frame. We then aggregate the embeddings over the temporal dimension for each node and edge using a 1-D convolutional neural network. Another graph neural network takes in the temporal aggregations and predicts a discrete distribution over the edge types, where the first edge type denotes "null edge". Conditioned on a sample from the discrete distribution, the model will then predict the continuous edge parameters. The edge type and edge parameters together constitute the *causal summary graph*, which determines the existence and the actual mechanism of the interactions between different constituent components.

In particular, we first propagate the information spatially by feeding the keypoints through a graph neural network ϕ^{enc}, which gives us node and edge embeddings at each time step,

$$(\{\boldsymbol{h}_{m,i}^t\}, \{\boldsymbol{h}_{m,ij}^t\}) = \phi^{\text{enc}}(\tilde{\mathcal{V}}_m^t, \tilde{\mathcal{E}}^{\text{fc}}), \tag{6.5}$$

where the edge set, $\tilde{\mathcal{E}}^{\text{fc}}$, denotes a fully connected graph that contains an edge between each pair of keypoints with the edge attributes being zero. We then aggregate the information over the temporal dimension for each node and edge using 1-D convolutional neural networks (CNN):

$$\bar{\boldsymbol{h}}_{m,i} = \text{CNN}^{\text{obj}}(\boldsymbol{h}_{m,i}^{1:T}), \quad \bar{\boldsymbol{h}}_{m,ij} = \text{CNN}^{\text{rel}}(\boldsymbol{h}_{m,ij}^{1:T}), \tag{6.6}$$

which allows our model to handle input sequences of variable lengths.

Taking in the aggregated node and edge embeddings, we use another graph neural network, ϕ^{d}, that only makes predictions over the edges to predict the categorical distribution over the edge type:

$$\{\boldsymbol{g}_{m,ij}^{\text{d}}\} = \phi^{\text{d}}(\bar{\mathcal{V}}_m, \tilde{\mathcal{E}}_m^{\text{d}}), \tag{6.7}$$

where $\bar{\mathcal{V}}_m = \{\bar{\boldsymbol{h}}_{m,i}\}_{i=1}^N$ and $\tilde{\mathcal{E}}_m^{\text{d}} = \{(\bar{\boldsymbol{h}}_{m,i}, \bar{\boldsymbol{h}}_{m,j}, \bar{\boldsymbol{h}}_{m,ij}) | 1 \leq i, j \leq N, i \neq j\}$. The output $\{\boldsymbol{g}_{m,ij}^{\text{d}}\}$ represents the probabilistic distribution over the type of each edge. When an edge is classified as the first type, i.e., $\boldsymbol{g}_{m,ij}^{\text{d}} = 1$ is *true*, which we denote as "null edge", it will be removed in subsequent computation and no information will pass through it. Sampling from this discrete distribution is straightforward, but we cannot backpropagate the gradients through this operation. Instead, we employ the Gumbel-Softmax [Jang et al., 2016, Maddison et al., 2016] technique, a continuous approximation of the discrete distribution, to get the

biased gradients, which makes end-to-end training possible.

Conditioned on the inferred edge type $\{\boldsymbol{g}^{\mathrm{d}}_{m,ij}\}$, we would like to predict the continuous parameter on each one of the edges. For this purpose, we construct another edge set $\tilde{\mathcal{E}}^{\mathrm{c}}_m = \{(\bar{\boldsymbol{h}}_{m,i}, \bar{\boldsymbol{h}}_{m,j}, \bar{\boldsymbol{h}}_{m,ij}) | 1 \leq i, j \leq N, i \neq j, \boldsymbol{g}^{\mathrm{d}}_{m,ij} \neq 1\}$, and use a new graph neural network, ϕ^{c}, to predict the continuous parameters:

$$\{\boldsymbol{g}^{\mathrm{c}}_{m,ij}\} = \phi^{\mathrm{c}}(\bar{\mathcal{V}}_m, \tilde{\mathcal{E}}^{\mathrm{c}}_m). \tag{6.8}$$

We denote the resulting edge set as $\tilde{\mathcal{E}}_m = \{(\boldsymbol{o}_{m,i}, \boldsymbol{o}_{m,j}, \boldsymbol{g}_{m,ij}) | 1 \leq i, j \leq N, i \neq j, \boldsymbol{g}^{\mathrm{d}}_{m,ij} \neq 1\}$, where $\boldsymbol{g}_{m,ij} = (\boldsymbol{g}^{\mathrm{d}}_{m,ij}, \boldsymbol{g}^{\mathrm{c}}_{m,ij})$, indicating the topology of the *causal summary graph* with both the type and the continuous parameter of the edge effect. The inferred *causal summary graph* is then represented as $\tilde{\mathcal{G}}_m = (\tilde{\mathcal{V}}^{1:T}_m, \tilde{\mathcal{E}}_m)$.

6.2.4 Future Prediction Using the Forward Dynamics Module

The dynamics module, $f^{\mathcal{D}}_\psi$, predicts the future movements of the keypoints by conditioning on the current state and the inferred causal graph: $p_\psi(\hat{\mathcal{V}}^{T+1}_m | \tilde{\mathcal{V}}^{1:T}_m, \tilde{\mathcal{E}}_m) \triangleq f^{\mathcal{D}}_\psi(\tilde{\mathcal{V}}^{1:T}_m, \tilde{\mathcal{E}}_m)$, where we instantiate $f^{\mathcal{D}}_\psi$ as a graph recurrent network, ϕ^{dy}_ψ.

Since we are directly operating on the predicted keypoints from the perception module, the detected keypoints contain noise and introduce uncertainty on the actual locations. Hence, in practice, we represent the position in the future steps using a multivariate Gaussian distribution, where we predict both the mean and the covariance matrix of the next state for each keypoint.

6.2.5 Optimizing the Model

The perception module is trained independently using the reconstruction loss, $\mathcal{L}_{\mathrm{rec}}$. To train the inference module and the dynamics module jointly, we instantiate the objective function shown in Equation 6.4 by making an analogy to the ELBO objective [Kingma and Welling, 2013]:

$$\mathcal{L} = \mathbb{E}_{p_\phi(\tilde{\mathcal{E}}_m | \tilde{\mathcal{V}}^{1:T}_m)}[\log p_\psi(\hat{\mathcal{V}}^{T+1}_m | \tilde{\mathcal{V}}^{1:T}_m, \tilde{\mathcal{E}}_m)] - D_{\mathrm{KL}}(p_\phi(\tilde{\mathcal{E}}_m | \tilde{\mathcal{V}}^{1:T}_m) \| p_\psi(\tilde{\mathcal{E}}_m)), \tag{6.9}$$

For the prior $p_\psi(\tilde{\mathcal{E}}_m)$, we assume that each edge is independent and use a factorized distribution over the edge types as the prior, where $p_\psi(\tilde{\mathcal{E}}_m) = \prod_{ij} p_\psi(\tilde{\mathcal{E}}_{m,ij})$. The inference module and the dynamics module are then trained end-to-end using stochastic gradient descent to maximize the objective $\mathcal{L}$.

6.3 Experiments

The goal of our experimental evaluation is to answer the following questions: (1) Can the model perform one-shot discovery of the *causal summary graph* and identify the hidden confounders, including both discrete and continuous variables? (2) How well can the model extrapolate to graphs of different sizes that are not seen during training? (3) How well can the learned model facilitate counterfactual prediction via intervening on the identified summary graph?

Environment. We study our model in two environments: one includes masses, connected by invisible physical constraints, moving around in a 2-D plane, and the other one contains a fabric of various shapes where we are applying forces to deform it over time (Figure 6-3).

- **Multi-Body Interaction.** There are 5 balls of different colors moving around. At the beginning of each episode, we sample the invisible physical relations between each pair of balls independently, giving us the ground truth $\mathcal{E}_m$ that is fixed throughout the episode. For each pair of balls, there is a one-third probability that they are not connected or linked by a rigid rod or a spring. We also sample the continuous parameters for each existing edge and fix them within the episode, e.g., the length of the rigid relation or the rest length of the spring.

- **Fabric Manipulation.** We set up fabrics of three different types: a shirt, pants, and a towel, where we also vary the shape of the fabrics like the length of the pant leg or the height and width of the towel (Figure 6-5). We also apply forces on the contour of the fabric to deform and move it around. Our goal is to produce one single model that can handle fabrics of different types and shapes, instead of training separate models for each one of them.

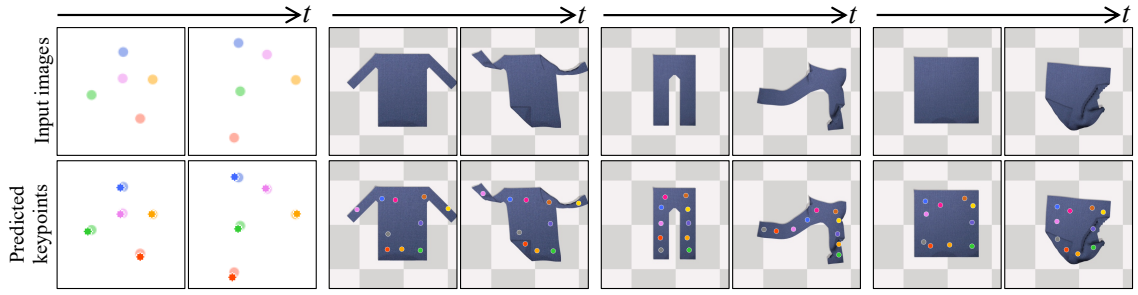
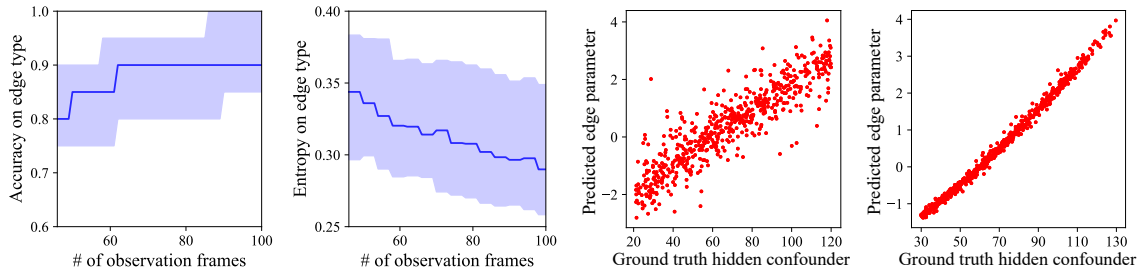


Figure 6-3: **Unsupervised keypoint detection.** The first row shows the input images, and the second row shows an overlay between the predicted keypoints and the image. The perception module assigns keypoints over the foreground of the images and consistently tracks the objects over time across different frames.



(a) Accuracy on edge type: {null edge, spring, rigid}

(b) Entropy of the probability distribution on edge type

(c) Scatter plot on the rest length of the spring relation

(d) Scatter plot on the length of the rigid relation

Figure 6-4: **Results on discovering the *Causal Summary Graph*.** Shown in (a) and (b), the accuracy of edge-type classification increases as the inference module observes more frames, which also effectively decreases the uncertainty, calculated as the entropy of the predicted distribution. As exhibited in (c) and (d), there is a strong correlation between the inferred continuous variable and the ground truth hidden confounder.

6.3.1 Results on Unsupervised Keypoint Detection

We employ the same architecture and training procedure described in Kulkarni et al. [2019] to train our perception module, $f_\theta^{\mathcal{V}}$. Figure 6-3 shows some qualitative results. Our perception module can spread the keypoints over the foreground of the image and consistently track the object. Please refer to our project page for video illustrations.

6.3.2 Discovery of the *Causal Summary Graph* and the Hidden Confounders

The inference module, $f_\phi^{\mathcal{E}}$, takes in a short sequence of the detected keypoints and aims to discover whether there is a causal relation, i.e., a physical connection, between each pair of keypoints and identifies the hidden confounders like the edge type and the edge parameters.

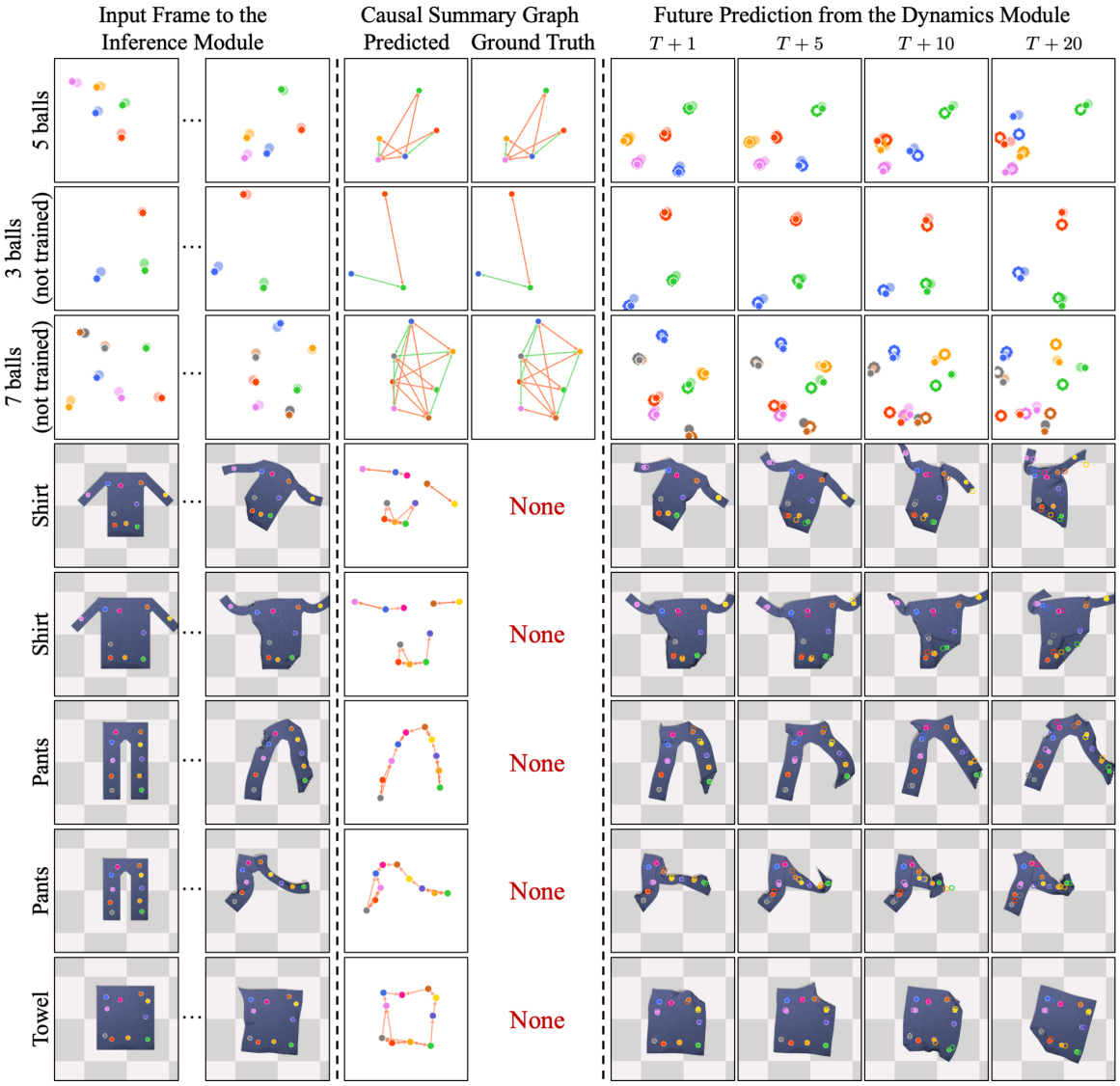


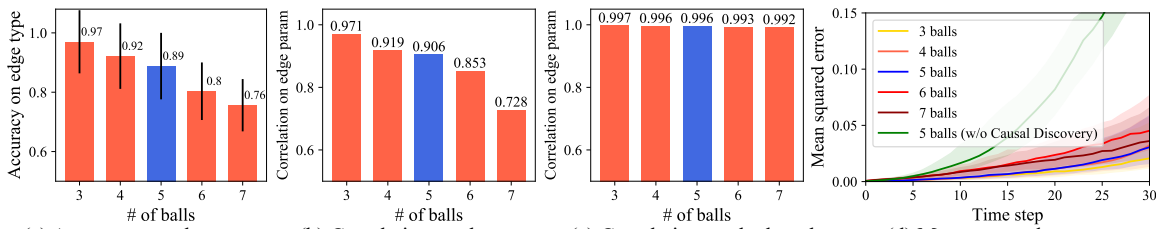
Figure 6-5: **Qualitative results on predicting the *Causal Summary Graph* and the future.** Our inference module observes a short sequence of images and performs one-shot discovery of the *causal summary graph*, which recovers the ground truth graph in the Multi-Body environment and captures the underlying connectivity structures in the Cloth environment. The unfilled circles in the right four columns indicate the model's prediction into the future. We overlay the predicted future keypoints with the truth future for comparison.

The predicted graph will be conditioned by the dynamics module, $f_\psi^{\mathcal{D}}$, for future prediction. The optimization procedure does not require any supervision on the attributes associated with the edges, which allows us to infer the hidden confounders in an unsupervised way.

In the Multi-Body environment, the perception module accurately tracks the location of balls, which allows us to perform a systematic evaluation of the model's performance by comparing its prediction with the ground truth *causal summary graph* used to generate the episodes. Because we are working in an unsupervised regime, where the predicted edge type is in a discrete latent space distinguishing between null edge, spring, and rigid relation, we need to find a global one-on-one mapping between the prediction, $\{\boldsymbol{g}_m^{\mathrm{d}}\}$, and the ground truth. We pick the one that gives us the highest accuracy, with the constraint that the first type, where there is no information passing through in the subsequent dynamics prediction, always corresponds to *null edge*. After the mapping, we evaluate the model's ability to predict the continuous confounder, $\{\boldsymbol{g}_m^{\mathrm{c}}\}$, by computing its correlation with the ground truth physical parameters like rest length of the spring connection.

The results are shown in Figure 6-4. As the model observes more frames, the classification accuracy increases, and the uncertainty decreases, which correlates with our intuition that as we obtain more observations from the environment, we have a better estimate of the exogenous variables that govern the behavior of the system. We also show the comparison with a baseline that is the same as our method except that it does not have the inference module. Our model significantly outperforms the baseline, indicating the importance of the correct modeling of the causal mechanism (Figure 6-6d). Figure 6-5 shows some qualitative results, where we include side-by-side comparisons between the identified *causal summary graph* and the ground truth.

For the cloth environment, the keypoints on the fabrics act as a reduced-order representation of the original system, where we do not know the ground truth *causal summary graph*. We encode the action as a 6-dimensional vector: the first three are the coordinates of the dragged point, and the other three indicate the movement, which will then be concatenated with the embedding of every keypoint. As shown in Figure 6-5, the same inference module produces different causal graphs for different types of fabrics that reflect the underlying connectivity patterns, which illustrates the model's ability to recognize the underlying dependency structure.



(a) Accuracy on edge type: {null edge, spring, rigid}

(b) Correlation on the rest length of the spring relation

(c) Correlation on the length of the rigid relation

(d) Mean squared error on future prediction

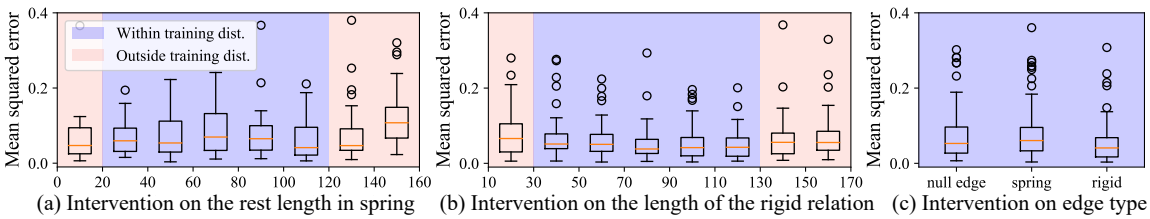
Figure 6-6: **Results on extrapolating to unseen graphs of different sizes.** Our inference module and dynamics module are trained only in environments containing 5 bodies. Thanks to the inductive bias captured by the graph neural networks in our model, it automatically generalizes to scenarios with different numbers of bodies from training. The blue bars in the figures show the performance on the test set in the same distribution we trained on, and the orange bars illustrate results on extrapolation. Surprisingly, the model has a better performance in environments with 3 and 4 balls, even if the model has never seen them before.

6.3.3 Extrapolation to Unseen Causal Graphs of Different Sizes

To evaluate our model's performance on extrapolation, we also create another 4 test sets in the Multi-Body environment, including 3, 4, 6, and 7 bodies, respectively, for which we need to train separate perception modules to reflect the number of moving components. However, the inference module and the dynamics module do not require retraining; instead, they can directly generalize to systems of different numbers of bodies. As shown in Figure 6-6, the blue bar shows the performance on the test set that has the same number of balls as the training set, while the other bars illustrate the model's ability to perform extrapolation. Interestingly, for environments with fewer balls, e.g., 3 or 4 balls, even if the model is not directly trained on these scenarios, the performance is yet better.

6.3.4 Counterfactual Prediction and Extrapolation on Parameter Change

In our experiment, we make counterfactual predictions by intervening on the estimated hidden confounders and evaluate how well the model predicts the future by making the same intervention on the ground truth simulator. The estimated confounders are in the latent space, which requires a mapping function to get the corresponding parameters in the original simulator. We use the same mapping as described in Section 6.3.2 to find the corresponding discrete variables, and train a simple linear regressor for transforming the continuous variable. Figure 6-7 shows the performance on counterfactual predictions, which illustrates our model's ability to answer "what if" questions and extrapolate to parameter ranges that are outside



(a) Intervention on the rest length in spring (b) Intervention on the length of the rigid relation (c) Intervention on edge type

Figure 6-7: **Results on counterfactual prediction.** We make counterfactual predictions by intervening on the identified *causal summary graph* and evaluate the performance by comparing the predicted future with the original simulator undergoing the same intervention at $T + 30$. The modeling of the causal mechanism allows it to extrapolate to parameter ranges outside the training distribution.

the training distribution.

6.4 Related Work

Causal discovery. Methods for causal inference from observations can broadly be categorized into three classes. Constraint-based methods (such as PC and FCI) rely on conditional independence tests as constraint-satisfaction to recover Markov-Equivalent Graphs [Entner and Hoyer, 2010, Spirtes et al., 2000, Colombo et al., 2011]. Score-based methods (such as GES) assign a score to each DAG, and perform searching in this score space [Chickering, 2002, Zheng et al., 2018]. The third class of methods exploits such asymmetries or causal footprints to uniquely identify a DAG [Shimizu, 2014, Kalainathan et al., 2018, Goudet et al., 2017, Zhang and Hyvärinen, 2009]. Further, causal discovery from a combination of observational and interventional data has been studied in the literature [Hyttinen et al., 2013, Ghassami et al., 2018, Kocaoglu et al., 2017, Wang et al., 2017a, Shanmugam et al., 2015, Peters et al., 2016, Rothenhäusler et al., 2015, Ke et al., 2019]. Many of these approaches either assume full knowledge of the intervention, make strong assumptions about the model class, or have scalability limitations.

Relational neural models. Including Chapter 3 and 5, many recent works have attempted modeling multi-body dynamics with graphs [Santoro et al., 2017, Battaglia et al., 2018, 2016] and attention [Goyal et al., 2019, Vaswani et al., 2017]. However, these methods assume the latent generative causal graph is stationary, resulting in poor generalization to variations in either graph structure or its functional parameters. A few recent works [Kipf et al., 2018, Alet et al., 2019] have tried to infer the relationship between different entities in the system using

a variational or meta-learning framework, where Kipf et al. [2018] also discussed connection to Granger causality. Still, we differ from them by directly working with image data and modeling not only the discrete but also the continuous hidden confounding variables.

Dynamics from videos. Video modeling and prediction have found much attention recently [Ye et al., 2019, Hsieh et al., 2018, Kumar et al., 2019, Yi et al., 2019]. The idea of learned latent space embeddings for unsupervised loss computation has also enjoyed recent success in prediction: Watter et al. [2015], Hafner et al. [2019b,a], and Li et al. [2022] (detailed in Chapter 4). Other people have employed keypoints (or particles) to provide more succinct and generalizable representations across a variety of use cases: keypoint/particle representation [Macklin et al., 2014, Mrowca et al., 2018, Ummenhofer et al., 2020, Sanchez-Gonzalez et al., 2020, Kulkarni et al., 2019, Minderer et al., 2019, Dundar et al., 2020] (including works [Manuelli et al., 2021, Li et al., 2018] discussed in detail in Chapter 2 and 3), deformable object modeling [Jakab et al., 2018, Suwajanakorn et al., 2018], instance independent class templates [Manuelli et al., 2019]. The work presented in this chapter builds on ideas from unsupervised visual representation learning and leverages it for visual causal discovery wherein the underlying model components use relational modeling to output a *Causal Summary Graph*, which has not been achieved in prior work for complex video datasets.

6.5 Discussion

Our method extracts a structured keypoint-based representation from videos, identifies the causal relationships between different constituting components, and makes predictions into the future. The model neither assumes access to the ground truth causal graph, nor the hidden confounders, nor the dynamics that describes the effect of the physical interactions; instead, we learn to discover the dependency structures and model the causal mechanisms end-to-end from images in an unsupervised way, which we hope can facilitate future studies of more generalizable visual reasoning systems.

Chapter 7

Learned Structured World Models for State and Parameter Estimations

In the second chapter of Part II, we dig deeper by considering physical inference tasks in more complicated systems involving the interactions between deformable objects, fluids, and rigid bodies. The ability to perform physical reasoning and to adapt to new environments is intrinsic to humans that we can intuitively recognize objects' physical properties and predict their motion, even when the objects are engaged in complicated interactions. Yet, such abilities still remain challenging to state-of-the-art computational models.

To handle systems involving objects of a diverse set of materials, in this chapter, we follow Chapter 3 to represent the scene using particles but build on top of it by presenting a neural framework that simultaneously reasons about system states from image observations and makes future predictions. The framework consists of a visual prior, a dynamics prior, and an inference module. The visual prior predicts a particle-based representation of the system from visual observations. An inference module operates on those particles, predicting and refining estimates of particle locations, object states, and physical parameters, subject to the constraints imposed by the dynamics prior (instantiated as the DPI-Nets presented in Chapter 3). We refer to the whole process as *visual grounding*. We demonstrate the effectiveness of our method in environments involving rigid objects, deformable materials, and fluids. Experiments show that our model can infer the physical properties within a few observations, which allows the model to quickly adapt to unseen scenarios and make accurate predictions into the future.

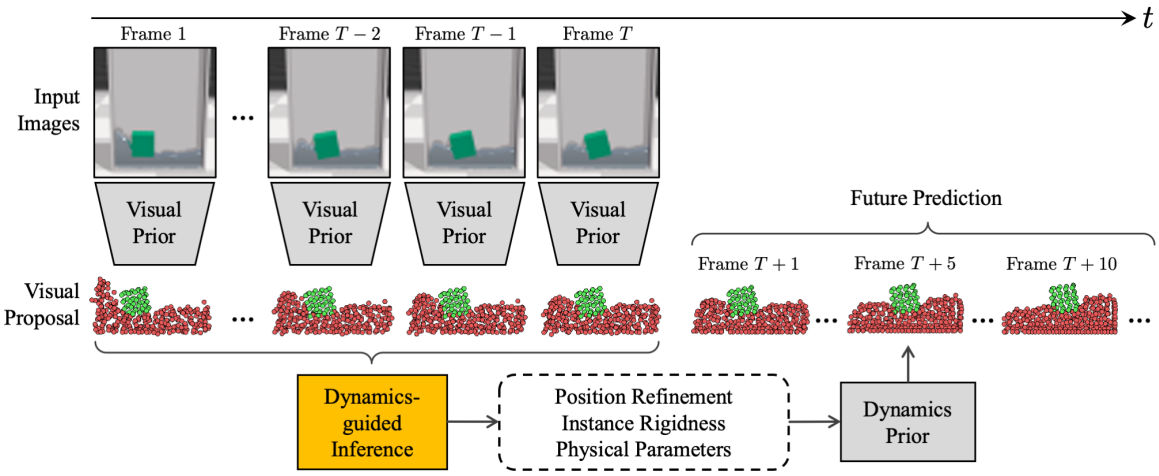


Figure 7-1: **Overview of Visually Grounded Physics Learner (VGPL).** The model takes a sequence of image frames as input, reasons about the underlying physical properties, and makes future predictions. The input frames first go through the perception module (*visual prior*), which reconstructs the input scene in particle representation by giving a proposal on particle positions and instance groupings. The inference module then refines the proposal by updating the particle positions, estimating the rigidness of each instance, and predicting the physical parameters of the scene. The dynamics module (*dynamics prior*) takes in outputs from the inference module and predicts particle positions into the future. Please check our project page for video illustrations: http://visual-physics-grounding.csail.mit.edu/.

This chapter was previously published as Li et al. [2020a] in the International Conference on Machine Learning (ICML) 2020 along with co-authors Toru Lin, Kexin Yi, Daniel M. Bear, Daniel L. K. Yamins, Jiajun Wu, Joshua B. Tenenbaum, and Antonio Torralba.

7.1 Introduction

Understanding the physical properties of interacting objects has been a long-standing goal in computer vision, robotics, and artificial intelligence. As humans, by merely watching objects interact, we are able to distinguish between different object instances, reason about their physical properties, and make predictions on their future motion. More impressively, our ability to recognize, model, and predict the dynamics of physical systems applies to not only rigid bodies, but also deformable objects such as elastic materials and fluids [Bates et al., 2019]. Given the example shown in Figure 7-1, humans can automatically identify the separation between liquid (water) and solid (floating cube), estimate their properties such as gravity, density, and viscosity, and predict key features of their future motion through mental simulation [Battaglia et al., 2013, Hamrick et al., 2016].

For computational systems, physical reasoning on interacting deformable objects has been

a highly challenging task, due to the diverse dynamical characteristics of different materials and their interactions. Take fluid as an example: fluids can deform, separate, merge, compress, and oscillate into arbitrary shapes, and some are hard to perceive due to their transparent nature. Prior works on system identification for robotics usually make strong assumptions on the structure of the underlying system [Ljung, 2001], or require a lot of data to train a forward model [Finn and Levine, 2017], and therefore have a hard time modeling complex deformable objects like fluids and adapting to new scenarios. The DPI-Nets [Li et al., 2018] presented in Chapter 3, on the other hand, have achieved strong results in modeling a variety of rigid and deformable object dynamics based on particle representations. Particles provide a dense and flexible representation that is well-suited for representing objects with diverse material and dynamical properties. Particles also facilitate relational inductive biases for more generalizable dynamics modeling. The learned dynamics model is able to accurately predict the forward evolution of the particle set based on their pre-designated grouping and physical parameters (stiffness, viscosity, gravity, etc.). However, inference of these essential properties directly from visual observations remains a challenging research problem, which limits DPI-Nets' applicability in more complicated real-world scenarios.

In this work, we focus on the problem of physical reasoning about interacting deformable objects and build on top of DPI-Nets by proposing a particle-based framework that jointly refines the particle locations and estimates their physical properties based on learned visual and dynamic priors. Our model, named Visually Grounded Physics Learner (VGPL), first generates a coarse proposal of particle positions and grouping from raw visual observations (*visual prior*). It then uses a learned dynamics model (*dynamics prior*) to guide the inference of several essential system properties such as refinement of particle positions, object rigidness, and physical parameter estimation (*dynamics-guided inference*). With those inferred quantities, our model can predict the future evolution of the system.

We evaluate our model in environments involving interactions between rigid objects, elastic materials, and fluids. Experiments demonstrate that our model, within a few observation steps, is able to refine the particle positions proposed by the visual prior, accurately predict the rigidness of the objects, and infer the physical parameters, which enables quick adaptation to new scenarios with unknown physical properties and making predictions into the future.

7.2 Related Work

Researchers have long been using neural networks to learn physical simulators [Chen et al., 1990, Wan et al., 2001]. Recently, including our works presented in Chapters 3, 5, and 6, people have demonstrated better generalization performance by using graph neural networks to capture the compositionality in dynamical systems [Battaglia et al., 2016, Chang et al., 2017, Sanchez-Gonzalez et al., 2018, Mrowca et al., 2018, Li et al., 2019b, Ummenhofer et al., 2020, Sanchez-Gonzalez et al., 2020]. These works make very few assumptions on the structure of the underlying systems, making them both general and flexible. Still, it remains as a question that how well they can handle raw visual inputs and adapt to environments of unknown physical properties.

Wu et al. [2015] introduced a method of inferring physical properties using MCMC, while others have tried differentiating through physics-based simulators to extract gradients [Todorov et al., 2012, Tedrake and the Drake Development Team, 2019, Degrave et al., 2019, Schenck and Fox, 2018b, Hu et al., 2019b, de Avila Belbute-Peres et al., 2018, Hu et al., 2019a, Liang et al., 2019], which showed strong results in solving inverse problems of various physical environments. However, their optimization process for dealing with the inverse problems is usually both time-consuming and prone to local optimum. Also, most of them directly operate on the state information of dynamical systems, lacking a way of handling raw visual inputs. This work aims to bridge the perception gap, enable physical reasoning from visual perception, and perform dynamics-guided inference to directly predict the optimization results, which allows quick adaptation to environments with unknown physical properties.

Including our work discussed in Chapter 4, people also have studied ways of reasoning about the physics and learning forward model directly from visual inputs [Finn and Levine, 2017, Babaeizadeh et al., 2018, Hafner et al., 2019b, Ha and Schmidhuber, 2018, Wu et al., 2017]. However, these works either directly learn dynamics model over pixels or operate on a latent space, which limits their ability to reason about the physical properties explicitly and make accurate long time future predictions. Other researchers have shown better performance with intermediate representations like instance masks [Fragkiadaki et al., 2016, Watters et al., 2017, Janner et al., 2019, Yi et al., 2019] or dense visual descriptors [Xu et al., 2019]. Instead, our model assumes a particle-based intermediate representation [Macklin and Müller, 2013], allowing us to model interactions between objects of different materials, including

rigid bodies, deformable objects, and fluids.

7.3 Approach

We present the Visually Grounded Physics Learner (VGPL), a model that learns to infer the properties of a complex physical system guided by a learned dynamics model and grounded to visual inputs. VGPL uses particles as the underlying state representation for physical modeling and inference. As shown in Figure 7-1, VGPL first generates a coarse proposal of the particle states from input visual observations via a perception module (*visual prior*), including the positions and groupings of the particles. Our model then applies an inference module on these proposals, generating the refined positions of the particles, and estimating other physical properties such as object rigidity and physical parameters. Finally, we use a dynamics module (*dynamics prior*) to guide inference of these properties, which can predict future particle states from historical trajectories, conditioned on these properties. We describe details of VGPL below.

7.3.1 Problem Formulation

Consider a system that contains M objects and N particles in its state representation. Given the visual observation $O = \{o^t\}_{t=1}^T$, our model first obtains a proposal of the particle position $\hat{X}'$ and the grouping information $\hat{G}$ for each particle, which is a probability distribution over the object instances, via a learned visual prior f_V. VGPL also incorporates a learned dynamics prior f_D that predicts future states based on the history of particle positions and physical properties of the system. These properties, including the rigidness of each object instance $\hat{Q}$ and the environmental physical parameters $\hat{P}$, are inferred by an inference module f_I. The inference module also generates a refinement $\Delta\hat{X}$ to the proposed particle locations. Our full model is summarized by the following equations:

$$(\hat{X}', \hat{G}) = f_V(O), \tag{7.1}$$

$$(\hat{P}, \hat{Q}, \Delta\hat{X}) = f_I(\hat{X}', \hat{G}), \tag{7.2}$$

$$\hat{X} = \hat{X}' + \Delta\hat{X}, \tag{7.3}$$

$$\hat{X}^{T+1} = f_D(\hat{X}, \hat{G}, \hat{P}, \hat{Q}). \tag{7.4}$$

The main objective of visual grounding is to infer the physical properties $(\hat{P}, \hat{Q})$ and refine positions $\Delta\hat{X}$ from the visual proposals of the states, such that the dynamics model predicts the most accurate particle trajectories. Our inference module f_I is tuned to minimize the following objective, constrained by fixed visual and dynamical priors f_V, f_D:

$$(\hat{P}^*, \hat{Q}^*, \Delta\hat{X}^*) = \underset{\hat{P},\hat{Q},\Delta\hat{X}}{\arg\min} \|\hat{X}^{T+1} - X^{T+1}\|. \tag{7.5}$$

In practice, the dynamics model iteratively predicts multiple steps into the future and this loss is computed over a finite time window.

7.3.2 Visual Prior

The visual prior proposes the particle state representation (position and grouping) from visual observations. The model architecture is built upon the point set generation network from Fan et al. [2017]. Given a sequence of visual observation images $O = \{o^t\}_{t=1}^T$, the model first uses a convolutional encoder for extracting latent features and then applies two fully connected heads for predicting the position and grouping of the particles. The model outputs the normalized particle positions in each frame, as well as the probability distribution over all object instances that the particle might belong to, $(\hat{X}', \hat{G}) = f_V(O)$. In particular, the particles are in the 3-dimensional space, $\hat{X}' = \{(x_i^{t\prime}, y_i^{t\prime}, z_i^{t\prime})\}_{i=1,t=1}^{N,T}$ and $\hat{G} = \{G_i^t\}_{i=1,t=1}^{N,T}$ is a set of probability distributions over the object instances.

The visual prior is trained on the ground-truth particle states acquired from the physics engine. The full loss function is written as

$$\mathcal{L}_\mathcal{V} = \frac{1}{TN} \sum_{t=1,i=1}^{T,N} \left[\|\hat{X}_i^{t\prime} - X_i^t\|^2 + H(\hat{G}_i^t, G_i^t) \right], \tag{7.6}$$

where H stands for the cross entropy loss.

In practice, in order to impose temporal consistency across different time steps, the network inputs a short sequence of images, and predicts the particle states over the same time window within a single pass.

7.3.3 Dynamics Prior

We adopt a particle-based dynamics model as the prior knowledge for guiding inference of the physical properties. At each time step, the positions of the particles X define a point cloud that indicates the spatial span of the objects in the environment. The particles form groups G to represent different object instances. Each particle has a binary rigidity label Q that indicates whether the object it belongs to is a rigid body. Finally, the environment also has a set of real-valued physical parameters P, e.g., viscosity, gravity, stiffness, etc.

Physical state representation. To better model the time evolution of individual particle states and their interactions, we represent the physical state of the system with a graph $\langle V, E \rangle$. Each vertex $v_i \in V$ contains the position information of a single particle concatenated with the physical parameters, $v_i = (X_i, P)$. Each edge $(s, r) \in E$ contains a binary value $a_{sr} \in \{0, 1\}$ that indicates whether the sender v_s and the receiver v_r belong to the same object. Since the underlying interactions between the particles are local, at each time step the particles are connected to their neighbors within a specified distance threshold d_e.

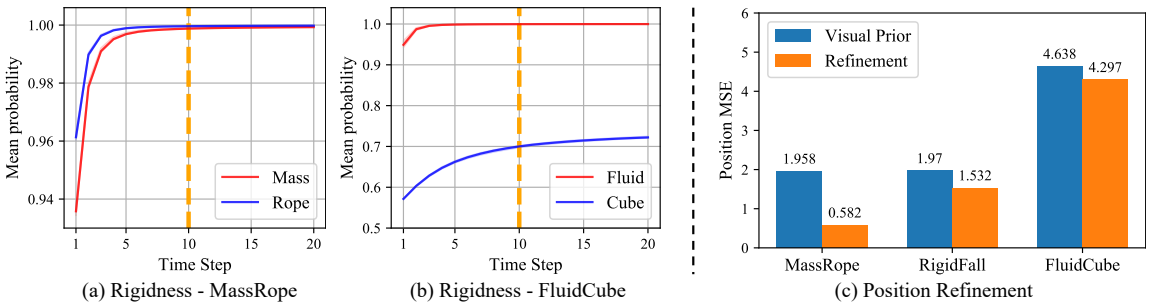
Spatial message passing. At each time step t, we use a graph neural network to perform the following updates on the graph representing the current physical state

$$g_{ij}^t = \phi_e(v_i^t, v_j^t, a_{ij}) \qquad (i, j) \in E \tag{7.7}$$

$$h_i^t = \phi_v(v_i^t, \sum_{j \in \mathcal{N}_i} g_{ji}^t) \qquad i = 1, 2, \ldots, N. \tag{7.8}$$

Here $\mathcal{N}_i$ is the set of all "neighbors" of vertex i with edges pointing to it. This process, which we refer to as *spatial message passing*, also employed by many other physics modeling systems [Battaglia et al., 2016, Sanchez-Gonzalez et al., 2018], generates a particle-centric encoding of the physical state h_i^t at each vertex and time step. The same type of message passing on graph is also used in the inference module as we will discuss in Section 7.3.4.

Dynamics prediction. We use the dynamic particle interaction network (DPI-Nets) introduced in Chapter 3 [Li et al., 2018] to perform dynamical updates on the particle state based on the vertex embeddings obtained from spatial message passing. To incorporate temporal information, the network inputs multiple historical steps of the encoded physical



(a) Rigidness - MassRope (b) Rigidness - FluidCube (c) Position Refinement

Figure 7-2: **Quantitative results on rigidness estimation and position refinement.** In (a) and (b), we show our model's performance on the rigidness estimation task in MassRope and FluidBox environments respectively. We use the mean probability of the ground truth rigidness label as the metric. The inference module was trained on inputs with only 10 time steps (the orange dashed line), but can extrapolate to both shorter and longer input sequence. Longer observation sequence leads to higher confidence, which is in line with our intuition. In (c), we show our model's performance on the position refinement task by comparing particle positions proposed by visual prior (in blue color) and after refinement by inference module (in orange color). We use the Mean Squared Error (MSE) between ground truth and predicted positions as the evaluation metric, scaled by 10^4. In all environments, MSE decreases after refinement.

state and predicts the particle positions one step ahead. The model handles rigid and non-rigid objects differently, therefore the update rules depend on the particles' grouping and rigidness:

$$\hat{X}_i^{T+1} = \phi(\{h_i^t\}_{t=1}^T | G_i, Q_i). \tag{7.9}$$

If a particle belongs to a rigid body, its motion can be decomposed into the translation of the body center plus the rotation of the particle with respect to the center. The update rule will apply a rigid body transformation (i.e., translation + rotation) to all particles belonging the same object to enforce the rigidity condition. For particles belonging to non-rigid objects, their position updates are independently computed per particle by a predictor network. Please refer to Chapter 3 or Li et al. [2018] for further details.

7.3.4 Dynamics-Guided Inference

The key step of grounding a learned dynamics model to visual inputs is to infer the physical properties underlying the observed system, in our case, the rigidness of the objects Q, and the environmental physical parameters P. We apply an inference module f_I to predict these properties from the observed particle proposals generated by the visual prior (Equation 7.3). The module also outputs refinement on the particles' positions $\Delta\hat{X}$. Since these properties are not directly accessible to the model, we use the learned dynamics prior

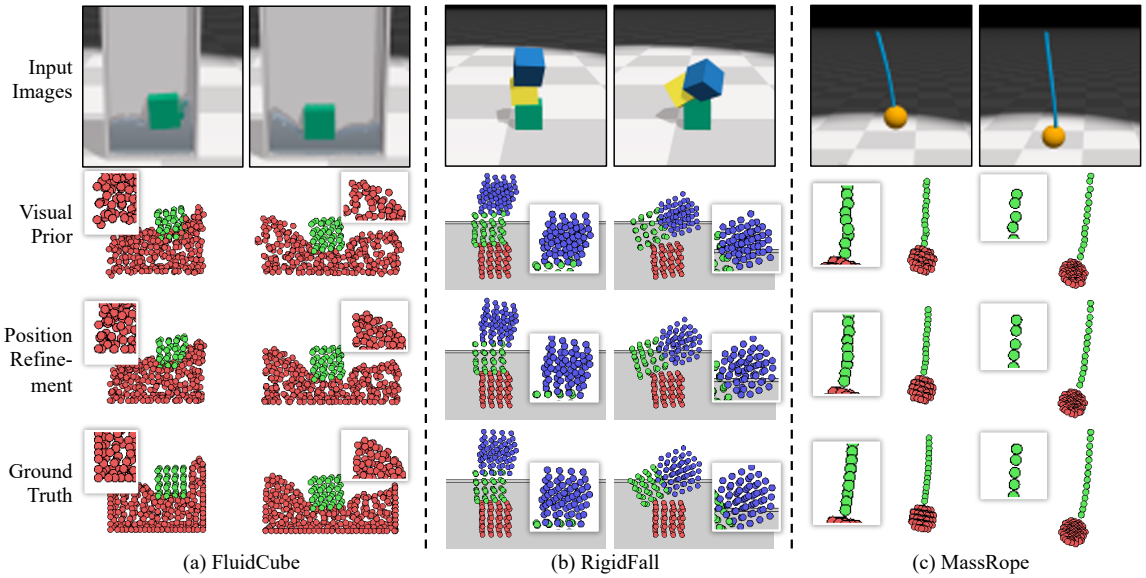


Figure 7-3: **Qualitative results on particle position refinement.** For each environment, we show side-by-side comparisons of two frames from the outputs of the visual prior, two frames from the outputs of position refinement, and two frames from the ground truth. For each output frame, we provide a zoom-in view to illustrate details of the particles. After refinement, (a) the fluids can better preserve the density constraint, (b) the rigid object is closer to the correct shape, and (c) the rope becomes less bumpy. The predicted particle positions after refinement all become closer to the ground truth.

to guide the inference of these properties. Details of the module are presented below.

Spatiotemporal message passing. Given a sequence of length T of the proposed particle positions and grouping $\hat{X}'$, $\hat{G}$, our inference module generates an embedding via *spatiotemporal message passing*. Similar to the dynamics prior, the input physical state is represented by a graph $\langle V, E \rangle$, where at each vertex $v_i = (\hat{X}', \hat{G})$ and the edges are connected between particles within a distance threshold d_e.

At each time step t, we first perform spatial message passing on the graph representation as described in Section 7.3.3 to obtain the vertex embeddings $\{h_i^t\}_{i=1}^N$ (Equations 7.7, 7.8). We then pass information on these embeddings along the temporal direction via a bi-directional recurrent network:

$$u_i^t = \phi_\tau(\{h_i^\tau\}_{\tau=1}^T)^t \qquad t = 1, 2, \ldots, T. \tag{7.10}$$

In practice, we use the multi-layer perceptron (MLP) for ϕ_e and ϕ_τ, and the bi-directional Gated Recurrent Unit (GRU) [Chung et al., 2014] for ϕ_τ. The weights of ϕ_τ are shared across all vertices.

Particle position refinement. We apply a refinement head ϕ_x on the spatiotemporal embedding u_i^t to predict refinement $\Delta\hat{X}$ on each particle's position at each time step

$$\Delta\hat{X}_i^t = \phi_x(u_i^t). \tag{7.11}$$

In our model, ϕ_x is chosen to be a MLP whose weights are shared across all particles and time steps.

Object rigidness estimation. To estimate the rigidness of each object in the system, a principled way is to start from embeddings that are associated with each object instances in the system. This is obtained by gathering the vertex embeddings from all particles belonging to the object and take the element-wise average to obtain an embedding vector of the object

$$w_j^t = \sum_{i \in \mathcal{O}_j} u_i^t / |\mathcal{O}_j|, \qquad j = 1, 2, \ldots, M, \tag{7.12}$$

where M is the number of objects in the system and $\mathcal{O}_j$ is the set of all particles belonging to the jth object, $\mathcal{O}_j := \{i | G_i = j\}$. This object embedding is then sent to a neural network to estimate the probability distribution on the rigidness

$$\hat{Q}_j^t = \phi_q(w_j^t). \tag{7.13}$$

In our model, ϕ_q is a MLP with sigmoid output, shared across all object instances and time steps.

Physical parameter estimation. Finally, we estimate the environmental physical parameters. Since the parameters are global, we use the full embeddings of all particles from all time steps and feed them together into a network ϕ_p to output a set of real numbers representing the estimated mean of the parameters, via $\hat{P} = \phi_p(\{u_i^t\}_{i=1,t=1}^{N,T})$. In practice, we set ϕ_p to be an MLP with hyperbolic tangent output.

Training. We use the pre-trained dynamics prior to guide the training of the inference module without access to the ground-truth of the inferred quantities. As shown in Equation 7.4, the dynamics model inputs the inferred quantities, including the refined position, grouping, rigidness, and physical parameters, and predicts the future positions of the particles. We

take the $\mathcal{L}_1$ distance between the predicted positions and ground truth as the loss function. The inference module is trained by stochastic gradient descent whose gradients are computed by back-propagating through the dynamics prior. Parameters of the dynamics prior stay frozen during training.

7.4 Experiments

We study our framework under three environments that incorporate different types of objects and facilitate rich interactions. In this section, we show results and present ablation studies on various inference and prediction tasks.

7.4.1 Environment

We use NVIDIA FleX [Macklin et al., 2014], a particle-based physics engine to generate all data for training and testing. The data includes visual observations and the corresponding particle states. For all three environments, we use 90% of the data for training and 10% for testing.

RigidFall. This environment simulates the motion and interaction of three rigid cubes. The cubes initially form a vertical stack with random noise added to their horizontal positions. The stack is released from above a rigid horizontal surface, and the cubes collide with one another as they fall under gravity. Each cube consists of 64 particles ($4 \times 4 \times 4$). The physical parameter of this environment is the gravitational acceleration, which is randomly sampled from $[-15.0, -5.0]$ for each simulation. The full dataset contains 5,000 simulations, each of which has 120 time steps.

FluidCube. In this environment, a rigid cube floats on top of a container of homogeneous fluid. The container can move horizontally to shake the fluid inside. During simulation, the container is initialized with a horizontal velocity of 0 and assigned a random horizontal acceleration at each time step. The rigid block is consisted of 48 particles, and the fluid is consisted of 300 particles. The viscosity of the fluid is randomly chosen from the range $[1.0, 100.0]$. We generate 2000 samples, each of which has 300 time steps.

Methods	MassRope	RigidFall	FluidCube
DensePhysNet	24.5% (15.1)	25.7% (15.4)	28.6% (15.0)
Ours w/o Rigidness	3.4% (2.2)	7.4% (4.1)	22.2% (14.7)
VGPL (ours)	**2.9% (1.3)**	**3.7% (2.7)**	**17.5% (13.6)**

Table 7.1: **Quantitative results on parameter estimation.** Below, we compare our model with DensePhysNet [Xu et al., 2019] and another model whose dynamics prior does not impose any constraints for rigid body motion. We measure the performance using the Mean Absolute Error (MAE) between each model's prediction and the ground truth. The numbers show the percentage of the MAE error with respect to the maximum parameter range. Numbers in parentheses report the standard deviation.

MassRope. In this environment, a rigid spherical mass is attached to an elastic rope whose upper end is pinned to an actuator that drives the rope's motion. We use positive y-direction as the upward direction, and the initial xyz-position of the actuator is $[0, 1, 0]$. The mass swings under a constant gravitational force and other internal forces such as rope tension. During simulation, the actuator at the upper end of the rope is assigned random accelerations along the horizontal plane (i.e. x- and z- directions), which also changes accelerations of the mass. The rigid mass is consisted of 81 particles, and the deformable rope is consisted of 14 particles. Rope stiffness is randomly chosen from range $[0.25, 1.20]$. We generate 3000 simulations, each of which includes 200 time steps.

7.4.2 Implementation Details

We present detailed model architecture and training paradigms below. All models are implemented in PyTorch [Paszke et al., 2019] and trained with the Adam optimizer [Kingma and Ba, 2015].

Visual prior. Our visual prior network consists of a feature encoder and fully connected output heads. The feature encoder has 4 stacked convolutional blocks, with 32, 64, 128 and 256 channels. Each block includes one 3×3 convolutional layer with batch normalization and ReLU activation. The prediction heads for particle position and grouping are both bi-layer MLPs with hidden size 2048.

The network is trained on the rendered visual observations of the system as well as the corresponding particle positions and grouping labels. We use a batch size of 50 and a learning rate of 10^{-4} to train the model for 2700 iterations on all environments. The particle positions

Methods	FluidCube				RigidFall				MassRope		
	$T+1$	$T+5$	$T+10$	$T+20$	$T+1$	$T+5$	$T+10$	$T+20$	$T+1$	$T+5$	$T+10$
w/o Rigidness	**3.864**	5.100	7.631	13.62	2.283	10.68	43.93	198.1	0.898	4.849	16.40
w/o Refinement	4.530	6.349	8.584	10.50	2.640	6.720	16.71	57.10	2.298	3.628	7.493
w/o Param. Est.	3.894	5.363	7.557	10.19	**2.110**	6.229	16.04	51.91	0.845	4.612	24.48
VGPL (ours)	3.887	**5.038**	**6.531**	**7.998**	2.112	**6.190**	**15.73**	**50.78**	**0.807**	**2.724**	**7.338**

Table 7.2: **Quantitative results on future prediction.** We show the Mean Squared Error (MSE) between the future predictions of the particle positions and the ground truth on all environments, scaled by 10^4. We evaluate our model's performance by ablating on different aspects of the model: (1) without rigidness estimation, (2) without parameter estimation, and (3) without positions and groupings refinement. As shown in the table, with better and more thorough estimation of physical properties, we can predict the future positions more accurately, especially when making long-term predictions.

are normalized. The sequence length of input and output data per forward pass is set to be 4. At inference time, given a sequence of input frames, we run the network on a sliding window over the sequence. In order to enforce temporal consistency, we move the window one step forward at a time and append the output at the last step of the moving window to the result sequence.

Dynamics prior. We adopt the DPI-Net [Li et al., 2019a] as the model for the dynamics prior. As explained in Section 7.3.3, the network operates on a graph representation of the physical state at each time step and predicts the information at the vertices at the next step. The distance threshold for edge connection between two vertices is set to be $d_e = 0.08$. In practice, the vertex and edge information are first separately encoded by two 3-layer MLPs with hidden and output size 150 before sent to the propagator networks ϕ_e, ϕ_v for spatial message passing. Both ϕ_e and ϕ_v include one fully connected layer with output size 150. The predictor heads for both rigid and non-rigid particles are 3-layer MLPs with hidden size 150 and ReLU activations.

Our dynamics prior is trained on the ground-truth particle trajectories for 10 epochs under each environment. We use a batch size of 4 and a learning rate of 10^{-5}. The model observes 4 past time steps and predicts 1 time step into the future. The particle positions are normalized at input and denormalized at output.

Inference module. We use the same network architecture in the inference module for feature encoding and spatial message passing as in the dynamics prior: 3-layer MLPs for vertex and edge embedding and single fully connected layer for message passing. All hidden and output layers have size 150. For temporal message passing, we use a bi-directional GRU

with two hidden layers of size 150. The prediction heads are MLPs with 1 hidden layer of size 150 and output size determined by the dimensions of the prediction targets. The rigidness and parameter estimation heads have extra sigmoid and tanh output activations respectively.

Our inference module is trained for 2 epochs on each environment, using a batch size of 2 and a learning rate of 10^{-5}. Length of the input and output sequences is set to be $T = 10$. The inference module is implemented by two separate networks with the same architecture: one for particle position refinement and rigidness prediction, and the other for physical parameter estimation. Each network takes in the proposals from the visual prior and predicts the desired variables, which, together with the ground-truth labels of the other network's output, are sent to the dynamics prior to predict particle positions. Loss is computed by comparing the predictions with the ground truth trajectories.

7.4.3 Results

We evaluate the performance of VGPL on the following tasks: accuracy of the inferred parameters including rigidness, position refinement and parameter estimation; and the prediction accuracy of future particle trajectories conditioned on these inferred properties. We also conduct ablation studies on these tasks to quantitatively evaluate the contributions from different model components.

Rigidness estimation. In this task, we train the inference module on input sequences of length $T = 10$ and evaluate rigidness estimation on sequences of lengths $T = 1$ through $T = 20$. We focus on two environments for this task, MassRope and FluidCube, since RigidFall only contains rigid objects.

We use the mean probability weight on the correct rigidness label as the quantitative measure of inference accuracy. The model will choose the correct label at inference time if this probability is above 0.5. Figure 7-2a-b shows the relation between the inference accuracy and the input time steps for all object types existing in these environments. Our model achieves a good performance at input length 10 on all object types, especially for both objects in MassRope and fluid in FluidCube, with the mean probability close to 1.0. We also observe nice results under different input time steps. The mean probability further increases as the input length is increased beyond 10. This result shows temporal message passing in our inference module is generalizable to various input lengths.

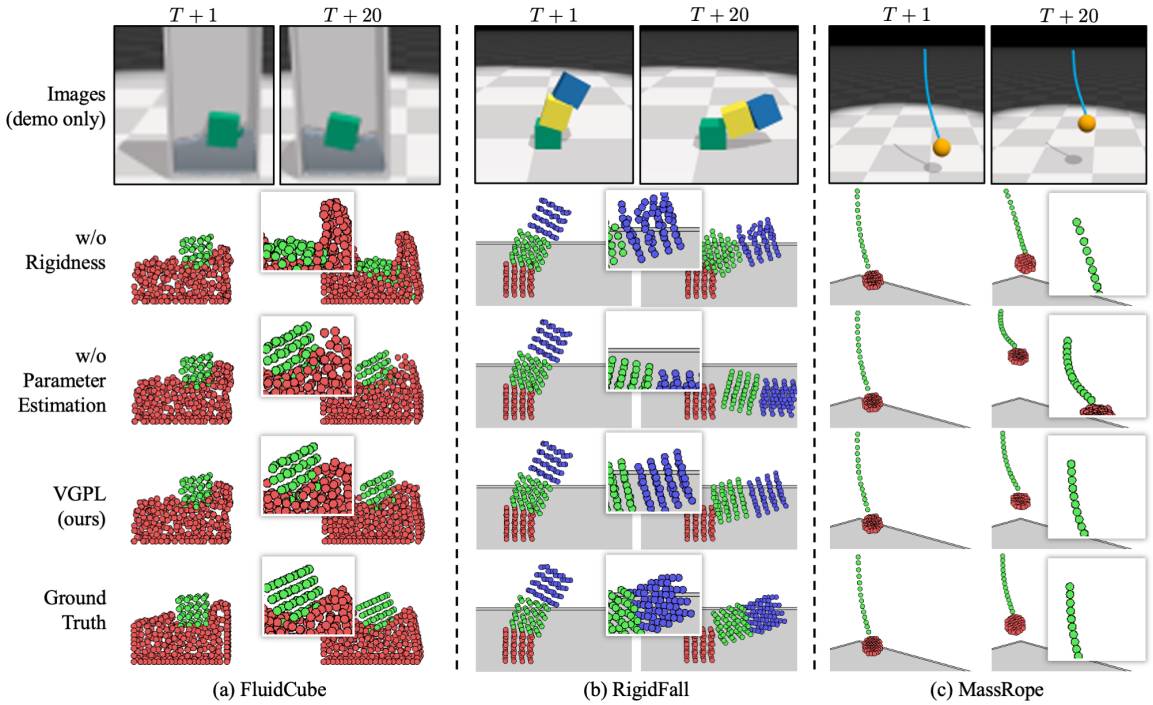


Figure 7-4: **Qualitative results on future predictions.** For each environment, we show results on predicted particle positions after 1 and 20 time steps. We compare the ground truth with the results of our model, together with versions without rigidness estimation or parameter estimation. For output frames after 20 steps, we provide zoom-in views to show more details of the predicted particles. As shown in the figure, without proper estimation of the rigidness, (a) the rigid cube melts into the fluids and (b) the rigid cube scatters. Without an accurate estimate of the physical parameters, (b) the rigid boxes fall faster onto the ground, and (c) the rope contracts more than the ground truth. In all environments, our model performs the best, especially on the longer horizon.

Our result also presents a notable gap between the mean probability of the cube versus the fluid in the FluidCube environment (Figure 7-2b). This is due to the fact that the cube particles mostly move along the same direction as the fluid particles, and therefore are harder to recognize the rigidness. Intuitively, the rigidness of the cube becomes more obvious when it is moving against the water particles, not when it is "riding the tide". As suggested by the result, a longer input sequence includes more opposite motion patterns between the cube and particle. It, therefore, leads to higher mean probability, which corresponds to higher confidence in the correct label of the rigidness.

Position refinement. We evaluate position refinement via the deviation of the predicted positions from the ground truth trajectories. Figure 7-2c shows a quantitative comparison between the positions before and after the refinement. The result shows improvements in the mean squared error (MSE) on all environments, especially for MassRope where the MSE

decreases by more than 3 fold.

We also show qualitative results in Figure 7-3 to compare visualizations of the particles before and after refinement with the ground truth. As shown in the figure, in FluidCube, the fluid particle density becomes more uniform after the refinement, which is in agreement with the underlying assumption of the physics simulator that the incompressible fluid preserves density. In RigidFall, particle refinement is able to correct the deformation of the cube. This correction will largely affect the collision property of the cubes in dynamics modeling. In MassRope, the particles on the rope become less bumpy after the refinement.

Physical parameter estimation. DensePhysNet [Xu et al., 2019] has shown to be able to learn representations that carry rich physical information and can directly be used to decode physical object properties such as friction and mass. We compare with DensePhysNet by evaluating how well the models can estimate the physical parameters. We employ the same model and training procedure as used in DensePhysNet that iteratively takes the action and the current visual observation as input and tries to predict the optical flow, which is estimated using the algorithm developed by Liu et al. [2009]. We then train a linear decoder that maps the resulting dense representation to the ground truth physical parameter. On FluidCube, RigidFall, and MassRope, the parameters of interest are the fluid's viscosity, gravitational acceleration, and the stiffness of the rope, respectively.

As shown in Table 7.1, where the numbers represent the percentage of the mean absolute error (MAE) with respect to the full range of these parameters, our model significantly outperforms DensePhysNet, showing the benefit of our formulation and the use of the visual and dynamics priors. We also compare the result with another model whose dynamics prior treats all particles equally without imposing any constraints for rigid body motion. Our model shows higher accuracy on all environments, suggesting that a stronger dynamics model can provide better guidance to the inference module, and therefore lead to a more accurate estimation.

Forward dynamics prediction. One important standard for judging the overall performance of visual grounding is the model's ability to accurately predict the future states of the system. To evaluate this, we send the inferred physical properties (i.e. position refinement, rigidness, and physical parameter) back to the dynamics prior and run forward pass on

the network to iteratively predict the particles' future trajectories. We compute the mean squared error between the predicted particle positions and the ground truth after 1, 5, 10, 20 time steps as the quantitative benchmark to evaluate the performance over different time horizons (Table 7.2). We also show qualitative results of predicted states in Figure 7-4.

To further study the impact of each inferred property on the overall performance, we perform ablation studies on each of the properties and compare with the full model. In the w/o rigidness model, the dynamics prior independently predicts the motion of each particle as non-rigid objects. This model predicts accurately within very short time horizon ($T + 1$) but fails after a few time steps as the rigid bodies melts into other shapes Figure 7-4a-b. In the w/o refinement model the dynamics prior inputs the coarse position proposals from the visual prior. This model shows poorer accuracy than the full model under all conditions due to the inaccurate inputs. The w/o parameter estimation model replaces the inferred parameter by a random number uniformly drawn from the parameter's range. Prediction of this model remains physically correct but deviates far from the ground truth at large time horizon Figure 7-4b-c. Overall, we show our full model achieves stronger performances than other baselines and demonstrate that all of the three inferred properties are essential to the task.

7.5 Discussion

Humans have a strong ability to mentally simulate a variety of different substances, which helps us to distinguish between rigid and deformable objects and infer the material properties from visual observations. In this work, we propose a model, named Visually Grounded Physics Learner (VGPL) that *grounds* the physical properties from vision, with the help of learned visual and dynamics priors. Our model employs a particle-based intermediate representation, which allows us to handle rigid bodies, deformable objects, and fluids. We have demonstrated in our experiments that our learned model can quickly adapt to new environments of unknown physical properties and make accurate predictions into the future.

Part III

Structured Models of Physical Interactions via Vision and Touch

Chapter 8

Learning Hand-Object Interactions Using a Scalable Tactile Glove

Part III of this thesis focuses on advancing the sensing capabilities of the robots by obtaining not only vision but tactile information during physical interactions (i.e., y_t in Equation 1.2). I will discuss our efforts in constructing multi-modal sensing and learning platforms with a focus on capturing the detailed tactile responses when making contact with the environment, which could lay the foundation for more structured and physically grounded modeling of the interaction process.

Humans can feel, weigh and grasp diverse objects, and simultaneously infer their material properties while applying the right amount of force—a challenging set of tasks for a modern robot [Bartolozzi et al., 2016]. Mechanoreceptor networks that provide sensory feedback and enable the dexterity of the human grasp [Johansson and Flanagan, 2009] remain difficult to replicate in robots. Whereas computer-vision-based robot grasping strategies [Mahler et al., 2019, Levine et al., 2016, Morrison et al., 2018] have progressed substantially with the abundance of visual data and emerging machine-learning tools, there are as yet no equivalent sensing platforms and large-scale datasets with which to probe the use of the tactile information that humans rely on when grasping objects. Studying the mechanics of how humans grasp objects will complement vision-based robotic object handling. Importantly, the inability to record and analyse tactile signals currently limits our understanding of the role of tactile information in the human grasp itself—for example, how tactile maps are used to identify objects and infer their properties is unknown [Saal et al., 2017]. Here we use a

scalable tactile glove and deep convolutional neural networks to show that sensors uniformly distributed over the hand can be used to identify individual objects, estimate their weight and explore the typical tactile patterns that emerge while grasping objects. The sensor array (548 sensors) is assembled on a knitted glove, and consists of a piezoresistive film connected by a network of conductive thread electrodes that are passively probed. Using a low-cost (about US$10) scalable tactile glove sensor array, we record a large-scale tactile dataset with 135,000 frames, each covering the full hand, while interacting with 26 different objects. This set of interactions with different objects reveals the key correspondences between different regions of a human hand while it is manipulating objects. Insights from the tactile signatures of the human grasp—through the lens of an artificial analogue of the natural mechanoreceptor network—can thus aid the future design of prosthetics [Osborn et al., 2018], robot grasping tools and human–robot interactions [Bartolozzi et al., 2016, Okamura et al., 2000, Cannata et al., 2008, Romano et al., 2011]. More video illustrations and details of developed sensors can be found on our project page: `http://stag.csail.mit.edu/`.

This chapter was originally published as Sundaram et al. [2019] in *Nature*, with co-authors Subramanian Sundaram, Petr Kellnhofer, Jun-Yan Zhu, Antonio Torralba, and Wojciech Matusik, in which Subramanian Sundaram and Petr Kellnhofer have made major contributions to the content of this chapter.

8.1 Main

Humans effortlessly manipulate objects and tools by applying precisely controlled forces [Marzke, 1997, Niewoehner et al., 2003, Feix et al., 2015a]. To understand the tactile feedback involved in the human grasp, we can use emerging machine learning tools to attempt to distil high-level properties and relationships from high-dimensional tactile data. Such tools require large-scale tactile datasets with high spatial resolution. However, large tactile datasets of human grasps covering the full hand do not exist because densely covering the human hand with tactile sensors is challenging. These tactile sensors come with strict requirements for the form-factor, resolution and mechanical compliance. Whereas electronic skins have made progress on the compliance requirements [Chortos et al., 2016], an electronic tactile glove with dense coverage and capable of collecting large datasets has yet to be demonstrated. The Tekscan Grip system (with its 349 sensors) is the closest

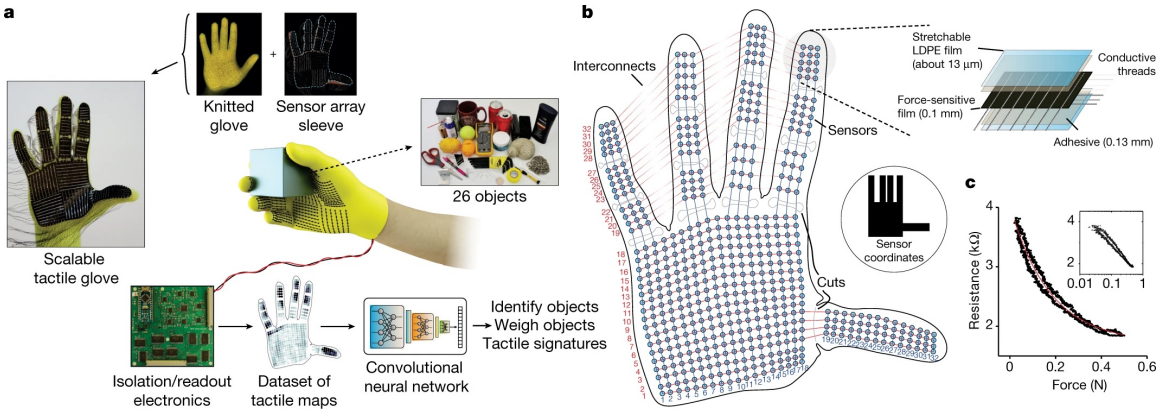


Figure 8-1: **The STAG as a platform to learn from the human grasp.** (a) The STAG consists of a sensor array with 548 elements covering the entire hand, attached to a custom knit glove. An electrical readout circuit is used to acquire the normal force recorded by each sensor at approximately 7.3 fps. Using this setup allows us to record a dataset of 135,187 tactile maps while interacting with 26 different objects. A deep convolutional neural network trained purely on tactile information can be used to identify or weigh objects and explore the tactile signatures of the human grasp. The glove shown at the centre is a rendering. (b) The design of the STAG architecture shows the individual locations of the 548 sensors, along with the interconnects, slot and 64 electrodes. The piezoresistive sensor array is fabricated by laminating simple materials and can be extended to different architectures easily (Figure 8-6 and 8-8). (c) Each sensor element responds to normal force by exhibiting a change in the through-film resistance. The sensor characteristics are repeatable across multiple devices and reliable over the long term (Figure 8-5). The inset shows the same characteristics in a logarithmic force scale (axes labels are as for the main plot).

high-cost commercially available system, but does not fully cover the hand (details and a comparative list are included in the Supplementary Information on the *Nature* website: `https://www.nature.com/articles/s41586-019-1234-z`). Current high-resolution optical tactile sensors [Li et al., 2014, Yamaguchi and Atkeson, 2016] and biomimetic multimodal sensor integrations [Wettels and Loeb, 2011, Park et al., 2015] have not successfully mapped a full human hand. Broadly, hurdles in creating a scalable tactile feedback network and acquiring large tactile datasets covering the hand have impeded our fundamental understanding of the human grasp.

We first present a simple method of fabricating a low-cost, scalable tactile glove (STAG) covering the full hand with 548 sensors. The STAG can record tactile videos (with frame rate approximately 7.3 Hz), measuring normal forces in the range 30 mN to 0.5 N (with quantization of about 150 levels and a peak hysteresis of about 17.5%). Importantly, the device can be constructed with low-cost materials (around US$10) and be used over long intervals. The STAG can be translated to a variety of different designs (see below). We introduce a large-scale dataset of tactile maps (135,000 frames) recorded using the STAG while

manipulating objects with a single hand; see Section 8.2 for dataset acquisition conditions. The spatial correlations and correspondence between finger regions that emerge from the dataset represent the tactile signatures of the human grasping strategy (Figure 8-1a). Here we observe and learn from successful daily human–object interactions with the long-term goal of aiding the development of robots and prosthetics.

The similarities in the underlying shape perception primitives between the visual and tactile domains are known [Yau et al., 2016]. We therefore hypothesized, on the basis of visual perception studies (showing that 16 × 16 pixels were sufficient for face recognition [Bachmann, 1991] and 32 × 32 pixels for scene recognition [Torralba et al., 2008] in visual data), that a similar minimal sensor count is suitable for a tactile sensor. The STAG consists of a sensing sleeve with 548 sensors attached on top of a custom knit glove. Figure 8-1b shows the locations of the 548 sensors and the 64 electrodes (fabrication details are included in the Methods). Fabricated gloves are shown in Figure 8-1a and Figure 8-4a (see Figure 8-4b for a high-resolution scan of the glove). The sensor array consists of a force-sensitive film (0.1 mm thick) addressed by a network of orthogonal conductive threads (0.34 mm) on each side, insulated by a thin adhesive (0.13 mm) and a low-density polyethylene (LDPE) film (about 13 μm). Each point of overlap between the orthogonal electrodes is sensitive to normal force, modulating the electrical resistance through the force-sensitive film. The force-sensitive film is laser-cut to fit the custom knit glove (yellow) along with holes to guide thread placement, and slots at the finger joints. The sensor laminate is thin and mechanically flexible. The typical force response of a single sensing element (Figure 8-1c), measured as the through-film resistance, changes from about 4 kΩ (unloaded) to below 2 kΩ (at a 0.5 N normal load). Each sensing element is sensitive to small forces (starting at about 25 mN; Figure 8-5a) and saturates beyond 0.8 N. The force response in the working range (30 mN to 0.5 N) is consistent across multiple devices (Figure 8-5b) and over multiple cycles (1,000 cycle tests in Figure 8-5c-d). The sensor elements show a stable resistance up to $60°C$ and become insulating at temperatures above $80°C$ (differential scanning calorimetry and resistance measurements are shown in Figure 8-5e-f).

We use a modified version of a grounding-based electrical isolation scheme [D'Alessio, 1999] (including charging resistors to improve the readout speed) to extract individual sensor measurements (the readout circuit costs about US$100; see Section 8.2 for fabrication details). The circuit topology is shown in Figure 8-4c along with the fabricated printed circuit-board

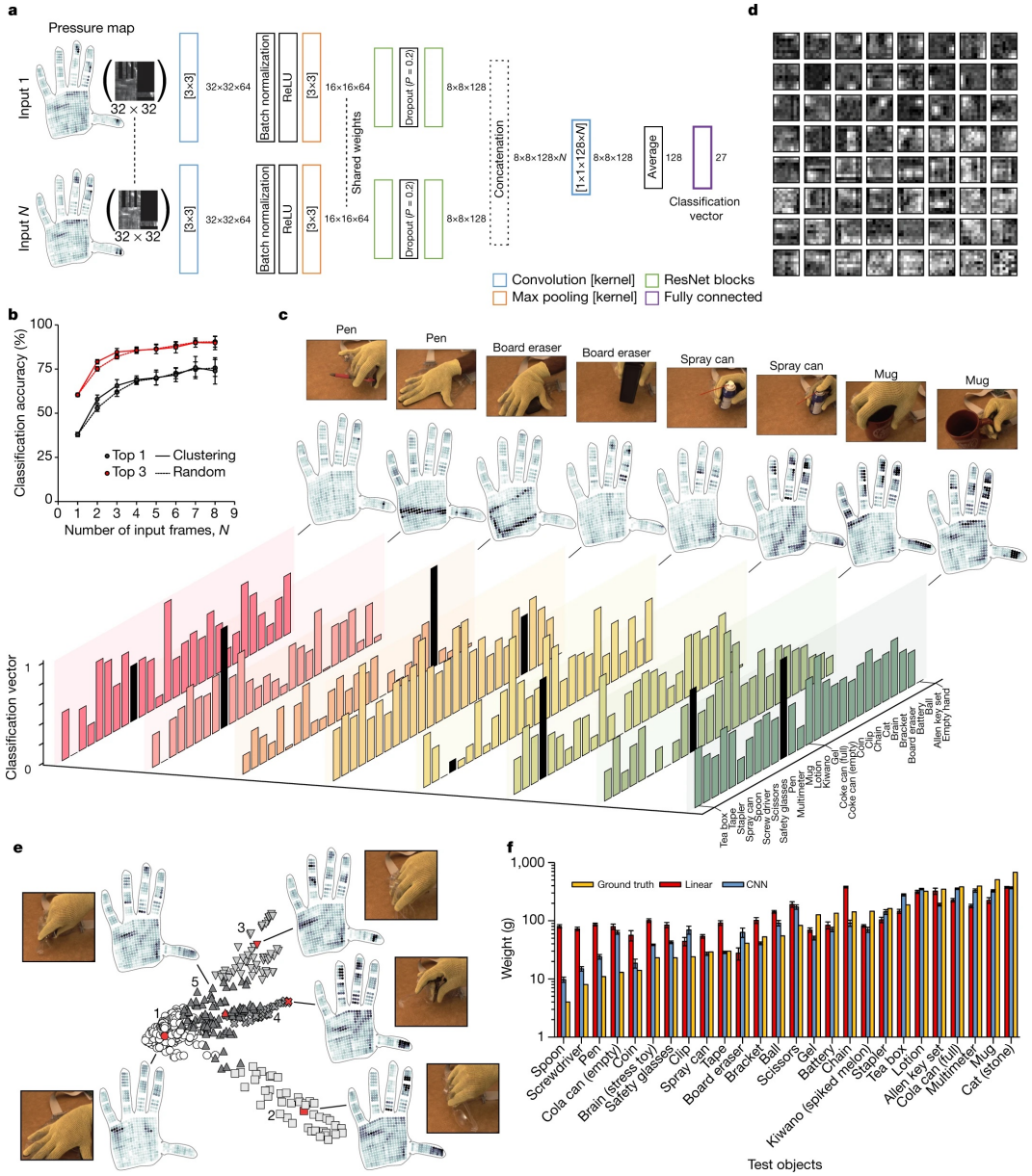


Figure 8-2: **Identifying and weighing objects from tactile information.** (a) The CNN architecture used for identifying objects from tactile information takes as input N arrays of tactile data (32 × 32 arrays). Rectified Linear Units (ReLU) are used to introduce non-linearity into the model. 'Dropout' is a regularization technique that randomly drops out nodes of the network to reduce overfitting to the training data. 'Max pooling' is used to reduce dimensionality of the data by passing only the locally highest activations. (b) The object identification accuracy is enhanced when using a diverse set of tactile maps from N distinct clusters as input when compared to a random choice of inputs; results are averaged over ten training runs (mean ± s.d.). Here each distinct cluster (as shown in (e)) is a group of similar grasps. (c) A representative set of examples during single-hand manipulation of objects. Tactile maps, corresponding visual images and the classification vectors (bottom) from single tactile map inputs are shown—the ground-truth object labels are marked in black. (d) The convolution filters learned by the scaled version of the network are shown (see Section 8.2). Note that the inputs and the network are scaled to visualize the filters at a higher resolution. The original 3 × 3 convolution filters of the network in a and the original ImageNet-trained ResNet filters are shown in Figure 8-10i-j respectively. (e) Clustering tactile maps from a single object interaction helps to identify a diverse set of tactile maps that correspond to that object. Five different clusters are used to extract five tactile inputs (N) shown in red. (f) 'Leave-one-out'-based CNN weight prediction results compared with a linear model (mean ± 95% confidence interval).

image (Figure 8-4d). The sensor response at the output of the amplifier (and the analog-to-digital converter, ADC) is linear with respect to the force. We note that the STAG design can be simplified to rapidly fabricate regular arrays; Figure 8-6 shows 1,024-element sensors with sensor spacing of 2.5 mm. Such regular arrays fixed on flat surfaces can record the resting identities of different objects (Figure 8-7). Furthermore, despite the weak extensibility of the force-sensitive film, we can enhance the achievable stretchability by incorporating auxetic designs [Ko et al., 2015] into the sensor structure as shown in Figure 8-8. The auxetic prototype with 10 $\times$ 10 elements can be stretched in multiple directions, as well as folded or crushed (Figure 8-8e-f).

The reliability of our STAG prototype allows us to record tactile videos (and corresponding visual images for illustration) during interactions with a set of 26 objects (Figure 8-9) with a single hand over many hours (total length of recordings exceeding 5 hours; see Section 8.2 for dataset acquisition details). We identify the specific frames in which objects are in contact with the glove (see Section 8.2 for filtering procedure). We train a convolutional neural network (CNN) to identify objects using these filtered frames (32 $\times$ 32 arrays in sensor coordinates). We use a ResNet-18-based architecture [He et al., 2016] that takes N input frames (Figure 8-2a; see network implementation in the Methods). The classification accuracy improves with the number of inputs and reaches its maximal performance with about seven random input frames (Figure 8-2b). This is expected, because multiple contacts with an object help to identify it more accurately. Figure 8-2c shows eight example tactile frames along with their output classification vectors. Here we observe that it is easy to identify the mug when it is held by the handle but it can be confused with a can or other objects when lifted from the sides. Likewise, the elongated shape of the pen is easier to identify when it is in contact with the palm than when it is held between fingers. Interestingly, when a mug is held by the handle (or while a spray can is being held), the distinct hand pose captured in the tactile map from sensors around the joints (proprioceptive data) may also help in object classification.

The first 3 $\times$ 3 convolution filters learned by our network are shown in Figure 8-10i. To understand the features at a higher resolution, we scaled the input resolution by three and adapted the network elements appropriately (see Section 8.2). The first layer convolution filters learned by the adapted network are shown in Figure 8-2d. The network primarily learns blob-like point detectors, edge detectors and low-frequency filters. The visual domain

filters learned by standard ResNet-18 trained on the ImageNet [Russakovsky et al., 2015] dataset are shown in Figure 8-10j for comparison. Furthermore, we visualized the features of our trained network (with Network Dissection; see Section 8.2 for details) and observed that the early convolution layers are activated in small regions. The higher-layer convolution filters are often activated by more complex grasp-related concepts. Supplementary Figure 4b on the *Nature* website (`https://www.nature.com/articles/s41586-019-1234-z`) shows the activation maps of filters that respond to larger contact patterns, or when specific hand regions are used.

Humans are easily capable of associating similar grasps based on motor movements, and the identification of an object is probably better performed when choosing the most distinct (informative) set of grasps. Motivated by this, instead of choosing N random frames, we identified the most diverse set of N frames for an input recording by k-means clustering (an example with N = 5 clusters is shown in Figure 8-2e; see an interactive version of the map in Supplementary Data 1 on the *Nature* website: `https://www.nature.com/articles/s41586-019-1234-z`). The classification accuracy using N input frames (one from each cluster) shows that clustering provides a marginal improvement in accuracy when a small set of inputs is used ($N < 4$ in Figure 8-2b). We note that the results of clustering-based inputs converge with those of randomly chosen inputs for large N because a random selection captures the data well when N is large. The corresponding confusion matrices are shown in Figure 8-10a–h. We observe that objects with similar shapes, sizes or weights are more likely to be confused with one another. Light objects such as the safety glasses, the plastic spoon or the coin are more easily misclassified, whereas large, heavy objects with distinct signatures, like the tea box, can easily be detected even with a small number of input frames.

The object identification tests described above help to evaluate the capability of the STAG in capturing useful data. We also evaluated the classification performance of lower sensor counts by downsampling the tactile data either uniformly or based on different regions of the hand. The classification accuracy drops considerably as the effective number of sensors is reduced, thereby highlighting the need for a high sensor count.

In addition to identifying objects, humans can easily estimate the weight of objects from tactile signals. The ability to estimate weights is of practical use in robotics and has been the focus of human perception experiments [Brodie and Ross, 1984]. To estimate the weights of objects from tactile interactions, we used a restricted dataset of multi-fingered grasps where

the object was picked up from above. After an object is picked up, a single frame is used as an input to a CNN to predict its weight. Note that the training and test data have disjoint sets of objects (see Section 8.2). Figure 8-11 shows a representative set of tactile frames and corresponding images. Results in Figure 8-2f show that our network performs better than a naive linear model over the entire weight range.

We looked at the typical sequence of tactile maps immediately before and after an object is grasped (an example is shown in Figure 8-3a) to understand the grasp in depth. The hand is increasingly articulated to fit the object closely, during which time the proprioceptive signal in the tactile map increases gradually until contact during the 'reach' phase [Johansson and Flanagan, 2009]. When contact is first made with an object ('load' phase), the mean pressure of the frame increases suddenly, resulting in a steep temporal gradient; the red dot shows the detected frame. In brief, we identify the prior local minimum as the frame just before contact (blue dot), which has the maximum hand pose signal (see complete processing details in the Methods). This empty hand pose frame is subtracted from the local maximum frame (green dot; lift and hold phase), which is treated as the frame with the maximal object information. This approach helps in decomposing the tactile map into the hand pose signal and the object-related pressure map (a detailed description of the decomposition is included in the Methods). We analysed the Pearson correlation coefficient between a selected sensor and the remaining sensors in the glove as shown in Figure 8-3b. Our correlations are shown in the range from 0 to 1; we did not observe any substantial negative correlations between sensors. We find the largest correlations between the fingertips and the thumb base, where the forces are dominantly applied; this is an expected signature of precision grip in humans. The measured correlations for each sensor can be viewed in our interactive map on the project website. The corresponding correlation between sensors at the fingertips and the full hand, with the decomposed hand pose signal shows little structured correlation, in part demonstrating the effectiveness of our decomposition method (Figure 8-12). Canonical-correlation analysis on the decomposed object-related tactile map across the different regions of the hand shows the collaborative role between the distal phalanges of the large fingers, which is most often used in generating forces during object grasps (Figure 8-3c). In the other phalanges, the distribution is more uniform, corresponding to closed grasps where a large part of the hand surface is in contact with the object at once (check our project page for an interactive map of the region-level correlations). The correlations between different sensors

indicate the collaborations between different hand regions; the nature of the human grasp is known to be collaborative [Napier, 1956, Lederman and Klatzky, 1987]. We have thus empirically and quantitatively observed such collaborations and their spatial extent purely from high-resolution tactile signals. To directly test the proprioceptive content of the tactile signals from the STAG, we articulated specific hand poses in the absence of an object (see G1 to G7 in Figure 8-13a) based on a standard grasp taxonomy [Feix et al., 2015b]. We observed that the tactile maps related to specific hand poses can be classified with 89.4% accuracy; a visualization of the clustering using t-distributed stochastic neighbour embedding (t-SNE) is shown in Figure 8-13b; confusion matrix from the classification test is shown in Figure 8-13c. Although hand recognition from visual images has become increasingly robust [Simon et al., 2017], extracting other meaningful feedback signals (such as establishing contact with an object) remains challenging without a scalable tactile sensing strategy.

Our results demonstrate the broad utility of high-dimensional tactile sensors as well as highlight their enabling potential for future work. The current study focuses mainly on the spatial relationships of tactile signals; the dataset also presents important relationships between sensors that are temporally linked together. These temporal relationships spotlight the dynamics of actions performed by humans. Linking these temporal relationships along with the spatial correspondences between tactile signals would greatly enhance our understanding of the basic principles of dexterous manipulation. Likewise, the dataset presented here also contains synchronized visual information along with the tactile data. In this regard, the STAG is a useful testbed for multimodal learning across visual and tactile domains, which is potentially useful for robotics applications. Finally, the STAG hardware platform itself can be augmented, for example, the STAG could be fitted with diverse sensors that mimic the different sets of mechanoreceptors in the human hand. In addition, transmitting data wirelessly from a wearable module and more compact packaging will extend its utility in manipulation tasks that require considerable mobility.

8.2 Method

8.2.1 STAG Sensor Fabrication

The STAG consists of a sensing laminate attached to a light, custom-knit glove designed not to interfere with hand movements (loosely knitted at half-gauge on a Shima Seiki

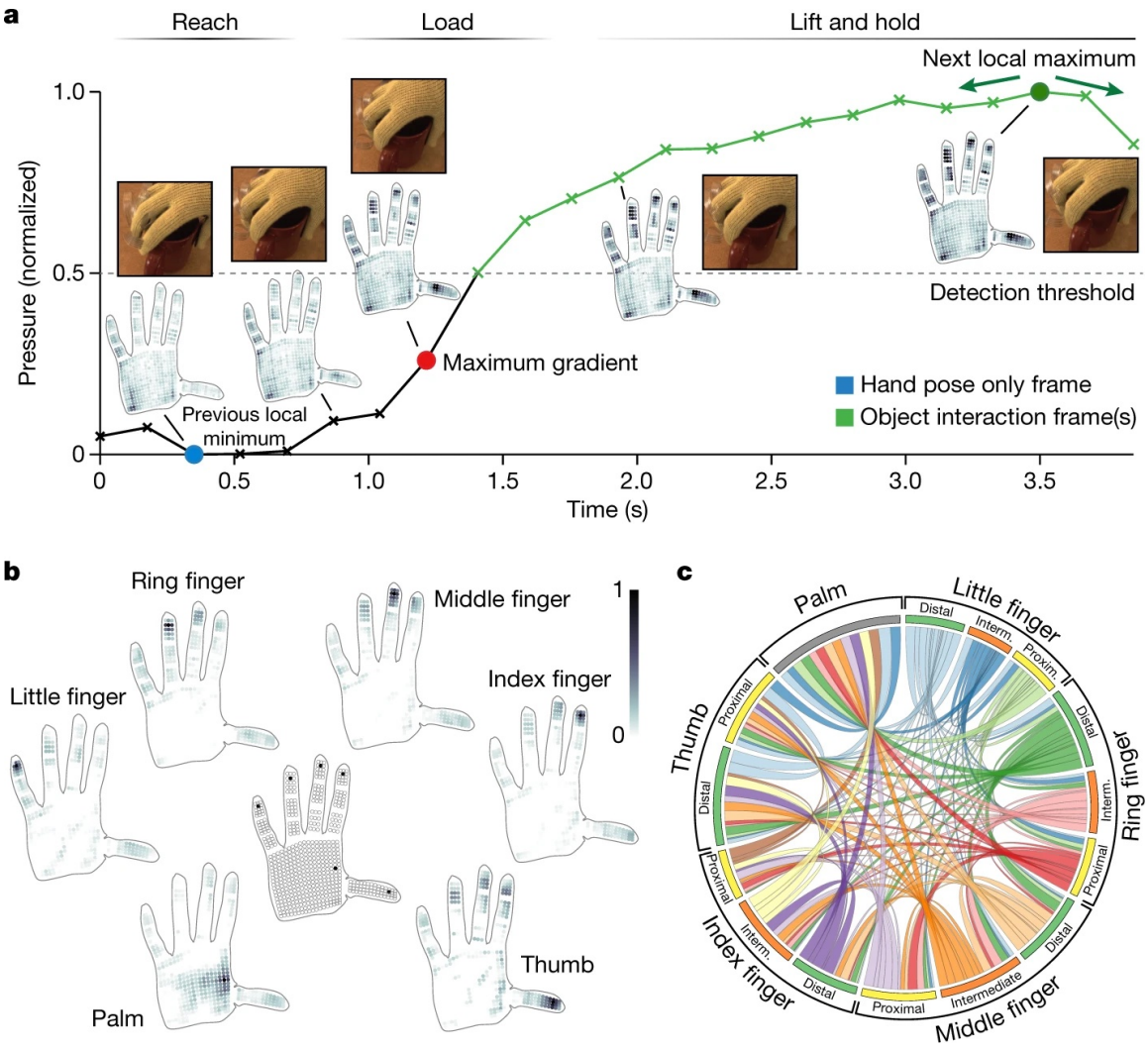


Figure 8-3: **Cooperativity among regions of the hand during object manipulation and grasp.** (a) In a typical interaction sequence, the hand increasingly gets articulated until the point of contact (reach phase) and experiences a sudden rise in tactile forces as the object is held. We use the maximum gradient (red) in the load phase of the action to decompose a tactile map of when the object is held into two parts—the hand pose signal (blue) and the object-related pressure (blue frame subtracted from the green frames). The object interaction frames are automatically detected and marked in green; full signal decomposition details are described in the Methods. (b) The set of decomposed object-related pressure frames can be used to extract correlations between specific sensors and the full hand. The maps here show the correlations between select pixels (marked in the central hand) and the full hand. Most fingertips are used with other fingers and the thumb (a signature of precision grasp), whereas the regions on the palm are typically used when grasping objects that cause the full palm to come into contact with the object. (c) The circular plot shows the relative correspondences between different parts of the hand (see Section 8.2). The distal phalanges of the large fingers are usually used with the thumb to generate forces while gripping an object and result in strong concurrence.

SWG091N2 15-gauge v-bed knitting machine). The sensing laminate is made by first laser-cutting (Universal PLS 6.150D CO_2 laser cutter; Universal Laser Systems) the force-sensitive film (FSF) (3M Velostat electrically conductive copolymer 0.1 mm thick; Adafruit Industries) to fit a hand. To prevent movement of the FSF during the laser cutting process, we attach the film to an acrylic board with a thin layer of water. The laser-cut film is washed to remove debris from the surface. The laser-cut pattern includes holes to route conductive thread electrodes (3-ply stainless steel conductive threads of 12 μm fibre diameter and about 0.34 mm overall diameter; SparkFun Electronics) and slots at the finger joints to allow unrestricted movement (pattern in Figure 8-1b; original design is available from the corresponding author). The conductive threads are sewn with a needle on either side of the FSF, ensuring that there are no points of direct contact between electrodes. Regions of overlap between the 64 row and column electrodes respond to forces through a change in the through-film resistance. The electrodes are first held taut and then subsequently held in place by attaching a thin, stretchable, two-sided acrylic adhesive tape (3M 468MP 200MP adhesive 0.13 mm thick) on both sides. The two exposed sides of the adhesive are insulated with a thin, stretchable polybutylene-coated LDPE film (about 13 μm; Saran; S.C. Johnson & Son). The laminate architecture is shown in Figure 8-1b and Figure 8-6b. The exposed conductive threads are coated with a polydimethylsiloxane mixture (PDMS, 1:10 ratio of crosslinker to pre-polymer; SYLGARD 184, DOW Corning). The PDMS-coated conductive thread electrodes are separately cured on a hot plate at $60°C$ for 2 h and left overnight at room temperature. The PDMS-coated conductive thread electrodes are attached to insulation-displacement connectors that can be connected to the readout circuit. Forming robust electrical connections with the conductive thread electrodes is typically challenging. Using PDMS to insulate the conductive threads and subsequently using insulation-displacement connectors resulted in robust electrical contacts, which was critical for the long-term use of the STAG. Likewise, the robustness of connections between fingers is critical to the long-term stability of the STAG. Furthermore, they are required to allow free movement of individual fingers without hindering motion and object interactions. We used a single conductive thread reaching through all the fingers for each row and insulated the region between fingers using the acrylic adhesive tape and LDPE film to form insulated ribbons routed along the sides of fingers (seen between fingers in Figure 8-4a-b).

The regular 32 $\times$ 32 array version of the sensor laminate was fabricated using a process

similar to that described above. In this case, a square of FSF was cut and assembled into a laminate in two separate versions. Figure 8-6b shows a design that is identical to the version used in the STAG. Figure 8-6a shows a simplified version of the above design by replacing the adhesive film and LDPE film with a 25.4-μm-thick polyimide film with adhesive on one side. Although it is simpler in construction, since the polyimide film is inextensible, we observed that this design (Figure 8-6a) is less flexible than the STAG laminate architecture, and is therefore better suited for use in fixed conditions. The spacing between the electrodes was set to 2.5 mm using a specially designed thread layout tool (Figure 8-6c). The layout tool consists of two pieces of acrylic, one of which has laser-etched grooves to hold the conductive threads in place with the correct spacing. The two designs in Figure 8-6a-b were used to fabricate the square 1,024-element sensor array shown in Figure 8-6d-e respectively.

Auxetic designs of the sensor (10 $\times$ 10 elements) were designed by first patterning the FSF with the design shown in Figure 8-8b. The holes allow the conducting threads to be routed (red and blue traces) to enable stretching. The cuts in the FSF allow individual sensor squares to rotate and enable extensibility in all directions. A close-up of the FSF after routeing the conductive threads is shown in Figure 8-8c. The double-sided adhesive and LDPE film (as in Figure 8-6b) are then added to both sides and the slots identical to those on the FSF are cut with a scalpel. The conductive thread electrodes are subsequently insulated with PDMS, and then connected to the insulation-displacement connector (finished design in Figure 8-8d). The auxetic sensor array can be folded, crushed and stretched in different directions as shown in Figure 8-8e-f.

8.2.2 STAG Sensor Characterization

To characterize individual sensor responses, we cut approximately one-inch squares of the FSF and attached orthogonal electrodes on either side as in the STAG architecture (inset of Figure 8-5f). The typical sensor force response was measured by applying controlled normal forces using a table-top mechanical tester (Instron 5944; Instron) and simultaneously recording the electrical resistance using a Sourcemeter (2611B Keithley Instruments). The loading rate was controlled at a specific strain rate in all tests (0.05-0.1 mm min^{-1}) until a maximum load of 0.5 N or 5 N was reached for the results shown in Figure 8-1c and Figure 8-5a-b. For the long-term tests in Figure 8-5c-d, we cycled the force between 20 mN and 0.5 N for 1,000 cycles at a controlled strain rate of 2.5 mm min^{-1}. The differential

scanning calorimetry (DSC) measurements were performed using a TA Instruments Q100 DSC (TA Instruments) at a temperature ramp rate of $10°C$ min^{-1}. The differential scanning calorimetry measurements show that the material is probably a two-polymer blend that softens at temperatures around $100°C$. FSF is considered to be thermoformable, but to directly check the feasibility of thermoforming our sensor and studying the temperature stability, we placed fabricated single-element prototypes in a convection oven at controlled temperatures for 10 min and measured the resistance through the film after removal from the oven (Figure 8-5f). The sensors show large changes in the resistance beyond $60°C$.

8.2.3 Circuit Architecture and Design

The passive matrix design of the STAG makes the fabrication simple and the design easily extensible to multiple platforms. However, immediately after fabrication, any two electrodes will appear to be electrically shorted together owing to a large number of spurious current paths; this is well known in passive arrays. A signal isolation circuit is required to overcome the extensive crosstalk between sensors [Lazzarini et al., 1995]. It is possible to eliminate a majority of crosstalk and parasitic effects using a balanced readout scheme. Here we used an improved version of an electrical-grounding-based readout architecture [D'Alessio, 1999] where one row of the sensor currently being read is grounded while all other rows are maintained at the reference voltage V_{ref} (2.5 V in our design; see Figure 8-4c-d; active row is indicated by a grey arrow). The current between all resistors in other rows is ideally 0 A since the voltage difference across all the resistors is 0 V. During this state, a 32:1 analog switch is used as an analog demultiplexer to raster through the row and read individual resistances one by one. After completing measurements at the active row, the 32 single-pole double throw (SPDT) switches are used to ground the next row while returning the currently active row to V_{ref}. We added charging resistors, R_c, to each column to charge the inverting node of each amplifier back to V_{ref} quickly and to reduce the input noise. We observed that this led to a more stable readout of the resistances without affecting the actual potential at the output of the amplifier. Note that the R_c arm of the circuit in each column is analogous to an adder circuit where one input is always 0, that is, the input to the R_c arm is V_{ref} which is the same as the voltage at the non-inverting terminal of the amplifiers. Rastering through the readout resistors in the matrix is controlled by an Arduino Nano by switching the 32 SPDT switches and the 32:1 analog switch. The single sensor measurement at the output of

the analog switch is converted to a digital signal (10-bit resolution; 0–1,023 corresponding to 0–5 V) and transmitted serially to a computer. An image of the fabricated printed circuit board is shown in Figure 8-4d, with the insulation-displacement connector cables inserted to connect the sensor array (seen at the top-right and bottom).

8.2.4 Dataset Acquisition Methods

Previous psychophysics studies on human tactile performance have used carefully designed experimental conditions (objects, awareness and tasks) that are each motivated by the purpose of the study. For instance, to demonstrate the orientation dependence of human tactile performance, previous studies have used protocols with blindfolded humans interacting with artificial objects of similar properties that are fixed in space [Newell et al., 2001]. Likewise, careful experimental designs have also been implemented in tactile interaction studies in robotics [Higy et al., 2016]. Our general objective here is to learn from successful human interactions with objects, which are typical of daily interactions. Our data acquisition methods were designed with this in mind, and motivated by a few cues from seminal human tactile perception studies of the past three decades [Lederman and Klatzky, 1987, Klatzky et al., 1985, Lederman and Klatzky, 2009]. In particular, the use of everyday objects is better as opposed to artificial objects (unless critically required for the task [Newell et al., 2001]) since human performance in object interaction and identification is underestimated when unfamiliar objects are used [Klatzky et al., 1985]. Therefore, the STAG prototype was used to record single hand manipulation of 26 different common objects with a few different sizes, weights and materials (Figure 8-9).

Visually aware and blindfolded conditions. The task setup influences the general hand movements and interactions in haptics and tactile recognition [Lederman and Klatzky, 1987, 2009]. Therefore, the level of awareness of objects during interactions is a critical part of experimental design. In this regard, human perception studies have shown that blindfolded subjects can identify common objects within 2–3 s almost perfectly; this interval increases to about 16 s when wearing a glove but with no loss in accuracy (summarized in a review [Lederman and Klatzky, 2009]). In blindfolded interactions, the level of awareness increases during these intervals, and the interactions become fully aware once the object is identified. Our large-scale dataset was captured with complete visual access to the object

(visually aware); it allows us to record human interactions with a constant level of awareness of the object at each frame. Each object was manipulated for 3–5 min at a time and included several different grasps and touch sequences. However, given that the objects are in sight, some of these grasps are more discriminative (that is, object-dependent). To test the generality of this visually aware dataset, we also performed a blindfolded study where each interaction was terminated as soon as the object was identified (at the moment the subject is aware of the object). Our original CNNs that were trained purely on the visually aware dataset were used in evaluating the blindfolded test set.

The blindfolded tests were performed by the subject (S.S.), where the task was to identify the object. In all these tests, the objects were placed on a soft foam surface to reduce sounds and the subject was made to listen to white noise through headphones. The tactile recording was started and the shoulder of the subject was tapped. The subject was asked to interact with the object, identify it and hold the object still. The tests were stopped as soon as the object was identified. Typically, the objects were identified by the subject in 6.16 ± 2.65 s. The tasks were performed continuously 104 times (4 times for each of the 26 objects) in a randomized order. One additional task was performed without any object as an empty hand control. The objects were identified correctly by the subject in 103 out of the 104 tests. The last 5 frames from each identification attempt (total 20 frames per object from 4 identification attempts) were used for a classification test using a CNN trained only on the visually aware data. We observe that the lighter objects are harder to identify more accurately using a single input frame, and the single-input ($N = 1$) top-1 classification accuracy is 28.19% (top-3 classification accuracy of 49.91%). Overall, the performance is slightly worse than the single-input classification tests of the frames from the visually aware dataset ($N = 1$, top-1 classification accuracy about 37.97% and top-3 classification accuracy about 60.43% in Figure 8-2b). The ability to identify objects using tactile frames from blindfolded tests using the original CNN trained only on the visually aware dataset demonstrates that a general set of object interactions is probably captured in the visually aware dataset.

8.2.5 Dataset Acquisition Metrics

In addition to recording interactions with 26 objects, we recorded the empty hand data while articulating the hand without interacting with any object. In all cases, we also recorded corresponding visual images from the experiment using a FLIR GS2-GE-20S4C-C camera

for illustration. Overall we recorded 135,187 frames for the visually aware dataset used for the object identification task. All experiments (visually aware and blindfolded tests) were performed by S.S. using a STAG worn on the right hand. We recorded the tactile maps from the glove along with timestamps (and corresponding visual frames) at an average frame rate of 7.3 frames per second. Each object sequence was recorded three times over different days and in a randomized order for the visually aware dataset.

A similar procedure was followed with the same set of objects in recording the dataset for the weight estimation task. To ensure that the weight of an object is not trivially associated with a particular grasp type, we standardized the grasp used in the recording to be identical for all objects. Each object was picked up from above using a multi-finger grasp (see Figure 8-11a for example images) where the weight of the object was supported by the fingers and the thumb. During each recording, the selected object was grasped, lifted, held and dropped to a flat table multiple times. The procedure was repeated in 10-s intervals for a total duration of 1 min. Each object was recorded in multiple recording sessions. In total, we recorded 11,682 frames for the weight estimation dataset.

Finally, we recorded a dataset of different articulated hand poses (empty hand and G1 to G7; Figure 8-13) based on a standard grasp taxonomy [Feix et al., 2015b] to analyse the proprioceptive content of the tactile information recorded with the STAG. We recorded each articulated hand pose in random order over 7 different recording sessions and collected 24,037 frames in total. The processing details are described in the 'Hand pose' section.

8.2.6 Object Identification

Processing and network design. The overall object identification scheme relies on the use of CNNs to extract meaningful information from tactile signals and classify objects. There are many examples of the use of CNNs with tactile sensors, especially in the context of robotics [Kappassov et al., 2015, Gao et al., 2016, Meier et al., 2016, Baishya and Bäuml, 2016]. This section describes the full details of our tactile data processing and network architecture.

The recorded tactile dataset (135,187 frames) contained useful signals in the range 500–650 (0–1,023 corresponds to 0–5 V at the amplifier output using a 10-bit ADC). The tactile map, transmitted as a 32×32 map from the readout circuit, is first normalized from 500–650 to 0–1. We discard all frames with any sensor reading over 950, which results from

shorted electrodes; this happens when the STAG is punctured by a sharp metal object, and is therefore rare and is easy to detect. We next remove frames without any useful signal when the hand is not in contact with the object. We use the empty hand recording as a reference and detect the maximum response for each sensor over time. We then consider a frame with an object to be valid if at least one sensor response is above the maximum response found in the empty hand recording. After filtering we obtain 88,269 valid frames.

To estimate the accuracy of the above frame-classification method, we manually inspected a random sequence of 150 frames from five different recordings (allen key set, multimeter, tape, mug and the foam model of a brain). In our inspection, we used the temporal sequence of both the tactile signals and the synchronized visual frames to determine the state of the grasp and to validate the thresholding-based algorithm. This allows us to notice changes in the pressure, object position and hand position, consequently making it easier to check for contact, and partially alleviates difficulties with occlusion. We observe that out of these 750 frames, there were only 57 false negatives (7.6%) where an object contact was omitted and only 3 false positives (0.4%) where a fake contact was detected. The main sources of false negatives are the weak contacts at the onset of the grasp where the pressure signal is weak; soft objects are more prone to this issue (24 for the foam model of a brain versus 1 for the mug). The main sources of false positives are the occasionally pronounced hand poses. This issue is rare because we threshold our data based on the empty-hand dataset, which covers a large range of possible hand movements. Overall, our automatic frame classification method is conservative and presents clean data.

We then split the dataset into training and test subsets; out of the three sets of trials in the visually aware dataset, we used two sets of trials for training, and one set of trials for testing. Each subset was randomly subsampled to contain a balanced number of valid frames for each of the object classes (26 objects and empty hand recording). The training set has a total of 36,531 frames (1,353 per class), and the test set has 16,119 frames (597 per class).

With the aim of predicting an object's identity based on its tactile signature, we treat each recording trial of a given object as a single instance of the problem, which simulates an agent exploring an object using multiple grasps. We feed N = 1...8 frames from the recording to the deep neural network to accommodate information from different grasp configurations and provide more varied sensory data. When we use $N > 1$ input frames to our network during evaluation we consider two different strategies for their selection. The first one is a

simple random choice of N frames from the recording. The second is aimed at minimizing the redundancy of data between the N frames by maximizing their variance. For this, we use principal component analysis to reduce the dimensionality of the tactile signal to 8. We then find N clusters via k-means clustering (Figure 8-2e); that is, we complement every randomly selected input frame with N − 1 other frames, each of which belongs to a different cluster.

We use a modified version of the ResNet-18 architecture [He et al., 2016] as the base of our network (Figure 8-2a). We reduce the filter size of the initial convolution layer from 7 to 3 and the stride from 2 to 1. This allows the entire filter to fit within the smallest features in our sensor data (finger width is 3 pixels). Since our inputs are 32 × 32 pixels, we remove the upper two of the four ResNet layer groups leaving the final feature vector size to be 128 values. To reduce overfitting to our training, set we introduce a spatial dropout layer [Tompson et al., 2015] with 20% drop probability between the two remaining block layers. Additionally, we also augment our training data by additive Gaussian noise with zero mean and a standard deviation of 0.015 during training To incorporate multiple input frames, we apply the same network with shared weights to each input. The outputs of all network branches are then concatenated and reduced to 128 dimensions by a per-pixel convolution. After spatial averaging, a final fully connected layer computes the classification vector. We implemented this network in the PyTorch (`https://pytorch.org/`) deep learning framework [Paszke et al., 2019]. We use Adam solver implemented in PyTorch to train our model and minimize the cross-entropy loss. We apply an initial learning rate of 10^{-3}, which we decrease by a factor of 10 every 100 epochs. We train the network using our training dataset for 200 epochs with batches of 32 samples. We report the average results over 10 training runs. Similar methods were used for the training and classification tests used to evaluate different sensor resolutions. In all cases, the tactile data was downsampled (by averaging) and resized to an input size of 32 × 32 in order to use the same network. These inputs were used for training and subsequent classification tests using methods identical to those described above.

The 3 × 3 filter of the first convolution layer is not easy to interpret visually owing to the low resolution. Therefore, we trained a scaled version of the model with the first convolution increased to 9 × 9 and the stride to 3, for visualization. Max pooling layers reduce the dimensionality of the model by spatially selecting the locally highest activation values and are used in both versions of the model. We adapted the filter size of the first max pooling

layer to 7 and its stride to 4 to make the resulting features similar to the original model. We also scaled the resolution of the inputs identically (by 3) using bilinear upsampling. The remainder of the model and the training procedure stayed the same. The learned convolution filters are shown in Figure 8-2d; the resulting features and performance are similar to the original model. See Figure 8-10i for the corresponding 3×3 convolution filters learned by the original network. The first convolution filters of the standard ResNet-18 architecture pre-trained on ImageNet [Russakovsky et al., 2015] are shown in Figure 8-10j for comparison.

To further understand the internal representation of the CNN, we perform Network Dissection [Bau et al., 2017], a widely used technique for analysing the deeper layers of a network. Specifically, for any convolution filter, the method first ranks the tactile maps according to the highest activation value. We subsequently select the tactile frames that rank first with the highest activations from each object category. This, in essence, shows the top candidates that different convolution filters have learned to look out for. We observe that the convolution filters in the first of the two ResNet blocks are most activated in smaller, spatially confined regions around the tactile signal peaks. However, interesting grasp-related concepts emerge in the second ResNet block. For instance, these filters are activated by different object patterns or by specific regions of the hand such as the palm or the thumb.

8.2.7 Weight Prediction

Dataset. We performed identical pre-processing of frames as for the classification task to obtain normalized 32×32 tactile maps. The collection sequence of the grasping procedure for weight estimation was predetermined; we used the frames between 4 s and 6 s of each sequence when the object was held (2,301 frames). To prevent the problem from being reduced to a simple classification task where each weight is associated with an object, we used a 'leave-one-out' approach where the object whose weight is predicted is not part of the training data.

Regression. Our goal is to predict weights (in grams) for a previously unseen object based on a single tactile frame; unlike object recognition, weight estimation is possible as soon as an object is held. Here we use a similar ResNet-based network architecture. Because this task is based on a single frame input, we remove the input branch concatenation (retaining a single branch of the CNN alone) and directly apply a fully connected layer to regress the

weight. Owing to the wide range of object weights in our dataset, we perform the prediction in the logarithmic space. We optimize the parameters of our model by minimizing the mean squared error between the predicted and measured ground truth weights (in logarithmic space). We used the same optimizer and learning parameters as before and train the network for ten epochs.

We construct a linear baseline for the weight estimation. This is motivated by the fact that the weight of an object is directly linked to the sum of all forces it applies when the object rests on a horizontal surface (that is, the naive estimate of weight is $aX + b$, where X is the sum of the tactile map). This baseline is not valid in practice, however, because the tactile response is affected by the articulation of the hand (grasp), as well as by the surface friction and the additional force used to avoid slipping. We used the same 'leave-one-out' training procedure to find the best choice of a and b that minimizes the mean squared error between the prediction and ground-truth weight in logarithmic space.

Figure 8-2f compares the results of our network with the linear baseline model. The error for each object corresponds to a different instance of our model trained with that particular object left out from the training dataset. We also computed a mean predicted error across all objects expressed in grams in linear space. We found that the average prediction error of our model is 56.88 g, which is less than the 89.68 g of the naive linear model. This result shows that nonlinear behaviours connected with grasps and the physical properties of the objects cannot be omitted and that the neural network can compensate these to some extent. The weight estimation errors in different weight ranges are listed in the table in Figure 8-11b. We observe that the CNN outperforms the linear baseline in all cases. Furthermore, the estimation errors are also listed as relative errors (normalized by object weights); this is analogous to the Weber fraction used in the tactile weight perception literature [Brodie and Ross, 1984]. We also tested an additional modification to the linear baseline by removing the hand pose component from the tactile signal using the methods outlined in the section 'Decomposing signal and sensor correlations' and Figure 8-3a. We observed that the modified linear baseline did not present any noticeable improvement over the naive baseline approach, and the performance of the CNN was better than both linear methods. We believe that this is due to the complex relationship between the tactile signals and the weight estimates and this is borne out by previous weight perception studies [Flanagan and Bandomir, 2000].

It is noteworthy that humans rely both on cutaneous and kinaesthetic senses (and their

inertial responses during object interactions) to gauge weight effectively. The Weber fraction is known to be about 1/3 when using the cutaneous sensors [Brodie and Ross, 1984]. Our estimated metric analogous to the Weber fraction is similar for moderately heavy to heavy objects (Figure 8-11b); this performance is comparably good considering that the human hands possess additional types of mechanoreceptors that are not used here.

8.2.8 Hand Pose

Dataset. To evaluate whether hand pose (proprioceptive) information can be retrieved from the glove even when no objects are being manipulated, we picked seven distinct grasps from the grasp taxonomy [Feix et al., 2015b] (along with a neutral hand pose for reference). We chose these specific grasps because they are often used and involved object interactions at the palmar side of the hand, which is covered with sensors in the STAG. Furthermore, we articulate each hand pose back and forth from the neutral hand pose; this covers a larger set of grasps in the taxonomy and increases the intra-class variance. We removed ambiguous articulations of the hand (close to the neutral pose) by thresholding. Frames with a mean pressure signal higher than 75% of a local dynamic range in a symmetric window with a 6-s temporal radius were considered as reliably valid hand poses. This filtering step reduced the number of frames to 7,697.

t-SNE embedding. We applied t-SNE [Van der Maaten and Hinton, 2008] to the tactile frames to discover structure in the signal and to evaluate its dependency on the performed hand pose. We first reduce the dimensionality of our pressure readings from the original 548 active sensors to 50 using principal component analysis and then applied t-SNE to obtain the final two-dimensional projection presented in Figure 8-13. The network described in the 'Object identification' section was also used to train and classify the hand poses from single tactile maps with 89.4% accuracy (average of 10 runs; 3,080 training frames and 1,256 distinct test frames). The confusion matrix corresponding to these tests is shown in Figure 8-13c. Most grasps can be identified correctly except for G1 and G6, which are occasionally confused.

8.2.9 Decomposing Signal and Sensor Correlations

We computed the mutual correlation of the sensors over our entire object identification and classification dataset. Each recorded frame contains two different signals: the hand pose signal and the object-related pressure due to forces between an object and the hand. The hand pose signal denotes the articulation of the hand and would be present even in the absence of an object. We assume the hand pose signal to be saturated once the hand reaches full articulation just before contact (reach phase in Figure 8-3a). This simplification holds in a majority of cases where the hand articulation does not change much after initial contact. In this condition, the additional force due to an object contact can be superimposed on the articulated hand. By design, the sensors respond to normal compressive force components between the upper and the lower electrodes. Therefore, the force components picked up by the sensors can be treated as additive. Furthermore, the output of the sensors as seen at the output of the amplifier circuit or the ADC is linear with respect to the applied force in the working range.

To extract the hand pose signal and the object pressure signal from the tactile frame, we implement the following decomposition procedure. We first compute the mean pressure response for every frame of each recording and filter it over time with a symmetric Gaussian kernel with the three-frame standard deviation (around 400 ms) to remove noise. The gradient of this signal (as a forward difference) can show possible object contact points as local maxima of the gradient (symmetrical window of five frames; about 700 ms). Considering that the presence of two distinct contacts in this window is unlikely, we first locate the nearest preceding minima (up to 20 frames away) of the filtered mean pressure signal. To ensure that this minimum is reliable and to avoid detecting random pressure fluctuations, we require the mean pressure value to be below 20% of the recording dynamic range. Similarly, we find the next local maxima and ensure it is larger than 20% of the dynamic range of the recording. The minimum and maximum are marked as the empty hand pose frame (blue dot in the reach phase; Figure 8-3a) and as the object manipulation frame (green dot in the lift and hold phase). To recover a larger amount of useful data, we further explore the frames surrounding the local maximum frame and include every frame up to a symmetrical 80-frame radius to the object pressure set until a frame that does not appear to be valid; here, valid frames are those that lie above the detection threshold (0.5) in this normalized scale. The

empty hand pose was subtracted from each of the object manipulation frames to recover the pure object-related tactile data.

To evaluate the effectiveness of this decomposition process, we manually analysed the first 60 s of recordings with five random objects of different sizes and weights (full cola can, mug, multimeter, pen, and tape). We observe that 64 grasps were detected out of the 93 while only producing 4 false positive detections. Typically, more detections were missed for light objects compared to heavier objects (for example, 12 for the tape versus 2 for the full cola can). Furthermore, the location of the before-contact point (blue) was on average 1.7 frames before the true location. The first detected object-contact point (first of the green frames) was on average 3.5 frames after the true location. Our algorithm is therefore conservative as it does not produce data from false locations and places the empty hand sample safely in the pre-contact space, as well as uses reliable post-contact frames as a source for extracting the object signal.

Pearson correlation coefficient and canonical-correlation analysis were used to analyse the correlations between sensors and sensor groups separately on the decomposed hand pose data and the decomposed object-related tactile data. Pearson correlations coefficients based on single sensor correlations uncover local relations between neighbouring sensors but are sensitive to the selection of specific sensors and do not present overall trends. To extract phalange level correlations, we used canonical-correlation analysis; results are shown in a circular plot [Krzywinski et al., 2009] in Figure 8-3c. The three finger phalanges, starting at the palm and going outward, are proximal, intermediate and distal. Note that the thumb does not have an intermediate phalanx by standard convention. Since canonical-correlation analysis always results in larger correlations than those achieved at the individual sensor level, we show the differences in the relations by subtracting the minimum correlation; the intermediate phalanx of the little finger and the proximal phalanx of the index finger show minimum correlation.

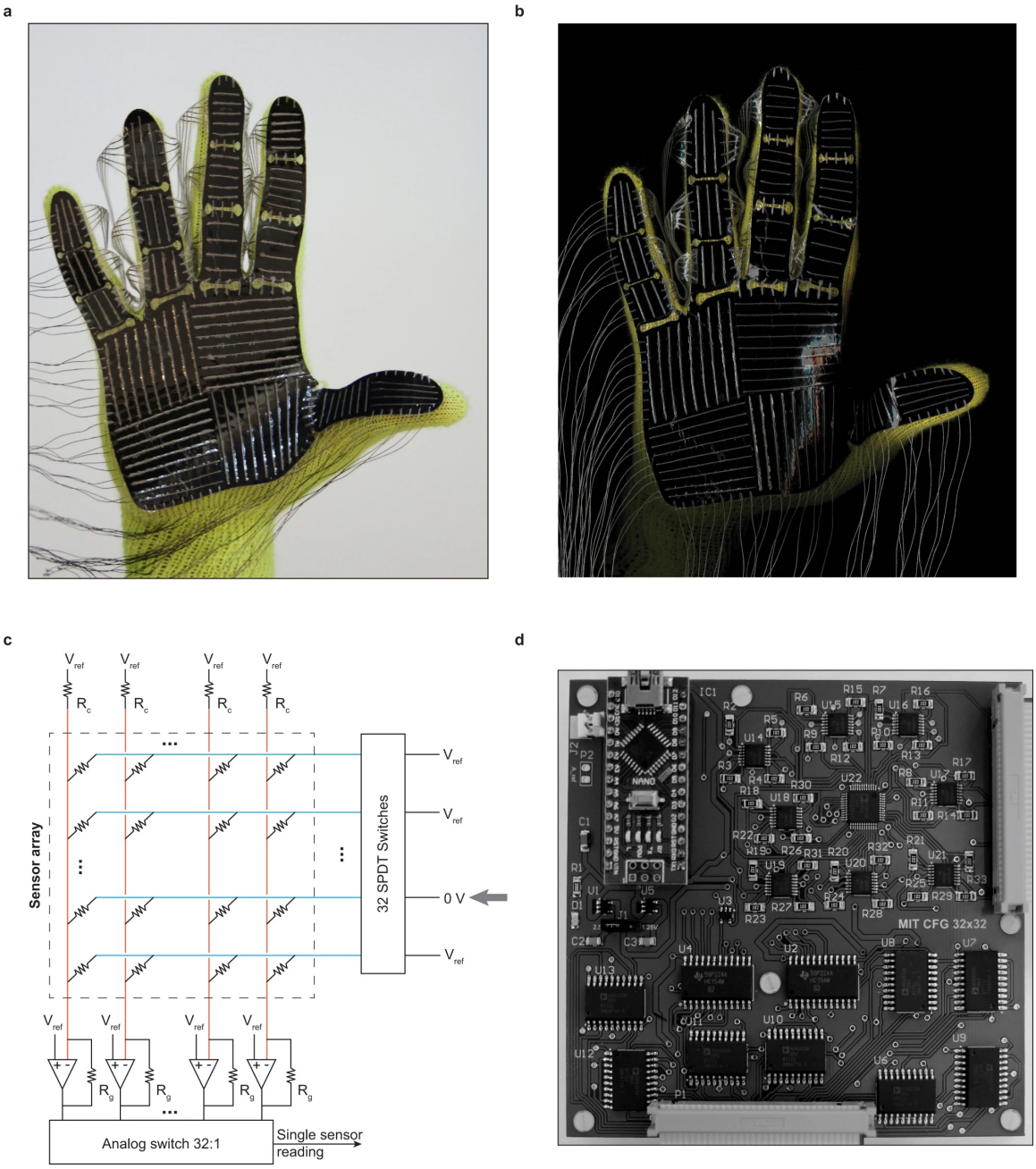


Figure 8-4: **STAG images and readout circuit architecture.** (a) Image of the finished STAG just before the electrodes are insulated. (b) Scan of the STAG. (c) Electrical-grounding-based signal isolation circuit (based on D'Alessio [1999]). The active row during readout is selected by grounding one of the 32 single-pole double throw (SPDT) switches. A 32:1 analog switch is used to select one of the 32 columns at a time. Here R_c is the charging resistor, V_{ref} is the reference voltage, and R_g sets the amplifier gain. (d) Fabricated printed circuit board that interfaces with the STAG. The two connectors shown on the top right and bottom are connected to the column and row electrodes of the sensor matrix. The charging resistors (R_c) are on the back of the printed circuit board.

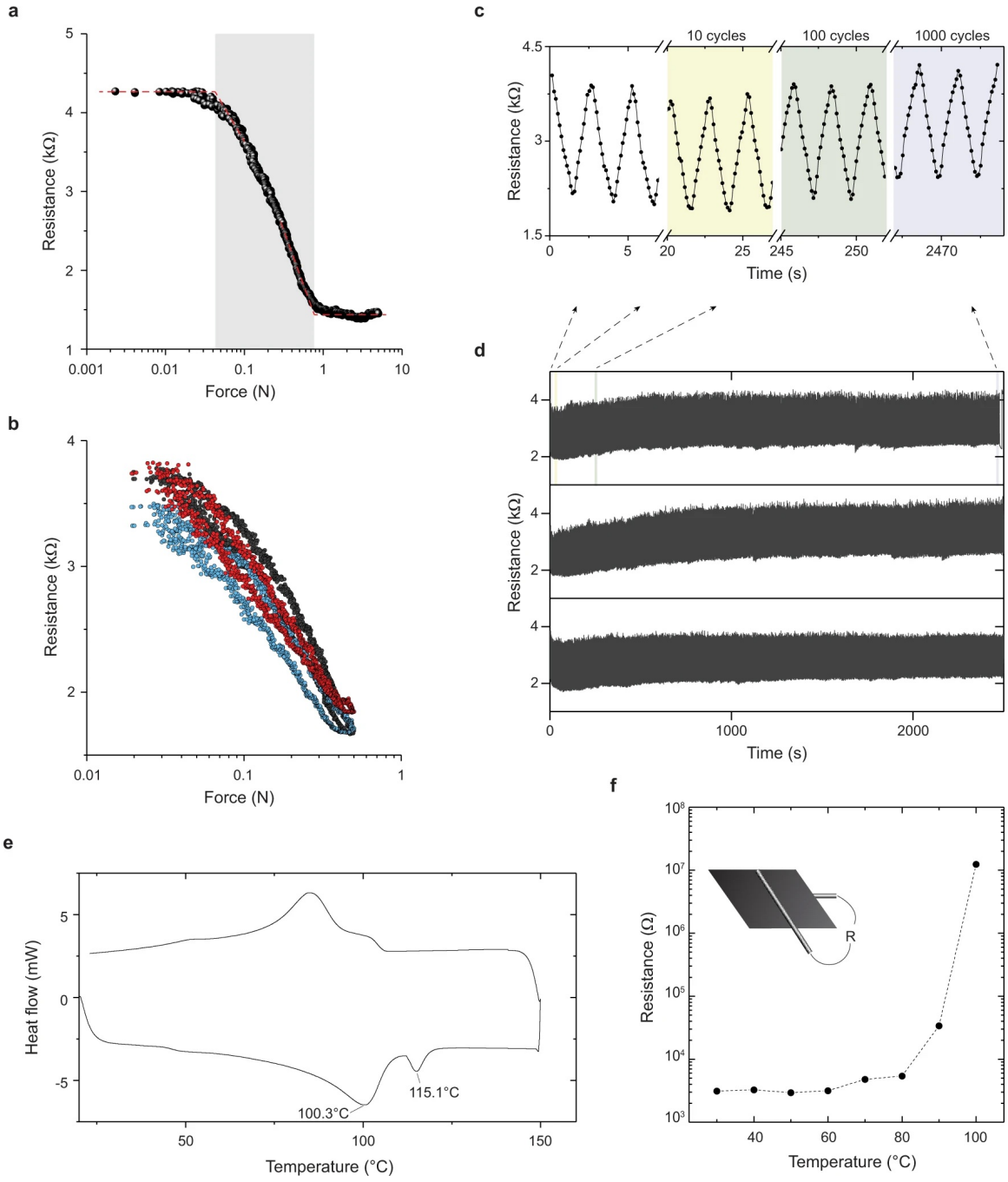


Figure 8-5: **Characteristics of the STAG sensing elements.** (a) The resistance of a single sensing element shows the linear working range (in logarithmic force units). The sensor is not sensitive below about 20 mN of force and saturates in response when a load exceeding 0.8 N is applied. (b) Response of three separate sensors in the force range 20 mN to 0.5 N. The sensors show minimal hysteresis (17.5 ± 2.8%). (c) The sensor response after 10,100 and 1,000 cycles of linear force ramps up to 0.5 N for three separate devices. The resistance measurements are shown in (d) over the entire set of cycles. (e) Differential scanning calorimetry measurements of the FSF material shows a two-polymer blend response with softening/melting temperatures of around $100°C$ and $115.1°C$. (f) Through-film resistance of an unloaded sensor after treating at different temperatures in a convection oven for 10 min. The film becomes insulating above about $80°C$.

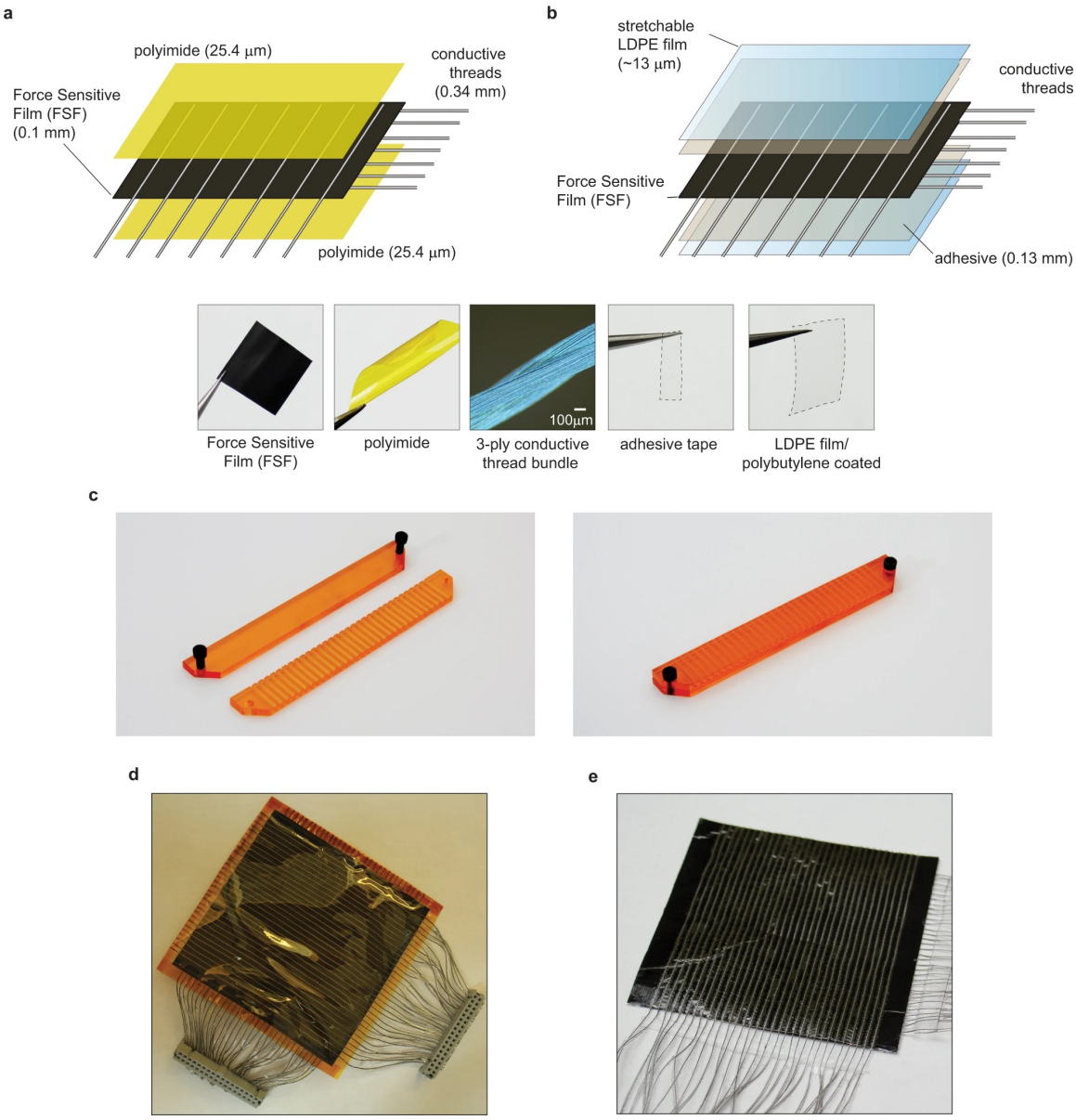


Figure 8-6: **Sensor architectures and regular 32 × 32 arrays.** (a) A simplified version of the sensor laminate architecture. (b) The sensor is assembled by laminating a FSF along with orthogonal electrodes on each side, that are held in place and insulated by a layer of two-sided adhesive and a stretchable LDPE film (see Section 8.2). (c) Fixture used to assemble parallel electrodes. The individual electrodes can be threaded into the structure (like a needle) for assembling parallel electrodes with a spacing of 2.5 mm. (d) Assembled version of the architecture shown in (a). (e) A regular 32 × 32 array version of the STAG based on the design in (b).

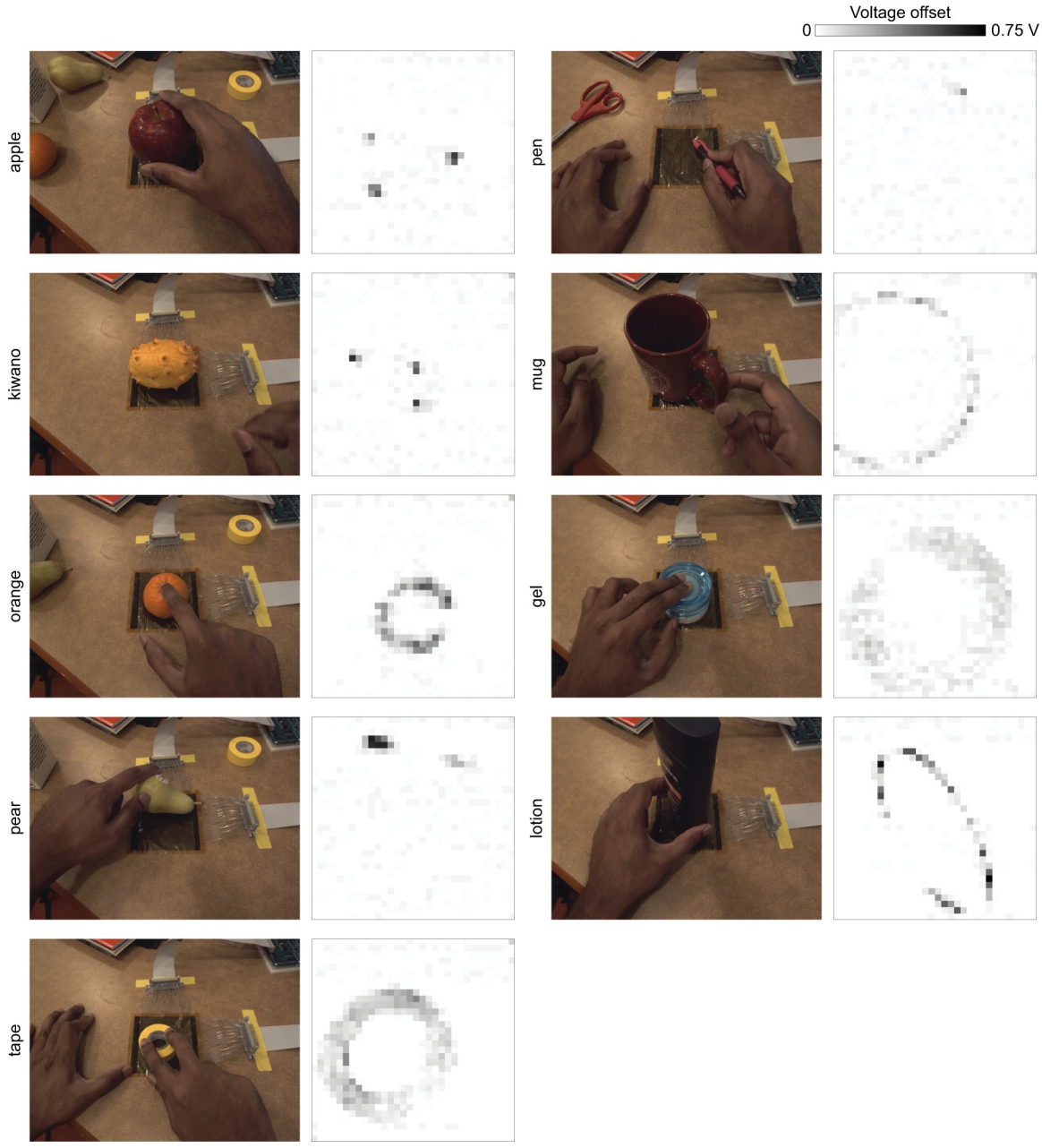


Figure 8-7: **Sample recordings of nine objects on regular 32 × 32 arrays on a flat surface.** Nine different objects are manipulated on a regular sensor array (Figure 8-6d) placed on a flat surface. The resting patterns of these objects can be seen easily. Pressing the tactile array with sharp objects like a pen or the needles of a kiwano yields signals with a single sensor resolution.

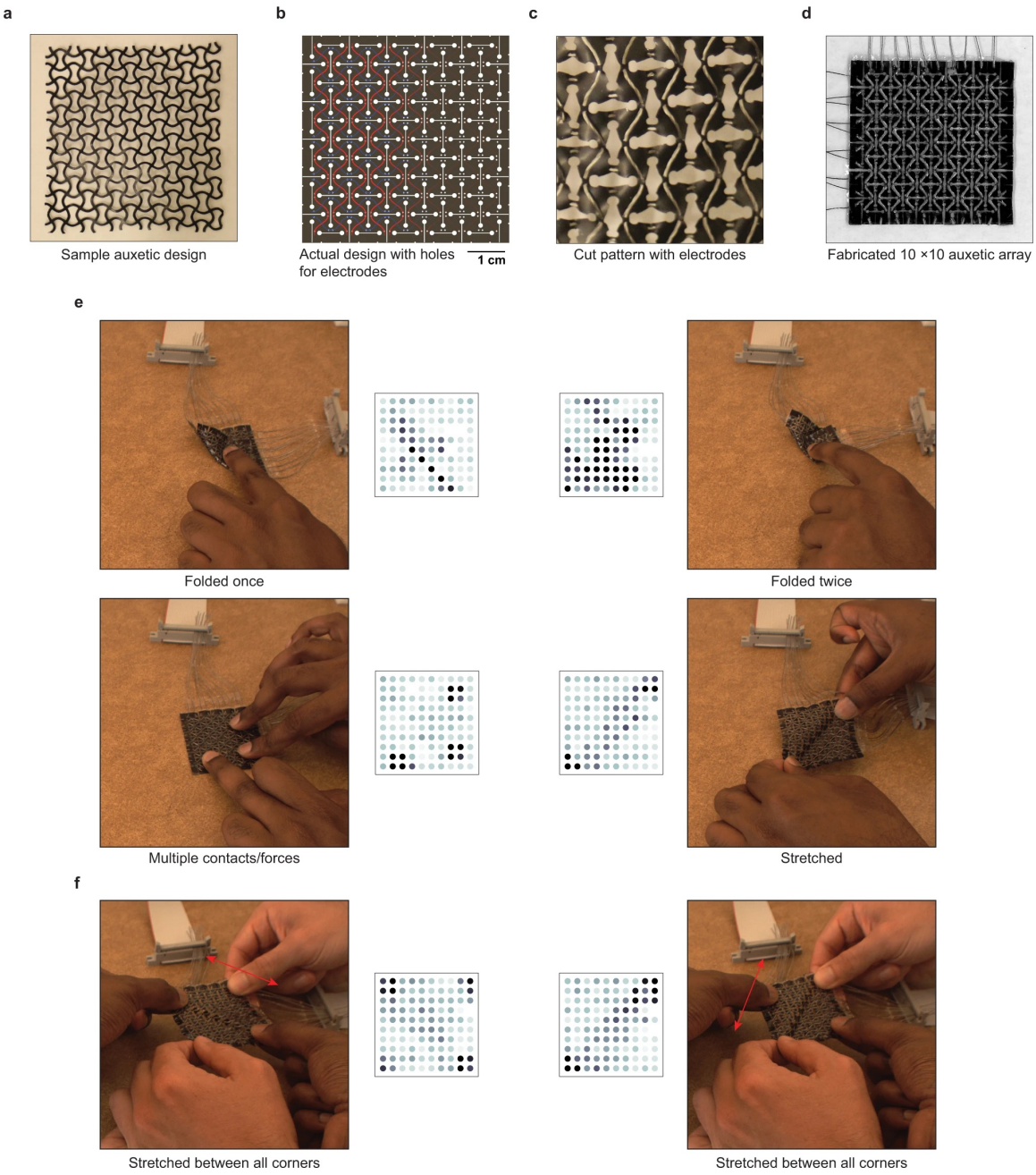


Figure 8-8: **Auxetic designs for stretchable sensor arrays.** (a) Standard auxetic design laser cut from the FSF. (b) The actual design of the auxetic includes holes to route the electrodes (shown in red and blue), and slots allow the square, sensing island to rotate, enhancing the stretchability of the sensor array. (c) Close-up of the fabricated array showing the conductive thread electrodes before insulation. (d) A fully fabricated 10 × 10 array with an auxetic design. (e) Auxetic patterning allows the sensor array to be folded, crushed and stretched easily with no damage. (f) The array can also be stretched in multiple directions.

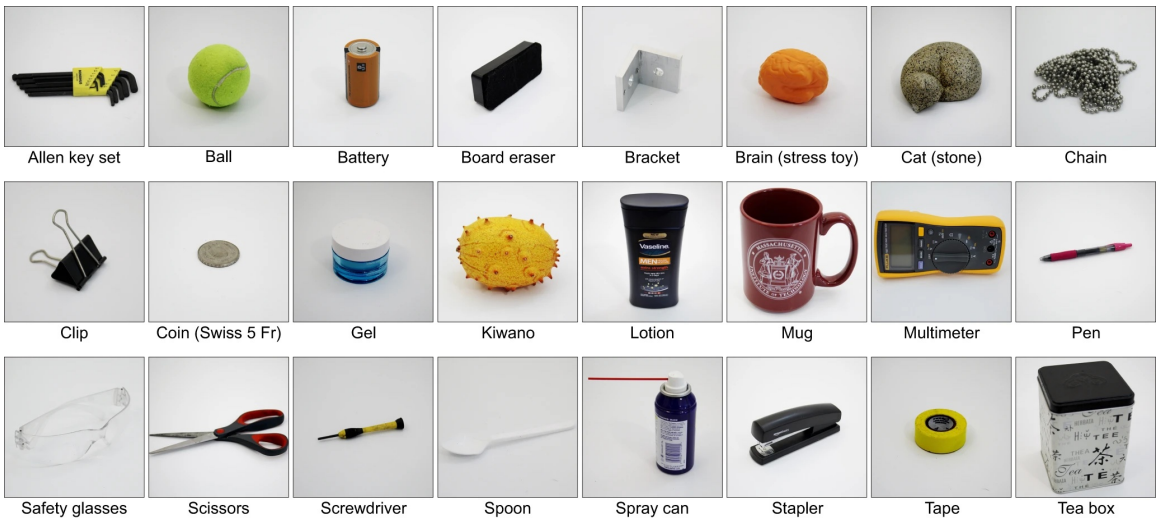


Figure 8-9: **Dataset objects.** In total, 26 objects are used in our dataset; images of 24 objects are shown here. In addition to these objects, our dataset includes two cola cans (one empty can and one full can).

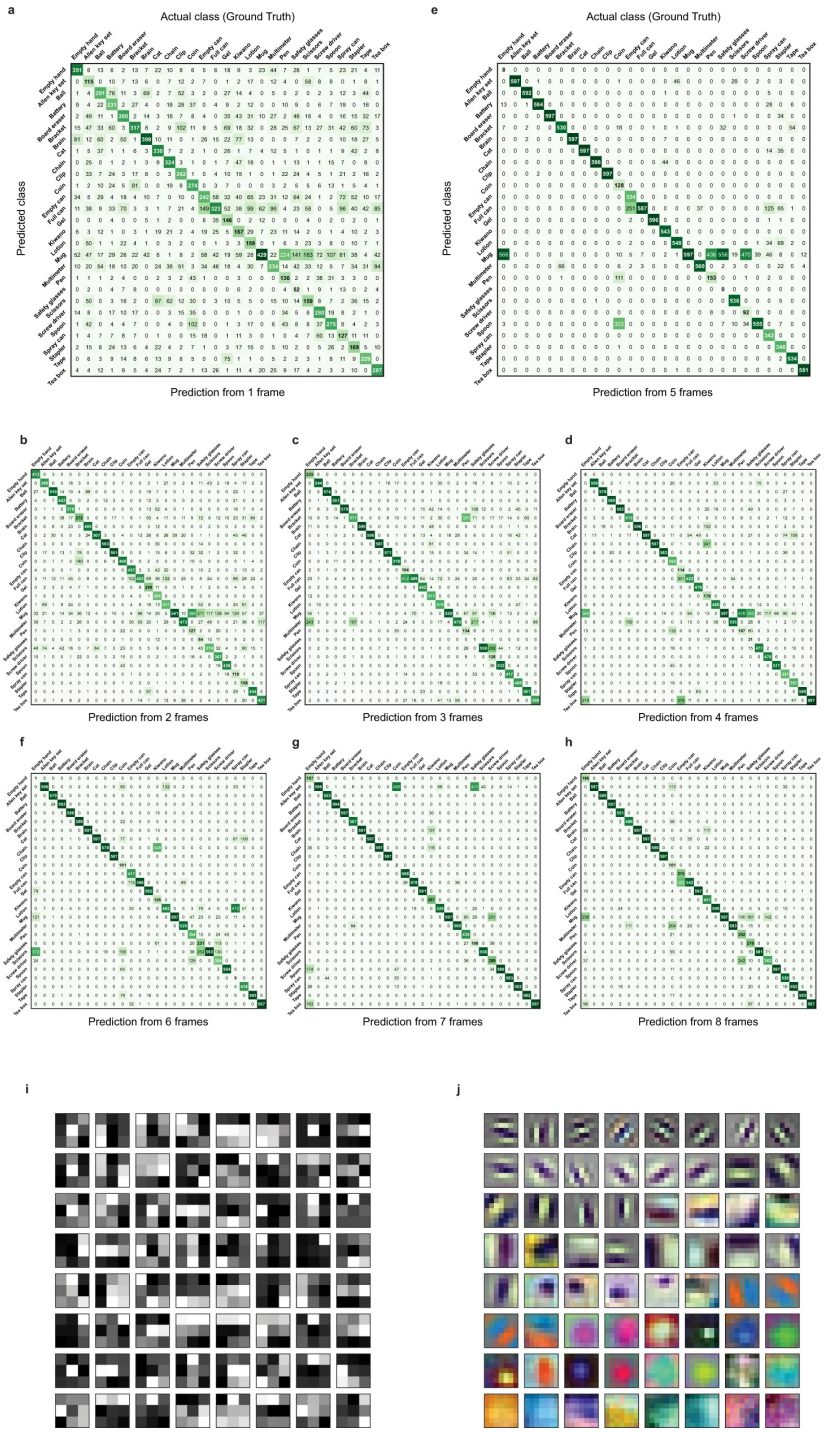
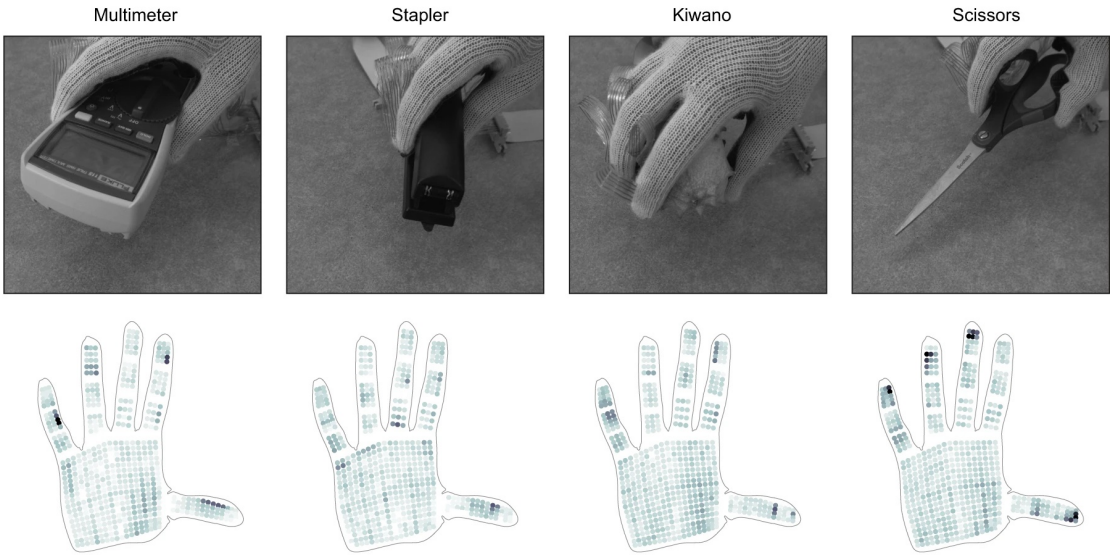


Figure 8-10: **Confusion maps and learned convolution filters.** (a–h) The actual object and predicted object labels are shown in these confusion matrices for different networks, each taking 1 to 8 (or N) inputs where each input is obtained from a distinct cluster for N > 1 (approach shown in Figure 8-2e; see Section 8.2). These matrices correspond to the 'clustering' curve in Figure 8-2b. Objects with similar shapes, sizes or weights are more likely to be confused with each other. For example, the empty can and full can are easily mistaken for each other when they are resting on the table. Likewise, lighter objects such as the safety glasses, plastic spoon, or the coin are more likely to be confused with each other or other objects. Large, heavy objects with distinct signatures such as the tea box have high detection accuracy across different numbers of inputs (N). (i) Original first-layer convolution filters (3×3) learned by the network shown in Figure 8-2a for N = 1 inputs. (j) Visualization of the first-layer convolution filters of ResNet-18 trained on ImageNet.

a



b

Weight range (g)	Errors					
	Linear baseline		Linear baseline (Hand pose removed)		CNN	
	Abs. (g)	Rel. (Weber)	Abs. (g)	Rel. (Weber)	Abs. (g)	Rel. (Weber)
< 30	57.09	4.63	65.73	5.33	16.49	1.34
31 - 150	83.36	1.07	79.82	1.02	53.30	0.68
151 - 700	136.75	0.45	144.30	0.47	110.97	0.36
Overall error	89.68	2.37	94.24	2.52	56.88	0.69

Figure 8-11: **Weight estimation examples and performance.** (a) Four representative examples from the weight estimation dataset, in which the objects are lifted using multi-finger grasps from the top. (b) The weight estimation performance is shown in terms of the mean absolute and relative errors (normalized to the weight of each object) in each weight interval. The relative error is analogous to the Weber fraction. We observe that the CNN outperforms the linear baseline with or without the hand pose signal removed. The overall errors of the two linear baselines are comparable.

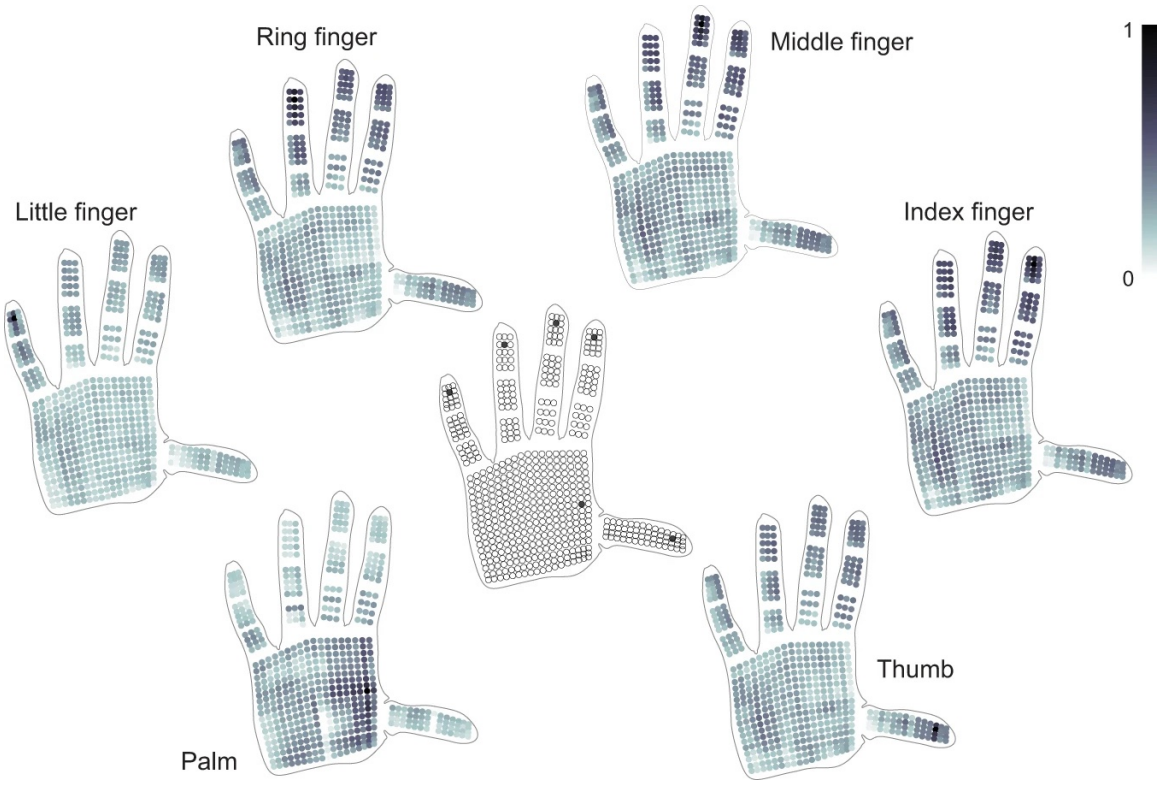


Figure 8-12: **Correspondence maps for six individual sensors using the decomposed hand pose signal.** The hand pose signal decomposed from object interactions is used to collectively extract correlations between the sensors and the full hand (analogous to Figure 8-3b where the decomposed object-related signal is used). The pixels at the fingertips show less structured correlations with the remaining fingers, unlike in Figure 8-3b.

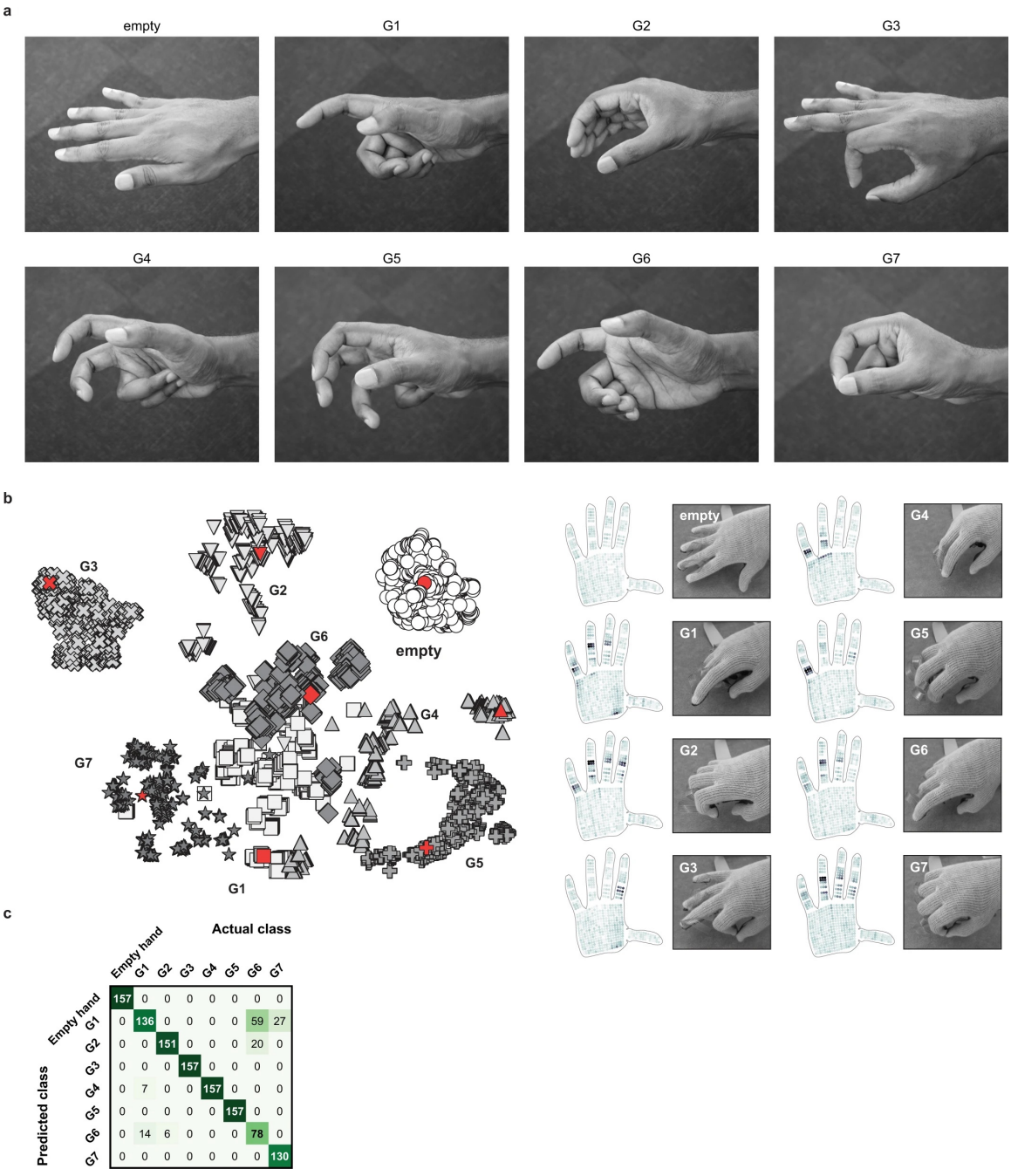


Figure 8-13: **Hand pose signals from articulated hands.** (a) Images of the hand poses used in the hand pose dataset. The poses G1 to G7 are extracted from a recent grasp taxonomy. In the recordings, each pose is continuously articulated from the neutral empty hand pose. (b) When the tactile data from this dataset is clustered using t-SNE, each distinct group represents a hand pose. Sample tactile maps are shown on the right. The corresponding samples are marked in red. (c) The hand pose signals can be classified with 89.4% accuracy (average of ten runs with 3,080 training frames and 1,256 distinct test frames) using the same CNN architecture shown in Figure 8-2a. The confusion matrix elements denote how often each hand pose (column) is classified as one of the possible hand poses (rows). It shows that hand poses G1 and G6 are sometimes misidentified but the other hand poses are identified nearly perfectly.

Chapter 9

Learning Human-Environment Interactions Using Tactile Textiles

The previous chapter shows that the dense tactile glove is capable of learning the contacting patterns during hand-object interactions and recognizing and weighing objects. However, the sensors only cover the palm area of the hand, whereas for humans, we have tactile sensing capabilities all over our skin, covering much larger surface areas of more complicated geometries. Therefore, in this chapter, we take on the problem of developing more flexible and large-scale tactile sensing platforms for more complicated 3D geometries in the form of wearable textiles. Sensors of this kind would be of great value in modeling and understanding tactile interactions, facilitating the study of human behavior and the development of applications in healthcare and robotics. However, such studies remain challenging because existing wearable sensory interfaces are limited in terms of performance, flexibility, scalability, and cost.

Here, we report a textile-based tactile learning platform that can be used to record, monitor, and learn human-environment interactions. The tactile textiles are created via digital machine knitting of inexpensive piezoresistive fibers and can conform to arbitrary three-dimensional geometries. To ensure that our system is robust against variations in individual sensors, we use machine learning techniques for sensing correction and calibration. Using the platform, we capture diverse human-environment interactions (more than a million tactile frames) and show that artificial-intelligence-powered sensing textiles can classify humans' sitting poses, motions, and other interactions with the environment. We also

show that the platform can recover dynamic whole-body poses, reveal environmental spatial information, and discover biomechanical signatures. Please check the video illustrations of developed sensors on our project page: `http://senstextile.csail.mit.edu/`.

This chapter was originally published as Luo et al. [2021b] in *Nature Electronics*, along with co-authors Yiyue Luo, Pratyusha Sharma, Wan Shou, Kui Wu, Michael Foshey, Beichen Li, Tomas Palacios, Antonio Torralba, and Wojciech Matusik, in which Yiyue Luo have made major contributions to the content of this chapter.

9.1 Main

Living organisms extract information and learn from the surroundings through constant physical interactions [Dahiya et al., 2009]. Humans are particularly receptive to tactile cues (on hands, limbs and torso), which allow complex tasks such as dexterous grasp and locomotion to be carried out [Winter, 1995]. Observing and modelling interactions between humans and the physical world are thus fundamental to the study of human behaviour [Someya et al., 2016], and also for the development of applications in healthcare [Yousef et al., 2011], robotics [Yang et al., 2018, Sundaram et al., 2019] and human–computer interactions [Poupyrev et al., 2016]. However, studies of human–environment interactions typically rely on easily observable visual or audible datasets [Zeng et al., 2014], and obtaining tactile data in a scalable manner remains difficult.

Wearable electronics have benefited from innovations in advanced materials [Yan et al., 2019, Rogers et al., 2010, Engler et al., 2006], designs [Moeslund et al., 2006, Rein et al., 2018, Kim et al., 2008, Wang et al., 2018] and manufacturing techniques [Ahn et al., 2009, Truby and Lewis, 2016]. However, sensory interfaces that offer conformal, full-body coverage and can record and analyse whole-body interactions have not been developed. Such full-sized tactile sensing garments could be used to equip humanoid robots with electronic skin for physical human–robot collaboration [Winter, 1995, Sundaram et al., 2019], could serve as auxiliary training devices for athletes by providing real-time interactive feedback and recording [Adams, 1971], and could assist high-risk individuals, such as the elderly, in emergencies (a sudden fall, for example) and early disease detection (heart attacks or Parkinson's disease, for example) [Wang et al., 2017b] by acting as unobtrusive health-monitoring systems.

In this chapter, we report a full-body tactile textile for the study of human activities

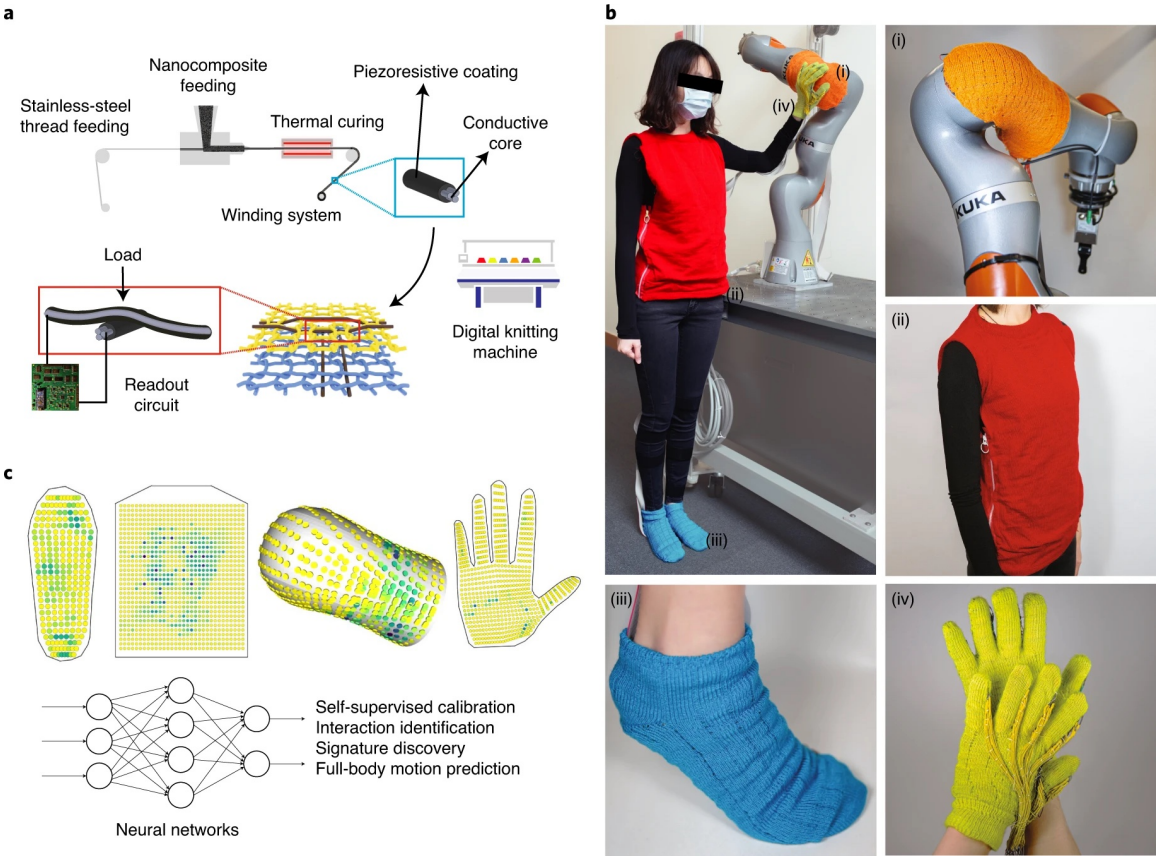


Figure 9-1: **Textile-based tactile learning platform.** (a) Schematic of the scalable manufacturing of tactile sensing textiles using a customized coaxial piezoresistive fibre fabrication system and digital machine knitting. A commercial conductive stainless-steel thread is coated with a piezoresistive nanocomposite (composed of polydimethylsiloxane (PDMS) elastomer as the matrix and graphite/copper nanoparticles as the conductive filler). (b) Digitally designed and automatically knitted full-sized tactile sensing wearables: (i) artificial robot skin, (ii) vest, (iii) sock and (iv) glove. (c) Examples of tactile frames collected during human–environment interactions and their applications explored using machine learning techniques.

(Figure 9-1). Our textiles are based on coaxial piezoresistive fibres (conductive stainless-steel threads coated with a piezoresistive nanocomposite), produced using an automated coating technique. The functional fibres can be turned into large-scale sensing textiles that can conform to arbitrary three-dimensional (3D) geometries, using digital machine knitting, an approach that addresses current challenges in the large-scale manufacturing of functional wearables. An artificial intelligence (AI)-based computational workflow is also developed here to assist in the calibration, recording and analysis of full-body human–environment interactions using the tactile textiles.

9.2 Conformal Tactile Sensing Textiles

Our low-cost (∼US$0.2 per metre) coaxial piezoresistive fibres are created using a scalable automated fabrication process (Figure 9-1a; further details on characterization and preparation are provided in Figure 9-2a–c and the Section 9.5). A pair of fibres are then orthogonally overlapped to create a sensing unit, which converts pressure stimuli (a normal force acting on the surface) into electrical signals [Dahiya et al., 2009, Zeng et al., 2014]. Figure 9-2d shows that the measured resistance of a typical sensor drops from 8 kΩ to 2 kΩ in response to an increasing applied normal force (or pressure) in the range of 0.1-2 N (with an average peak hysteresis of -22%). Our functional fibre has reliable performance over 1,000 load and unload cycles (Figure 9-2f). It also maintains stable performance in a wide range of daily environments, such as different temperatures (20–40°C) and humidities (40–60%). The performance of the sensing unit can be tuned on demand by adjusting the material compositions (copper and graphite weight percentages) and fabrication process (pulling speed and material feeding rate).

To fabricate 3D conformal tactile textiles in a scalable manner, we use digital machine knitting to seamlessly integrate the functional fibres into shaped fabrics and full-sized garments. Although weaving has been widely attempted for inserting functional fibres into fabric, knitting is chosen here as it has two primary advantages over weaving. First, the fabrication process is simpler: whereas woven fabric must be cut and sewn to form a garment, full-garment machine knitting can directly manufacture wearables with arbitrary 3D geometry [Narayanan et al., 2019]. Second, the interlocking loops of yarn (that is, stitches) used in knitting create a softer, stretchier fabric, which ensures comfort and compatibility during natural human motions.

Because of its relative stiffness compared to regular knitting yarn, the piezoresistive functional fibre is not suitable to be knitted directly by forming loops. We thus used an alternative knitting technique—inlay—which horizontally integrates the yarn in a straight configuration. We assembled two knitted fabrics with functional fibres inlaid perpendicularly to form a sensing matrix. Different knitting patterns can be used to tune the performance of the sensing matrix for customized applications. The highest achieved sensitivity of 1.75 kPa and a detection range up to 87.5 kPa are demonstrated in Figure 9-2e.

We computationally designed and automatically knitted several example garments: socks

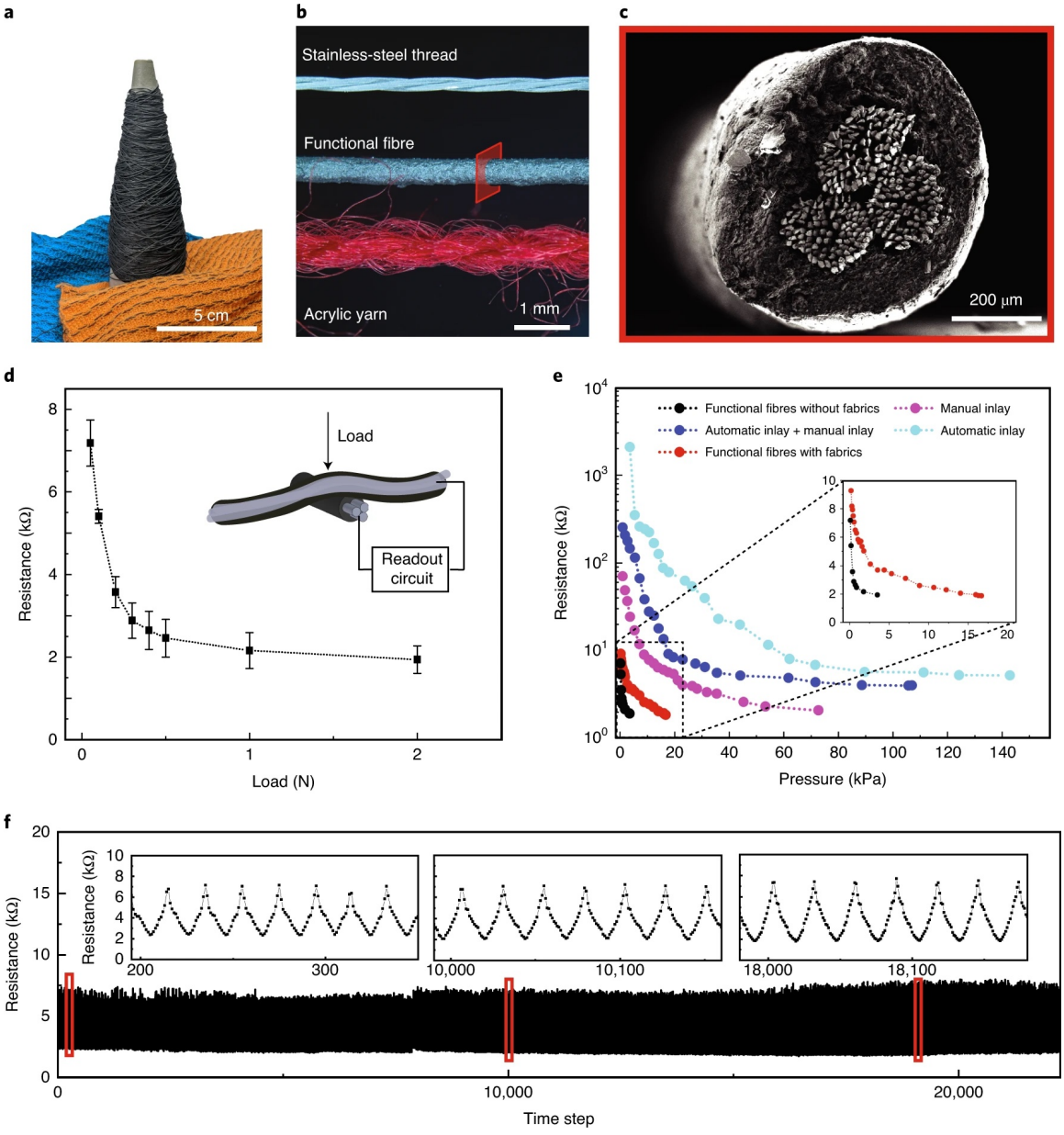


Figure 9-2: **Characterization of the functional fibre.** (a) Photograph of the piezoresistive functional fibre (>100m) and sensing fabrics. (b) Optical image of a stainless-steel thread, coaxial piezoresistive fibre and acrylic knitting yarn. (c) Scanning electron microscopy (SEM) cross-sectional image of the piezoresistive fibre. (d) The resistance profile of a typical sensor (composed of two piezoresistive fibres) in response to pressure (or normal force). Error bars indicate the standard deviation of four individual sensors. (e) The influence of fabric structures ('manual' inlay and automatic inlay) on device performance. Behaving as a buffer, the introduction of the soft fabric (red curve) decreases the sensitivity and increases the sensing range. Also, the ribbed structures obtained from automatic inlay induce gaps between two aligned fabrics (dark blue and light blue curves), further decreasing the sensitivity and increasing the detection range. (f) The stable performance of a typical sensor over 1,000 cycles of force load and unload.

(216 sensors over an area of 144 cm^2), a vest (1,024 sensors over 2,100 cm^2), a robot arm sleeve (630 sensors over 720 cm^2), and full-sized gloves (722 sensors over 160 cm^2). Thanks to the digital machine knitting system [Seiki, 2011], these garments are fully customizable: they can be adapted to individual shape, size, surface texture and colour preference, meeting the needs of personalization and fashion design.

9.3 Self-Supervised Sensing Correction

During the large-scale manufacturing of wearable sensors, it is inevitable that variations will be introduced and thus failure. Although researchers conventionally attempt to fabricate flawless sensor arrays [Briseno et al., 2006, Khang et al., 2006], we draw inspiration from living organisms, which develop the ability to adapt their sensory system in the presence of localized defects or environmental variations using overall sensing [Dahiya et al., 2009, Winter, 1995, Yousef et al., 2011]. We adopt a similar mechanism to provide robust sensing capabilities while relaxing the flawless requirement in sensor fabrication. This is critical for many applications, as it is impractical to perform individual calibration and correction of our sensing units due to their high-density, complex geometries and diverse applications. To implement such a mechanism, we developed a self-supervised learning paradigm that learns from weak supervision, using spatial-temporal contextual information to normalize sensing responses, compensate for variation and fix malfunctioning sensors. In particular, we first calibrate our tactile socks and gloves by collecting synchronized tactile responses and readings from a digital scale stepped or pressed by a wearer (Figure 9-3a). At each frame, the scale reading indicates the overall applied pressure, which is expected to correlate linearly with the sum of tactile responses at all sensing points. We then train a fully convolutional neural network with four hidden layers, which takes a small sequence of raw tactile array responses as input and outputs a single frame with the same spatial array resolution. The output represents the corrected tactile response of the middle frame of the input sequence. The neural network is optimized via stochastic gradient descent with the objective consisting of two components: one encourages the output to preserve the spatial details from the input, and the other requires the summed tactile response to be close to the reading from the scale. Our network consistently increases the correlation between the tactile response and the reference (reading from scale)—the correlation improves from 75.9% to 91.1% in the

right sock, from 92.4% to 95.8% in the left sock and from 77.7% to 88.3% in the glove (Figure 9-3a).

To further calibrate the sensing fabrics with arbitrary shapes, such as the vest and robot arm sleeve, we treat the corrected glove as a mobile, flexible 'scale' and record data from both the tactile glove and the target garments while researchers randomly press on the latter through the glove. With the same self-supervised learning framework (Figure 9-3b), the correlation between the responses increases from 32.1% to 74.2% for the tactile vest and from 58.3% to 90.6% for the tactile robot arm sleeve (Figure 9-3b). The self-supervised calibration network exploits the inductive bias underlying the convolutional layers [Ulyanov et al., 2018], learns to remove artefacts, and produces more uniform and continuous responses (Figure 9-3c–h). It enables the large-scale sensing matrix to be robust against variation among the individual elements and even their occasional disruption, consequently improving the reliability of the measurement.

9.4 Learning on Human–Environment Interactions

Using our full-body sensing garments, we collected a large tactile dataset (over 1,000,000 frames recorded at 14 Hz, details are provided in the Section 9.5) [D'Alessio, 1999] featuring various human–environment interactions, including diverse contacting patterns from sitting on chairs of different materials with different postures, complex body movement and other daily activities (Supplementary Videos 3–7 on the *Nature Electronics* website: `https://www.nature.com/articles/s41928-021-00558-0`). The capture of human behaviour, skills and crafts is essential for cultural preservation, transfer of knowledge, as well as for human and robot performance optimization [Sundaram et al., 2019, Adams, 1971]. We demonstrate the capability and utility of our platform by leveraging our data for environment and action classification, motion pattern discovery and full-body human pose prediction.

Our full-sized sensing vest shows the characteristic pressure distributions during sitting, standing, reclining and other actions, which indicate the wearer's pose, activity and the texture of the contacted surfaces. We captured a dataset for a wearer performing different poses over various surfaces (Figure 9-4a). Projecting the high-dimensional sensory responses into 2D space via t-distributed stochastic neighbour embedding (t-SNE) [Van der Maaten and Hinton, 2008], we observe that the recordings from different classes naturally form

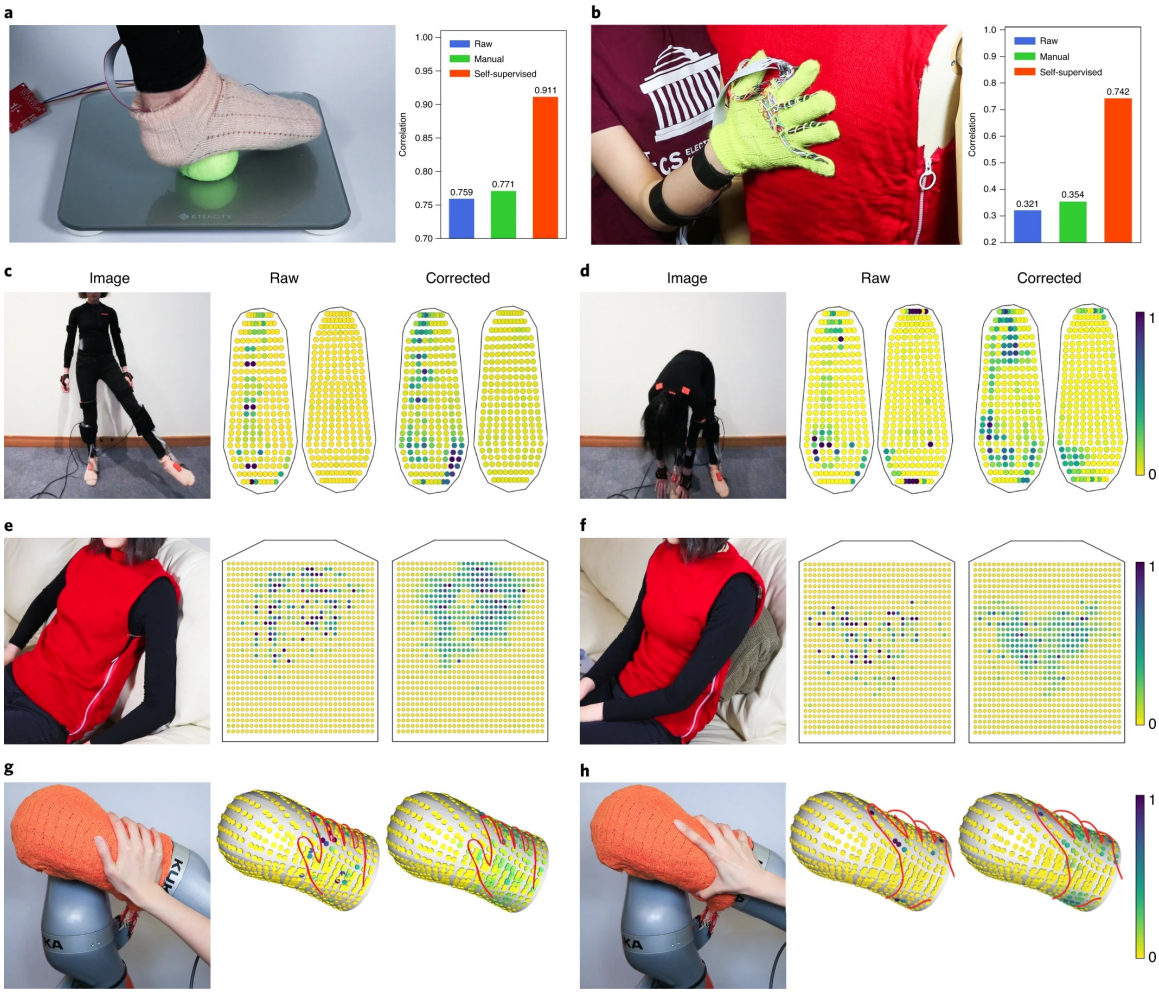


Figure 9-3: **Self-supervised correction.** (a) Procedure for correcting the tactile sock. The wearer steps on a digital scale (a tennis ball was placed between the foot and the scale to enhance conformal contact) and synchronized readings from the sock and scale are collected. The same method is used for tactile gloves. (b) Procedure for self-supervised correction of the vest using the response of the calibrated glove as the reference. The same method is also used for the robot arm sleeve. The bar plots on the right (a,b) show the correlations between the tactile response and the scale reading: 'Raw' indicates the correction of the original, unprocessed tactile signal; 'Manual' indicates the correlation of manual-adjusted data where all saturated tactile signals were clipped; and 'Self-supervised' indicates the correlation obtained after self-supervised correction. (c–h), Examples of raw and corrected readouts from the sock (c,d), vest (e,f) and robot arm sleeve (g,h). Our method removes artefacts and enhances the smoothness of the sensors. The colour bar indicates the relative pressure in each sensing point.

distinctive clusters, indicating the discriminative power of the vest (Figure 9-4b). Our vest also exhibits a discriminative resolution of 2 cm, which is superior to a human's back (∼4.25 cm) [Lederman and Klatzky, 2009]. We dressed a manikin in the vest and collect a dataset by pressing three letters cutouts—'M', 'I' and 'T'—against the back in different orientations (Figure 9-4c). The classification network takes a small window of tactile responses and predicts the type and orientation of the letter with an accuracy of 63.76%; the accuracy drops as the effective resolution decreases (Figure 9-4d). The discriminative capability of the sensing socks was demonstrated similarly with action classification. Details are provided in Supplementary Figure 11e on the *Nature Electronics* website.

Humans maintain the dynamic balance of the body by redirecting the centre of mass and exerting forces on the ground, which results in distinct pressure distributions on the feet [Someya et al., 2016, Bauby and Kuo, 2000, Scott et al., 2020]. Given this, we hypothesize that a person's pose can be estimated from the change of pressure distribution over time obtained by our tactile socks, shown as a sequence of pressure maps (Figure 9-4i-j). Here, the body pose is represented by 19 joint angles spanning over the legs, torso and arms (Figure 9-4e). We record synchronized tactile data from a pair of sensing socks and a full-body motion capture (MOCAP) suit, while the user conducts versatile actions. We model the pose prediction task as a regression problem using a convolutional neural network. The model processes a time-series of tactile array footprints that contain the evolving information about the contact events and predicts the human pose in the middle frame. We optimize the neural network by minimizing the mean squared error between the predicted and the ground truth joint angles (MOCAP data) using stochastic gradient descent.

The model learns to make accurate predictions that are both smooth and consistent over time, achieving a mean squared error that is 70.1% lower than our baseline, which always outputs the mean pose. As shown in Figure 9-4f–j, our model achieves higher accuracy for the poses in the torso and legs than in the arms. This is congruous with our observation that the pressure distributions on the feet are mostly affected by lower body movement and the majority of body mass is located in the torso [Someya et al., 2016]. We also observe that the recovered dynamic motion implies the environmental spatial information, for example, whether the stairs are upwards or downwards. The significance of sensing resolution and temporal information is reiterated, as the performance drops with a systematic reduction in either the input resolution of the tactile pressure map or the context size of the input tactile

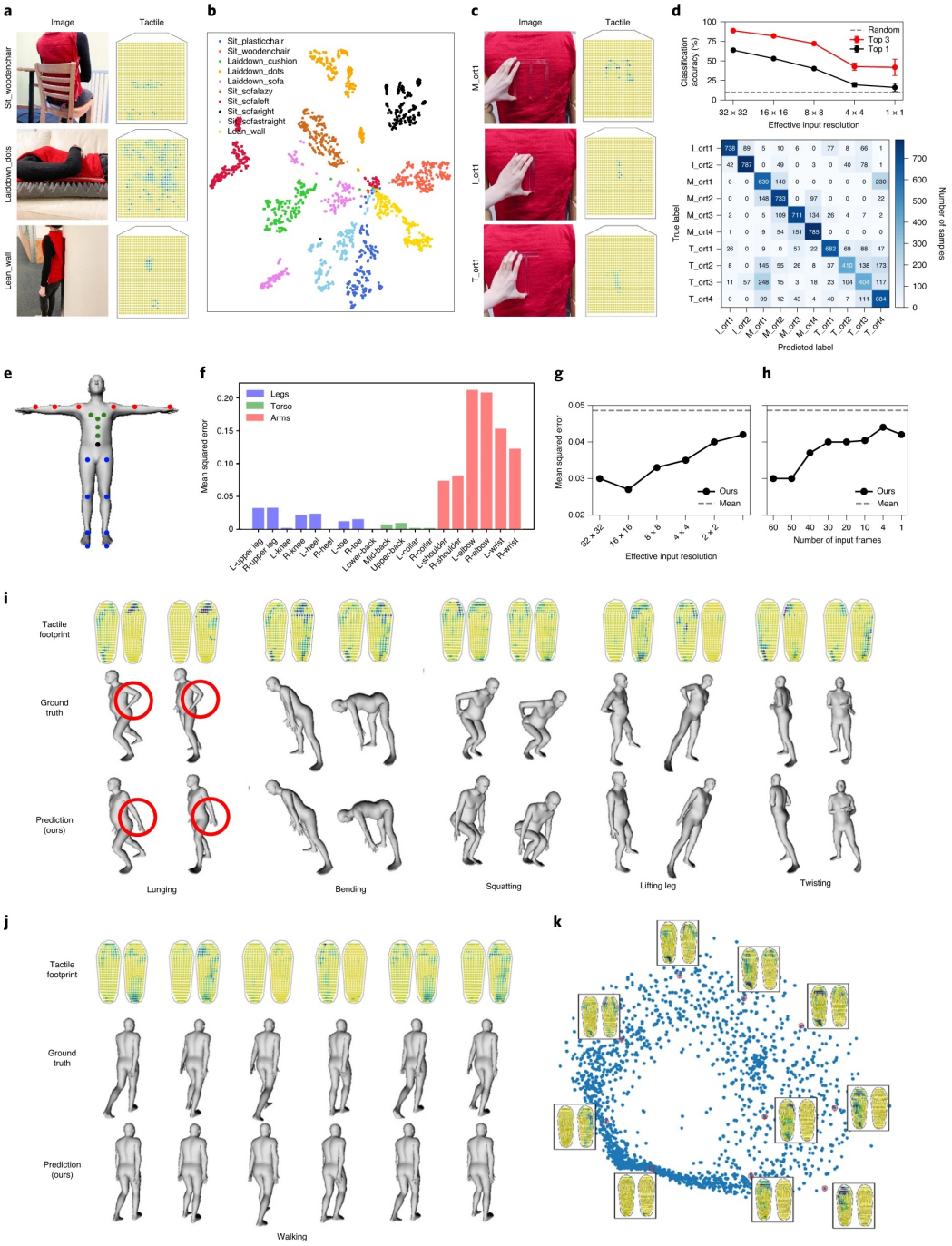


Figure 9-4: **Learning on human–environment interactions.** (a) Example photographs and tactile frames. (b) t-SNE plot from our pose dataset recorded by the tactile vest. The separation of clusters corresponding to each pose illustrates the discriminative capability of the sensing vest. (c) Example photographs and tactile frames of 'M', 'I' and 'T' pressed on the tactile vest. (d) Letter classification accuracy drops as sensor resolution decreases (top). The confusion matrix for classifying the letter and the orientation (bottom). (e) Location of the 19 joint angles representing body pose in our model. (f) Mean squared error in pose prediction. (g,h) Influence of sensor resolution (g) and the number of input frames (h, temporal window) on prediction performance. The dashed lines represent the baseline defined as the canonical mean pose of the training data. (i) Comparison of various poses recorded by MOCAP (ground truth) and the same poses recovered by our model from the tactile socks pressure frames. Notable errors in the arm region are highlighted in red. (j) Time series prediction of walking. (k) PCA on tactile maps from walking (insets are corresponding tactile frames).

frames (Figure 9-4g-h).

To further understand the patterns that emerged from the tactile dataset recorded by the sensing socks, we used principal component analysis (PCA) to project the tactile signals collected from walking into a 2D plane (Figure 9-4k). Intriguingly, the walking signals naturally form a circular pattern in the reduced PCA space. The pressure distribution smoothly transitions back and forth between the left and right foot as we traverse the circle, which describes signatures of the different phases during walking [Yousef et al., 2011].

Our results indicate that a pair of tactile socks can potentially replace the bulky MOCAP system. Our approach sets a path towards the analysis and study of human motion activities without much physical obtrusion in domains like sports, entertainment, manufacturing activities and care of the elderly. Furthermore, such footprints contain dynamic body balance strategies, demonstrating a valuable, instructive paradigm for robot locomotion and manipulation [Someya et al., 2016, Sundaram et al., 2019].

Finally, we also demonstrate that our 3D conformal sensing textiles can be used as tactile robot skin. Most modern robots rely solely on vision; however, large-scale and real-time tactile feedback is critical for dexterous manipulation and interaction skills, especially when vision is occluded or disabled [Okamura et al., 2000, Cheng et al., 2019]. Our functional textile enables conformal tactile coverage of the robot arm (Figure 9-1b–i), gripper, limbs and other functional parts with complex 3D geometries, endowing the robots with real-time tactile feedback (Figure 9-3e). Our platform offers a critical ingredient for unobtrusive multi-point collision detection and physical human–robot interaction, which remains challenging with embedded torque sensors in the robot arm [Sundaram et al., 2019].

9.5 Methods

9.5.1 Fabrication of Coaxial Piezoresistive Functional Fibre

The functional fibre was constructed in two parts: a conductive core and a piezoresistive sheath. The piezoresistive sheath was prepared by mixing graphite nanoparticle (400–1,200 nm, US Research Nanomaterials), copper nanoparticle (580 nm, US Research Nanomaterials) and PDMS elastomer (Sylgard-184, base to curing agent weight ratio of 10:1, Dow Corning). Silicone solvent OS2 (Dow Corning) was added to optimize the viscosity of the mixture for the coating. The mixture was thoroughly mixed by a speed mixer (FlackTek) at 2,500

r.p.m. for 90 s. The prepared piezoresistive composite was loaded into a syringe (Nordson), which was connected to a customized material reservoir with 500-μm-diameter inlet and 700-μm-diameter outlet (Figure 9-1a). Constant pressure (20 psi) was applied to the syringe with a Nordson dispenser, while the 3-ply stainless-steel thread (Sparkfun, DEV-13814) was fed to the inlet (Figure 9-1a). The thread was then pulled by a continuously rotating motor and coated with the nanocomposites. After thermal curing at 150°C, the resulting coaxial piezoresistive fibre (namely, the functional fibre) was wound into a roll (Figure 9-1a). Each sensor was constructed at the intersection of two orthogonally overlapped functional fibres, forming a layered structure with the piezoresistive nanocomposite sandwiched by two conductive electrodes (inset, Figure 9-2d).

9.5.2 Morphological Characterization

The longitudinal uniformity is illustrated by optical images (Olympus SZ61, Supplementary Figure 1a on the *Nature Electronics* website: https://www.nature.com/articles/s41928-021-00558-0). The microstructure of the coaxial piezoresistive fibre was further characterized using a scanning electron microscope (Zeiss Merlin, Supplementary Figure 1b on the *Nature Electronics* website). All fibres were characterized without additional conductive coating in the scanning electron microscope using a voltage of 3–5 kV.

9.5.3 Electrical Characterization

The individual sensor, constructed from two orthogonally aligned functional fibres, was used for electrical characterization. The resistance profile was recorded by a digital multimeter (DMM 4050, Tektronix) while an adjustable pressure (or normal force) was applied to the sensor by a mechanical testing system (Instron 5944). The applied load was controlled at a specific strain rate for all tests (0.05-0.5 mm min^{-1}). To evaluate the robustness of the sensor, we cycled the force between 0.1 and 0.5 N for more than 1,000 cycles at a constant strain rate of 0.1 mm min^{-1} (Figure 9-2f). We measured the performance of sensors composed of fibres with various graphite and copper compositions and coating thicknesses (Supplementary Figure 1a,b,h on the *Nature Electronics* website: https://www.nature.com/articles/s41928-021-00558-0). Sensor stability was studied with different fibre alignment angles, humidity and temperature (Supplementary Figure 1i–k). The effect of fabric structure on single sensor performance was evaluated by recording

the resistance of an isolated sensor embedded in knitted fabrics with different structures. Also, sensors consisting of different combinations of fabric structures inlaid with functional fibres (automatic and manual inlaying) were investigated (Figure 9-2e). The typical sensing resolution ranged from 0.25 cm^2 to 4 cm^2.

9.5.4 Mechanical Characterization

The tensile test was conducted on an Instron mechanical tester (Instron 5984). The fabricated functional fibre (diameter of 600 μm) was compared with stainless-steel core thread and two common kinds of acrylic knitting yarn (Tamm Petit C4240 and Rebel TIT8000), as shown in the Supplementary Figure 1g on the *Nature Electronics* website: https://www.nature.com/articles/s41928-021-00558-0. Samples with a length of 10 cm were pulled at a strain rate of 5 mm min^{-1}. The yield strength of the functional fibre is over six times larger than that of the acrylic knitting yarns. However, its ultimate strain is less than 10% of acrylic knitting yarn's ultimate strain. These mechanical characteristics require special accommodations during the machine knitting process. The functional fibre shows mechanical properties similar to that of the core stainless-steel thread.

9.5.5 Conformal 3D Sensing Textiles Manufacturing

Digital machine knitting. The full-sized conformal garments and coverings were computationally designed using KnitPaint [Seiki, 2011] and the stitch meshing framework, and later automatically fabricated with a V-bed digital knitting machine (SWG091N2, Shima Seiki), which has two beds of needles (front and back) forming an inverted 'V' shape. Each needle is composed of a hook, which catches the yarn and holds the topmost loop, and a slider, which can be actuated to move vertically to close the loop and assist in holding the loop. Movement of the yarn carrier is synchronized to the needle operation, where yarns are fed in with appropriate tension (Supplementary Figure 5a on the *Nature Electronics* website: https://www.nature.com/articles/s41928-021-00558-0). We used three basic needle operations during fabrication: knit, tuck and transfer. A knit operation actuates the needles to grab the fed yarn from the yarn carrier, forms a new loop, and pulls it through the existing loop to connect the loops in columns (Supplementary Figure 5b on the *Nature Electronics* website). A tuck operation actuates the needles to grab the yarn and hold it without forming a new loop (Supplementary Figure 5c on the *Nature Electronics* website). A

transfer operation actuates needles from both beds to pass the existing loop from one bed to the other (Supplementary Figure 5d on the *Nature Electronics* website). We also utilized racking, where the back bed shifts laterally to the left or to the right as a whole to create needle offsets during transferring to guide the complex 2D and 3D shaping.

Inlaying. Two methods of inlaying were utilized to seamlessly integrate the piezoresistive fibres into conformal 3D knitted textiles: automatic inlaying and manual inlaying. Automatic inlaying requires the fabric structure of 'ribs', which are textured vertical stripes created by alternating columns of knit stitches on the front and back bed. The functional fibre is forced to move simultaneously with normal knitting yarn, which is caught by the needles on the two beds and forms alternating knit stitches to hold down the inlaid functional fibre (Supplementary Figure 5e on the *Nature Electronics* website: https://www.nature.com/articles/s41928-021-00558-0). The ribbed structure (alternating stitches on the front and back bed) allows the straightforward inlaying design; however, it creates a textured gap when two fabrics are aligned orthogonally to act as a functioning device and lowers the sensitivity and accuracy. Manual inlaying requires consecutive movements of the normal knitting yarn and the functional fibre. Four steps of operations are conducted: (1) knitting with normal knitting yarn; (2) the transfer of specific loops from front to back bed; (3) moving the functional fibre across the fabric; (4) transfer of specific loops from back to front bed to their original positions. A flat inlaid fabric (without ribbed structure) was fabricated with manual inlay, and designs with continuous functional fibre coverage were achieved by alternating the transfer direction (from the front bed to back bed or from back bed to front bed). However, due to the fibre stiffness and the limited space between two beds, the functional fibre can barely stay down during the second round of stitches transfer and is easily caught by the needles, leading to the destruction of fibre functionality. Therefore, the manual inlay can only be applied to 2D structures.

Full-sized conformal 3D sensing textiles. Two knitted fabrics of specific 2D and 3D shapes with horizontally and vertically inlaid functional fibres were arranged as a double-layer structure to form the large-scale sensing matrix. To optimize the manufacturability and sensing performance of the garments, we exploited both automatic inlaying and manual inlaying. Different combinations of fabric texture (ribbed or not ribbed) were selected for

different devices to obtain the desired sensitivity and detection range (Figure 9-2e). Full-sized socks, vest, gloves and conformal robot arm (LBR iiwa, KUKA) sleeve were fabricated with the designs shown in Supplementary Figure 6 on the *Nature Electronics* website: https://www.nature.com/articles/s41928-021-00558-0. To connect the embedded sensing matrices to the readout circuit for data transmission without disrupting the main knitted fabric, cutting was performed at the edge of the fabric, where 1–2-cm functional fibres were pre-reserved through additional tucking at the edge for connection. A modified electrical-grounding-based circuit architecture [D'Alessio, 1999] was used to eliminate most crosstalk and parasitic effects of the passive matrix (Supplementary Figure 7a on the *Nature Electronics* website).

9.5.6 Self-Supervised Sensing Correction

Dataset. During data collection for the tactile sock correction, our researcher wore the sock (with a customized printed circuit board mounted on the calves) and conducted random stepping on a digital weighing scale (Etekcity). The tactile learning platform is powered by a portable energy source (Dell XPS 13 9380, 30 Wh), which allows real-time recording (data acquisition and serialization) while being carried by the researcher in a backpack. The fully charged laptop allows 4 h of recording. We recorded both the real-time tactile information from the sensing matrix and the readings from the scale. To ensure the conformal contact of all sensors, we placed a tennis ball on the scale during the data-collection process (Figure 9-3a). Sensing correction for the tactile gloves used a similar data-collection process. For data collection with gloves, the printed circuit boards were mounted on the arms. A person wore the glove and conducted random pressing on a digital scale (Lansheng). We used 3D printed shapes (dots, lines and curves) to ensure conformal contact of individual sensors. In total, we collected 25,024 frames for the left sock correction, 37,275 frames for the right sock correction and 108,305 paired frames for glove calibration. We split the dataset sequentially, with the first 80% reserved for training, another 10% for validation and the remaining 10% for testing.

Data processing and network architecture. The data-collection process is assumed to be quasistatic, so the readings from the scale should have a linear correlation with the sum of all sensor responses on the glove or the sock. Based on this observation, we developed a self-supervised learning framework that uses the scale's reading as weak supervision to guide

correction of the garment. We preprocessed all frames by replacing shorted or dead sensors with an average value of its surrounding. A fully convolutional neural network augmented with skip connections was developed. The model takes a small sequence of the processed frames as input and then adds an individual bias term to each sensor. The output of the model is a tensor of the same spatial resolution as the input with a channel size of 16. The scale prediction is then defined as the mean of all values in this tensor. We applied an affine transformation to the first channel of the tensor to derive the calibration result, which corresponded to the middle frame of the input sequence. The affine transformation was parameterized by two scalar numbers—a scaling factor and a bias term—which were shared by all sensors in the same garment.

The calibration network was expected to predict the readings from the scale accurately and preserve the details of the input frame. Hence, the overall objective was a reweighted sum of two components: one was the mean squared error between the predicted scale reading and the actual scale reading, while the other was an L1 distance between the calibration result and the corresponding input frame, where we scaled the second reconstruction loss with a factor of 0.1. We optimized the parameters in the convolutional layers, the position-wise bias and the global affine transformation using stochastic gradient descent, which employed the Adam optimizer with an initial learning rate of 0.001 and a batch size of 64. We decreased the learning rate with a factor of 0.8 whenever the loss on the validation set plateaued for five epochs.

Self-correction. After properly correcting the glove, we used it as our new 'scale' to calibrate other garments like the vest and the robot arm sleeve. During data collection, the customized printed circuit board was mounted on the leg for the vest and mounted on the robot arm for the sleeve. We randomly pressed the target garment embedded with the sensing matrix using the corrected glove. In total, 56,031 paired frames and 44,172 frames were collected for the vest and robot arm sleeve correction. We used the same data split, network architecture and training procedure, except that we used the sum of the sensors from the calibrated glove as the target value to minimize the scale prediction error.

9.5.7 Human Pose Classification

To evaluate the discriminative power of the vest, and the quality of the obtained signal, a neural-network-based classifier was developed to distinguish different sitting postures and contacted surfaces. We selected 10 different classes of poses and recorded 82,836 tactile frames, in total, to construct the dataset. The input to the model is a small sequence of 45 consecutive frames, which totals ∼3 s in real time. The sequences were selected from the dataset with a stride of 2, among which 5,000 sequences were used for validation (500 per class), another 10,000 sequences were reserved as the test set (1,000 per class), and we used the rest for training. We passed each tactile frame through three shared convolutional layers. The resulting 45 vectors with a length of 512 after flattening were then passed through a bidirectional gated recurrent unit [Cho et al., 2014] and two fully connected layers to derive the final probabilistic distribution over the 10 classes. We trained the model using the standard cross-entropy loss for 20 epochs with the Adam optimizer with learning rate of 0.001 and batch size of 32. We obtained an accuracy of 99.66% on the test set in distinguishing different lying postures and supporting surfaces. A real-time pose classification with an accuracy of 94% is demonstrated in Supplementary Video 8 on the *Nature Electronics* website: https://www.nature.com/articles/s41928-021-00558-0. Here, the real-time classifier predicts the probability of each class, accompanied by the ground-truth human pose.

9.5.8 Letter Classification

The dataset was collected from the tactile vest worn by a manikin when the models of three letters—'M', 'I' and 'T'—were pressed against the back. We recorded 62,932 frames in total for the 10 classes and retrieved a top-1 accuracy of 63.76% and a top-3 accuracy of 88.63% using the same network architecture, data splitting and training procedure as the pose classification task. To ablate the resolution of the sensor, we reassigned the value in each 2×2 grid with the average of the four values, which reduced the effective resolution to 16×16. We then used the same classification training pipeline to obtain accuracy. A similar procedure was employed for calculating the accuracy for sensors with effective resolutions of 8×8, 4×4 and 1×1.

9.5.9 Action Classification and Signature Discovery

In this experiment, we used tactile information from the two socks worn by a person to identify which action the wearer was conducting. The dataset consists of tactile frames retrieved from a pair of socks when the wearer conducts nine different activities: walking, climbing up the stairs, climbing down the stairs, fast walking, standing on toes, jumping, leaning on the left foot, leaning on the right foot and standing upright. The dataset contains 90,295 frames across the different action categories. The same classification pipeline achieved a top-1 accuracy of 89.61% and a top-3 accuracy of 96.97%.

To further analyse the patterns underlying the signals collected from the socks, we performed PCA to visualize their distribution. We used the 12,245 frames from the class of walking in the action classification dataset. We concatenated and flattened the signals from the left and right sock as the high-dimensional representation at each time step, each of which has 2,048 dimensions. We extracted the principal components with the highest and the second-highest variance to project the high-dimensional responses to a 2D space and visualize them in Figure 9-4k.

9.5.10 Full-Body Pose Prediction

Dataset. The key idea of full-body pose prediction is to look at how different sequences of tactile footprints correlate to the different poses as the person transitions from one pose to another. To this end, we simultaneously collected data from a pair of tactile socks (frame rate of 13–14 Hz) and an XSENS motion capture system (MOCAP, frame rate of 50 Hz) worn by a person. The person conducted exercises and other daily activities, including walking, bending forward, twisting, lunging and so on. The collected dataset is diverse in terms of poses as the person transitions through different tasks. The MOCAP is composed of 17 inertia-based sensors, mounted on 17 key points on the human body to record and estimate the orientation of 19 different joints during the movement. The real-time pressure imprints from both feet were recorded and fed into the network. The 19 different joints include the joints of the legs, arms and torso. The collected dataset includes 282,747 frames of concurrent MOCAP and tactile pressure maps: 236,036 frames (∼83.5%) were used as the training set, 10,108 frames (∼3.5%) as the validation set and 36,603 frames (∼13%) as the test set.

Data processing and network architecture. We trained a deep convolutional neural network to predict the 19 different joints of the human body, given the tactile footprints of the person. The network consists of two convolution layers, which take in a sequence of tactile frames from the socks from time step t - k to time step t + k. The input to the model is 30 consecutive frames of the pressure map of the left and right feet. These layers extract patterns from the 2D signal, and the resulting embedding is then passed through three fully connected layers to finally output the predicted relative joint angles of the human body, corresponding to the person's pose at time step t. The relative joint angles are the relative angle transformation of the distal joint with respect to the proximal joint represented in the axis angle format. The network was trained using stochastic gradient descent to minimize the mean squared error between the predicted joint angles and the ground truth.

We trained the network with a batch size of 128 and with a learning rate of 0.01 using the Adam optimizer. This was a design choice to predict the pose in the joint angle space instead of the position space, but the code provided can be easily modified to predict the pose in position space. The collected dataset also contains positions of the different joints that could be used to train such a model.

Evaluation. The model is validated with the test dataset, including various exercises. The ability of the model to infer each of the joint angles is illustrated in Supplementary Figure 13 on the *Nature Electronics* website: https://www.nature.com/articles/s41928-021-00558-0. The model can infer the motion of the lower body better than the motion of the upper body. It is also found that the motion of the hands does not induce a systematic change in the tactile pressure map, and hence their joint angles are more difficult to infer. There was no additional loss used to train the system to be temporally consistent. The network figured out the correspondence between the trajectory and the pose, and smoothly transitioned from one pose to another accordingly. More results are provided in Supplementary Figure 12 and Supplementary Video 5 on the *Nature Electronics* website. We also evaluated how the performance of the model changes with a systematic reduction in the resolution on each of the feet from 32×32 to 1×1 (with the same ablation method used for letter classification).

9.6 Discussion

We have reported large-scale tactile sensing textiles with dense sensor arrays that are conformal to arbitrary 3D geometries. The textiles are manufactured via an automated, inexpensive and scalable approach that combines functional fibres and full-garment digital machine knitting. A self-supervised sensing correction has been developed to normalize the sensor responses and correct malfunctioning sensors in the array. Demonstrations of various human–environment interaction learning using the system suggest that our approach could be of use in biomechanics, cognitive sciences and child development, as well as in imitation learning for intelligent robots.

Chapter 10

Conclusion and Future Works

In this dissertation, I have discussed a holistic picture of how my research has contributed to the robots' capability of making better physical interactions with the environment, all the way from **sensing**, **perception**, **dynamics**, and **control**. I have also discussed in depth how a suitable choice of structure in the system can deliver much better sample efficiency and generalization capability.

Specifically, this thesis (1) investigated the construction of novel multi-modal **sensing** platforms (i.e., vision and touch) to lay the foundation for structured modeling of the interaction process, (2) studied the **perception** problem of what scene representation and at what level of abstraction we should use to describe the environment that acts as sufficient statistics for a given downstream task, (3) performed comprehensive examinations of different functional classes for their abilities to exploit the structure and model the **dynamics** of the underlying system, and (4) showed how the learned dynamics model could be used to solve various inverse problems like **physical inference** and **model-based control** both in the simulation and the real world. I have also showcased various robotic systems successfully performing real-world manipulation tasks involving objects with complicated properties like deformable objects, granular materials, and fluids.

In the future, robots should be able to operate in unstructured environments and make complicated physical interactions as dexterous and effective as we humans do. Over the past years, there has been impressive progress in enriching robots' capabilities, but the performance is still far from ideal for real-world practical deployments. Huge performance gaps exist in virtually every aspect of the system between robots and humans, ranging

from perception, dynamics modeling, and planning/control. Progress on this front requires interdisciplinary studies involving various research areas, including but not limited to robotics, computer vision, machine learning, control, etc. The tools and techniques developed during the process would also deepen our understanding and expand the horizon of the respective subfields, and suggest new research topics. Below I identify three research directions that will naturally extend my past and ongoing research towards tackling some of the open challenges in this area.

From specialist to generalist: adaptive scene representations from raw sensory inputs. Despite rapid progress over the past years, automatic selection and adaptation of structured scene representations from high- dimensional observation data are still far from solved, especially for challenging objects like sushi, salad, cloth, etc., that undergo complex physical interactions with severe occlusions. However, humans' mental model of these objects, although not perfect, is still way better than the state-of-the-art computational models. For example, when buttoning our shirt, we do not have to know the full state of the cloth and only have to focus on local regions, or when grabbing a mug, we would have representations at different abstraction levels at different stages of the operation. Replicating or even surpassing such capability is no easy task. It requires us to draw insights from the cognitive science and neuroscience communities, and develop new learning paradigms to integrate structural priors in the optimization procedure that allows automatic and adaptive representation selection for effective dynamics modeling. The problem, which we termed as online model order reduction, is intriguing and would be of great value in expanding robots' capabilities.

Model-based optimization beyond planning & control. Prior works have shown impressive planning and control results through optimization with a learned dynamics model. However, a series of theoretical and practical questions remain unanswered, including questions about guarantees, robustness, the role of learning, how to incorporate domain-specific knowledge, and others. I desire to draw inspiration and bring tools from communities like theoretical machine learning and control to establish confidence and make formal claims about the reliability of the derived controller. One idea would be to construct and plan within the trust regions calculated from the model's confidence in its prediction. Going

beyond planning and control, the dynamics model is essentially a forward mapping from the inputs and model parameters to the system's outputs. I also aspire to use it for other inverse problems like inverse design, co-optimization of robot design and control, material and drug discovery, etc.

Towards Metaverse: seamless transitions between simulated and real-world physical interactions. Humans can hold an object in their hands and turn it over, feeling different parts and constructing a mental model of the object describing its shape, material properties, and dynamics. However, due to the inaccurate modeling of the physical interactions, it has always been a big challenge for computational models to map the real world into the simulation or match the simulation to real-world observations. Our scalable multi-modal sensing platforms [Sundaram et al., 2019, Luo et al., 2021b] will provide more detailed and physically-grounded 3D modeling of the scene and the interaction process from vision and touch. It will deepen our understanding of how different modalities complement each other and allow us to construct physics-based and/or learning-based simulators that can replicate the physical interactions and sensory response, enabling seamless transitions between the simulation and the real world. I envision the overall system can learn to model the geometry and characterize the scene dynamics from a few trials of interactions. It can then support interactive applications in virtual and augmented reality (VR/AR) and facilitate sim-to-real and real-to-sim transfer of challenging robotic manipulation tasks like dexterous in-hand manipulation/reorientation, non-prehensile manipulation, and human-robot interactions.

Now is a particularly exciting time to work in this interdisciplinary area with the rapid development of (i) large-scale data sets and physics simulators, (ii) computation and robotic hardware, and (iii) algorithms. There is huge potential to build intelligent agents that can perceive and interact with the world with unprecedented performance. I am excited to continue my academic journey and build close collaborations with various institutions at the intersection of robotics, computer vision, and machine learning, while more broadly collaborating with researchers across fields in control theory, computer graphics, mechanical engineering, cognitive science, neuroscience, computational fabrication, etc.

Appendix A

Learning Keypoint Dynamics via Self-Supervised Dense Correspondence

A.1 Dense Correspondence

Here we give a brief overview of the dense correspondence model formulation [Schmidt et al., 2017, Florence et al., 2018] with spatial distribution losses [Florence et al., 2019, Florence, 2019]. We briefly explain the loss functions and how the descriptors $\{d_i\}$ are localized.

A.1.1 Network Architecture

We use an architecture that produces a full resolution descriptor image. Namely it maps

$$W \times H \times 3 \rightarrow W \times H \times D \tag{A.1}$$

We use a FCN (fully-convolutional network architecture) [Long et al., 2015] with a ResNet-50 or ResNet-101 with the number of classes set to the descriptor dimension. Note that the FCN used in this work does not use striding and upsampling as in the architecture originally used in Florence et al. [2018].

A.1.2 Loss Function

For all shown experiments we use the spatially-distributed loss formulation with a combination of heatmap and 3D spatial expectation losses, as described in Chapter 4

of Florence et al. [2019].

Heatmap loss. Let p^* be the pixel space location of a ground truth match. Then we can define the ground-truth heatmap as

$$H^*_{p^*}(p) = \exp\left(-\frac{||p - p^*||_2^2}{\sigma^2}\right) \tag{A.2}$$

p represents a pixel location. A predicted heatmap can be obtained from the descriptor image $\mathcal{I}_D$ together with a reference descriptor d^*. Then the predicted heatmap is gotten by

$$\hat{H}(p; d^*, \mathcal{I}_D, \eta) = \exp\left(-\frac{||\mathcal{I}_D(p) - d^*||_2^2}{\eta^2}\right) \tag{A.3}$$

The heatmap can also be normalized to sum to one, in which case it represents a probability distribution over the image.

$$\tilde{H}(p) = \frac{\hat{H}(p)}{\sum_{p' \in \Omega} \hat{H}(p')} \tag{A.4}$$

The heatmap loss is simply the MSE between H^* and $\hat{H}$ with mean reduction.

$$L_{\text{heatmap}} = \frac{1}{|\Omega|} \sum_{p \in \Omega} ||\hat{H}(p) - H^*(p)||_2^2 \tag{A.5}$$

Spatial expectation loss. Given a descriptor d^* together with a descriptor image $\mathcal{I}_D$ we can compute the 2D spatial expectation as

$$J_{\text{pixel}}(d^*, I_D, \eta) = \sum_{p \in \Omega} p \cdot \tilde{H}(p; d^*, I_D, \eta) \tag{A.6}$$

If we also have a depth image $\mathbf{Z}$ then we can define the spatial expectation over the depth channel as

$$J_z(d^*, I_D, \mathcal{Z}, \eta) = \sum_{p \in \Omega} \mathbf{Z}(p) \cdot \tilde{H}(p; d^*, I_D, \eta) \tag{A.7}$$

The spatial expectation loss is simply the L1 loss between the ground truth and estimated correspondence using

$$L_{\text{spatial pixel}} = ||p^* - J_{\text{pixel}}(d^*)||_1 \tag{A.8}$$

We can also use our 3D spatial expectation J_z to compute a 3D spatial expectation loss. In

particular given a depth image $\mathbf{Z}$ let the depth value corresponding to pixel p be denoted by $\mathbf{Z}(p)$. The spatial expectation loss is simply

$$L_{\text{spatial z}} = ||\mathcal{Z}(p^*) - J_z(d^*, I_D, \mathcal{Z}, \eta)||_1 \tag{A.9}$$

being careful to only take the expectation over pixels with valid depth values $\mathcal{Z}(p)$.

Total loss. The total loss is a combination of the heatmap loss and the spatial loss

$$L = w_{\text{heatmap}} L_{\text{heatmap}} + w_{\text{spatial}}(L_{\text{spatial pixel}} + L_{\text{spatial z}}) \tag{A.10}$$

where the w are weights.

A.1.3 Correspondence Function

The correspondence function $g_c(\mathcal{I}_D, d_i)$ in Section 2.3.2 is defined using the spatial expectations J_{pixel}, J_z defined above to localize the descriptor d_i in either pixel space or 3D space. If in 3D we additionally use the known camera extrinsics to express the localized point in world frame.

A.2 Training Details

This section provides details on the simulation and hardware experiments.

A.2.1 Trajectory Data Augmentation

Many physical systems exhibit invariances in their dynamics. For example, in the environments we consider the dynamics are invariant to translation in the xy plane and rotation about the z axis. In other words, if we translate or rotate our frame of reference the dynamics don't change. Encoding this invariance into our dynamics model has the potential to greatly simplify the learning problem. Hogan et al. [2018] achieves this by parameterizing the dynamics relative to the object frame, however this approach requires having access to the ground truth object frame and assumes you are dealing with a single rigid object. Another approach, taken by Zeng et al. [2018, 2019] and Suh and Tedrake [2020], is to rotate the observation into a frame defined by the action. While this can work

Method	Learnable Parameters Θ
DS	$\{\theta_{dyn}\}$
SDS	$\{\theta_{dyn}\}$
WDS	$\{\theta_{dyn}, \alpha\}$
WSDS	$\{\theta_{dyn}, \alpha\}$
Transporter	$\{\theta_{dyn}\}$
Autoencoder	$\{\theta_{dyn}, \theta_{\text{autoencoder}}\}$

Table A.1: The set of learnable parameters for the different methods during the dynamics learning phase. For our methods and transporter, these parameters don't include the weights of the visual model which remain fixed during the dynamics learning phase.

well in the setting of simple manipulation primitives (e.g., push along a straight line for $5cm$) it doesn't naturally extend to the realtime feedback setting where you are commanding actions continuously at $5-10$ Hz without returning the robot to a reference position. Since in our approach the latent-state $\boldsymbol{z}$ is a physically grounded 3D quantity we are able to encode some of this invariance by using an alternative approach based on data augmentation. Given a latent-state action trajectory $\{(\boldsymbol{y}_t, \boldsymbol{a}_t)\}$ then transforming this trajectory using a homogeneous transform T, which consists of xy translation and z rotation, yields another valid trajectory $\{(\tilde{\boldsymbol{y}}_t, \tilde{\boldsymbol{a}}_t)\} = \{(T \cdot \boldsymbol{y}_t, T \cdot \boldsymbol{a}_t)\}$. At training time we augment the training trajectories by sampling such random homogeneous transforms T.

A.2.2 Training Details

All methods used the same architecture for the dynamics model $\hat{f}_{\theta_{dyn}}$, an MLP with two hidden layers of 500 units. All of our variants, along with the **transporter** baselines, use visual pre-training. The visual models are trained for 100 epochs and the model with the best test error is used. The dense-correspondence model uses both camera views for visual pretraining, while the transporter model uses only the images from the camera used at test time. For the dynamics learning all methods are trained for 1000 epochs using an Adam optimizer [Kingma and Ba, 2015] with a learning rate of 10^{-4}. For each method the model with the best test error was used for evaluation. Table A.1 details the set of learnable parameters for each method.

A.3 Online Model-Predictive Control

Following Nagabandi et al. [2020] we use the model-predictive path integral (MPPI) approach derived in Williams et al. [2015a]. Here we provide a brief overview but refer the reader to Williams et al. [2015a] for more details. MPPI is a gradient-free optimizer that considers coordination between timesteps when sampling action trajectories. The algorithm proceeds by sampling N trajectories, rolling them out using the learned model, computing the reward/cost for each trajectory, and then re-weighting the trajectories in order to sample a new set of trajectories. Let H be look-ahead horizon of the MPC, then a single trajectory consists of state-action pairs $\{(x_t^{(k)}, a_t^{(k)}\}_{t=0}^{H-1}$. Let $R_k = \sum_{t'=t}^{t+H-1} r(x_t^{(k)}, a_t^{(k)})$ be the reward of the k-th trajectory. Define

$$\mu_t = \frac{\sum_{k=0}^{N} \left(e^{\gamma R_k}\right) a_t^{(k)}}{\sum_{k=0}^{N} e^{\gamma R_k}}$$

A filtering technique is then used to sample new trajectories from the previously computed mean μ_t. Specifically

$$a_t^i = n_t^i + \mu_t \tag{A.11}$$

where the noise n_t^i is sampled via

$$u_t^i \sim \mathcal{N}(0, \Sigma), \quad \forall i \in 0, \ldots, N-1, \quad \forall t \in 0, \ldots, H-1 \tag{A.12}$$

$$n_t^i = \beta u_t^i + (1-\beta) n_{t-1}^i \quad \text{where } n_{t<0} = 0 \tag{A.13}$$

This procedure is repeated for M iterations at which point the best action sequence is selected. All of our experiments we used $N = 1000, M = 3, H = 10, \beta = 0.7$. The cost/reward in the MPC objective varied slightly between the hardware and simulation experiments, more details are provided below.

A.4 Simulation Experiments

To evaluate our method we consider four manipulation tasks in simulation. We use the Drake simulation environment [Tedrake and the Drake Development Team, 2019], which provides both the underlying physics simulation and rendering of RGBD images at VGA resolution $640 \times 480 \times 3$. Figure 2-3 shows image from the four simulation tasks that we

consider.

- **top down camera:** This environment, depicted in Figure 2-3a, consists of the sugar-box object from the YCB dataset [Calli et al., 2017] laying flat on a table. The robot is represented as cylindrical pusher (shown in green) and the action $\boldsymbol{a}$ is the $x-y$ velocity of the pusher in the plane. The environment timestep is $dt = 0.1$, so the agent must command actions at 10Hz. Two cameras are placed directly above the table, facing downwards. The camera positions are offset by 90 degrees about their z-axis. Our methods use both camera feeds for training the visual correspondence model, but only one camera feed at test time. An image from this camera is shown in Figure 2-3a. All other methods use only a single camera feed at both train and test time.

- **angled camera:** This environment is identical to *top down camera* but has different camera positions. Instead of being top down the two cameras are located on adjacent sides of the table and angled at 45 degrees, see Figure 2-3b. The setting of angled cameras is more similar to our hardware experimental setup and is useful for comparing approaches that use pixel space vs. 3D representations.

- **occlusions:** This environment uses the same setup of task *angled camera* the only difference being that the object is now laying on its side, see Figure 2-3c. This, together with the angled camera position, means that occlusions become a significant factor. In particular as the box rotates through the full 360 degrees in yaw, the sides of the box become alternately occluded or visible. The top face of the box is the only one that remains unoccluded for all poses of the object.

- **mugs (category):** This environment has the same top-down camera placements and cylindrical pusher as task *angled camera*. Instead of a single object however, we use a collection of 10 different mug models and vary the color and texture on each episode. This environment tests category-level vision and dynamics generalization. Two mug instances are shown in Figure 2-3d-e.

A.4.1 Data Collection

For each environment we collect a static dataset that is then used to learn the visual dynamics model. All methods have access to exactly the same dataset and the visual

pretraining for our method and the **transporter** baseline is done using this same dataset. For each task the dataset is generated by collecting 500 trajectories of length 40 using a scripted random policy. The simulator timestep is 0.1 seconds so a trajectory of length 40 equates to 4 seconds.

A.4.2 Evaluating Closed-Loop MPC performance

For each environment we evaluate the different methods by planning to a desired goal-state image and computing the pose error (both translation and rotation) using the ground truth simulator state. Goal states are generated sampling a random control input and applying it to the environment for 15 time steps. We further require that goal states are sufficiently far from initial states (in both translation and rotation). This generates a diverse set of initial and goal state pairs for evaluation. The simulator state is then reset to the initial state and we use closed-loop MPC to control the system to the goal state. The MPC cost function is simply the L2 distance between the latent state and the goal state. Ground truth state information is used to compute the error between the final and goal poses for the object.

A.4.3 Baselines

To demonstrate the benefits of our approach over prior methods we compare against several baselines.

- **Ground Truth 3D points (GT_3D):** This baseline is used for tasks *top down camera, angled camera, occlusions* since those environment use just a single object. $\boldsymbol{z}_{\text{object}}$ contains ground truth world-frame 3D locations of 4 points on the object. Knowing the location of 4 points is equivalent to knowing the object pose for a rigid object. We believe that this is a strong baseline that provides an upper bound on what is achievable with our descriptor-based methods that attempt to track points on the object.

- **Transporter:** We use the *Transporter* autoencoder formulation from Kulkarni et al. [2019] to pre-train a visual model. Following the original paper we use 6 keypoints and freeze the visual model while training the dynamics model. We investigate two variants using the transporter approach. In **transporter 2D** $\boldsymbol{z}_{\text{object}}$ are the pixel-space locations of the keypoints. In **transporter 3D** $\boldsymbol{z}_{\text{object}}$ are the 3D world frame locations

of the keypoints, computed from the pixel space by using the depth image together with the camera intrinsics and extrinsics.

- **Autoencoder:** This method jointly learns the visual model and the dynamics model. Specifically we jointly train a convolutional autoencoder together with a forward dynamics model. The loss is a combination of the dynamics loss, Equation 2.1, and an image reconstruction loss, which penalizes the L2 distance between the reconstructed and actual images. Note that this is exactly the autoencoder baseline from Yan et al. [2021]. Following Yan et al. [2021], images are downsampled to 64×64 before being passed into the network. The encoder architecture contains 6 2D convolutions with kernel sizes $[3, 4, 3, 4, 4, 4]$, strides $[1, 2, 1, 2, 2, 2]$ and filter sizes $[64, 64, 64, 128, 256, 256]$. Leaky ReLU activations are added between the convolutional layers. The final output is flattened and passed through a fully-connected layer to form the latent-state $\boldsymbol{z}_{\text{object}}$. We experimented with different dimensions for $\boldsymbol{z}$ from 16 to 64 and found that 64 worked best. Hence a 64 dimensional latent state is used for all experiments. The decoder follows the one in Hafner et al. [2019b] and consists of a dense layer followed by 4 transposed convolutions with kernels size 4 and stride 2 which upscales the output image to 64×64.

A.5 Hardware Experiments

A.5.1 Hardware Setup

We used a Kuka IIWA LBR robot with a custom cylindrical pusher attached to the end-effector to perform our hardware experiments, see Figure A-1. RGBD sensing was provided by two RealSense D415 cameras rigidly mounted offboard the robot and calibrated to the robot's coordinate frame. To enable effective correspondence learning between views, it is ideal to have views with *some* overlap such that correspondences exist, but still maintain different-enough viewpoints from each camera. At test time only a single camera is used to localize the dense-descriptor keypoints. The robot is controlled by commanding end-effector velocity in the xy plane at 5Hz. A high-rate Jacobian space controller consumes these 5Hz end-effector velocity commands and closes the loop to command the robot's joint positions at 200Hz.

Figure A-1: **Overview of our experimental setup.** It includes two external Realsense D415 cameras. Images from both cameras are used to train the dense-descriptor model, while only the right camera is used at runtime to localize the keypoints on the object.

A.5.2 One-Shot Imitation Learning

Although our learned dynamics model together with online MPC is able to plan over short to medium horizons, we can track much longer horizon plans by providing a single demonstration and using a trajectory tracking cost in our MPC formulation. This demonstration can in principle come from any source, in our case we used a human teleoperating the robot. We capture observations throughout the trajectory at $5Hz$ resulting in a trajectory of observations $\{\boldsymbol{o}_t^*\}_{t=0}^T$. Using our visual model we convert these observations into keypoints $\{\boldsymbol{y}_t^*\}_{t=0}^T$ and latent state $\{\boldsymbol{z}_t^*\}_{t=0}^T$ trajectories. These trajectories are used to guide the MPC. In particular the cost/reward function in the MPC is

$$r(z_t, a_t) = -||z_t - z_t^*||_2^2 \tag{A.14}$$

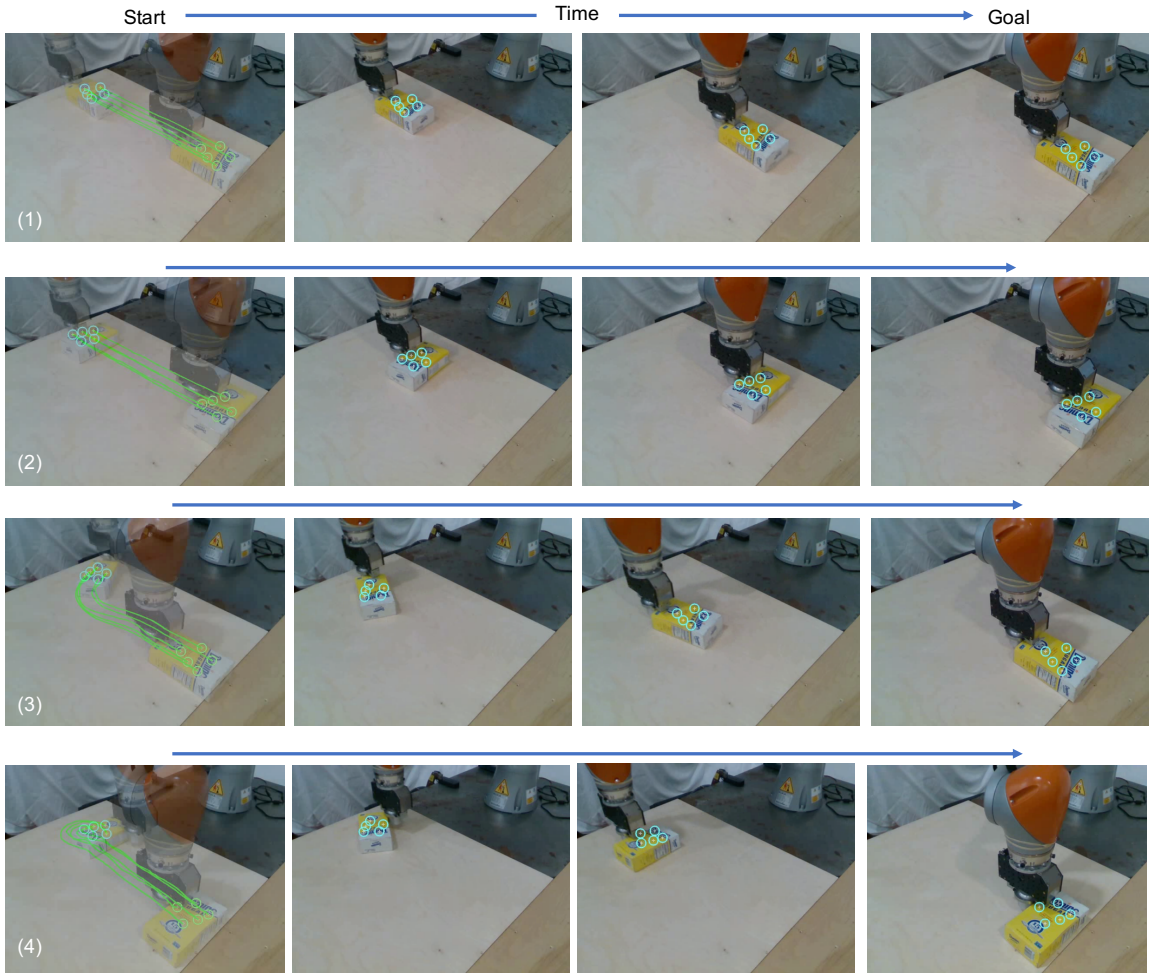


Figure A-2: **The 4 demonstration trajectories used for the hardware experiments.** The left image of each row shows the starting position blended with the goal position. The **SDS** keypoints are shown in teal for each frame. The green lines show the paths followed by the keypoints moving from the starting position to the final position. The right image of each row shows the final/goal position.

This trajectory cost allows us to accurately track long horizon plans (where the demonstrations are as long as 15 seconds) using an MPC horizon of 2 seconds. The four demonstrations trajectories used in the hardware experiments are illustrated in Figure A-2.

A.5.3 Results

In this section we provide more details on the hardware experiments from Section 2.4.4. Figure A-3 expands on Figure 2-4 showing the region of attraction of our MPC controller when attempting to stabilize the four different trajectories shown in Figure A-2. We define a trajectory as a success if the final object position is within 3 cm and 30 degrees of the goal

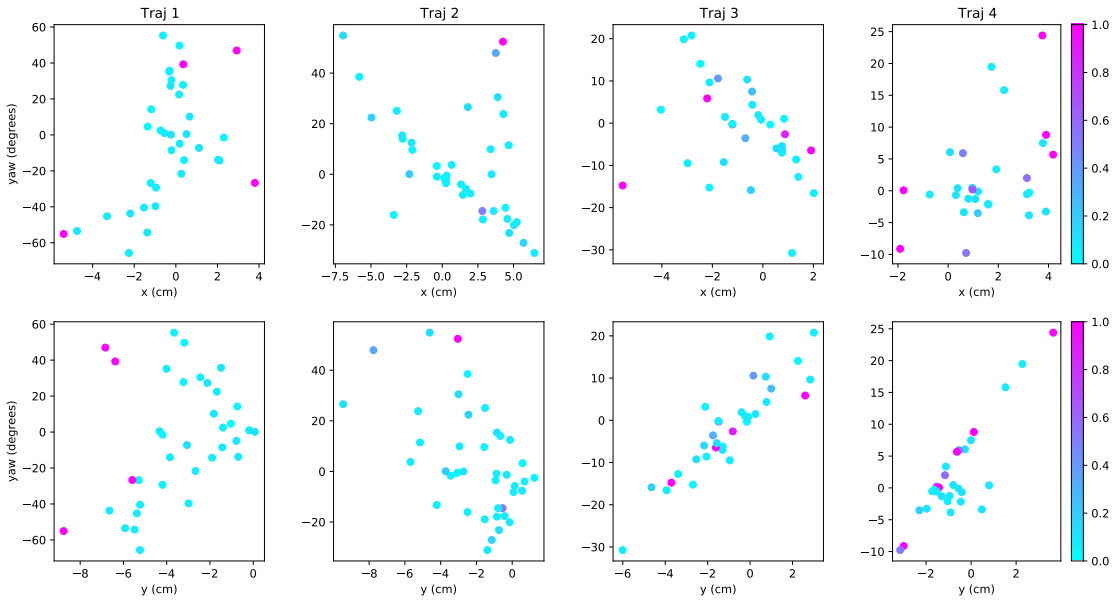


Figure A-3: **Scatter plots show results of our approach on the four different reference trajectory tracking tasks.** The axes of the plots show the deviation of the object starting pose from the initial pose of the demonstration. The top row shows the deviation in the x and yaw axes, while the bottom shows the deviation in the y and yaw axes. Axes are illustrated in Figure 2-4. The color indicates the distance between final and goal poses, lower cost is better. The numerical value is computed as cost $= \frac{\Delta\text{pos}}{3} + \frac{\Delta\text{angle}}{30}$ where $\Delta\text{pos}, \Delta\text{angle}$ are the translational (in centimeters) and angular (in degrees) errors between the object's final position and the goal position. The costs are rescaled and plotted in the range $[0, 1]$. The various reference trajectories, shown in Figure A-2, are of different difficulties, as reflected by the different regions of attraction of the MPC controller. Videos of the closed-loop rollouts can be found at project page.

position. Given that during trials we explicitly chose initial conditions to test the region of attraction of the MPC controller, success rates are not particularly meaningful, as the success rate depends on the initial condition. Table A.2 shows the average translational and angular errors among successful trajectories.

Trajectory	pos (cm)	angle °	success rate	num trials
1	1.23 ± 0.55	5.25 ± 3.99	87.5%	40
2	1.10 ± 0.319	7.32 ± 4.08	89%	36
3	1.16 ± 0.58	3.80 ± 2.54	73%	33
4	1.44 ± 0.30	9.73 ± 5.48	65%	29

Table A.2: **Quantitative results of hardware experiments.** A trial is considered a success rate if the final object position was within 3 cm and 30 degrees of the goal pose. Note that, as shown in Figure A-3 the initial conditions were intentionally chosen to test the region of attraction of our controller, thus the success rates are not meaningful in and of themselves and are included only for completeness. The pos (cm) and angle columns show the deviation of the final object position from the target. Note that the mean and standard deviation are only calculated over the successful trials. This serves to give a sense of the accuracy that can be achieved by using our closed-loop MPC controller.

Appendix B

Learning Particle Dynamics for Manipulating Rigid Bodies, Deformable Objects, and Fluids

B.1 Control Algorithm

Algorithm 1 Control on Learned Dynamics at Time Step t

Input: Learned forward dynamics model Φ
Predicted dynamics graph $\hat{G}_t$
Current dynamics graph G_t
Goal $\mathcal{G}_g$, current estimation of the attributes A
Current control inputs $\hat{u}_{t:T}$
States history $\bar{\mathcal{G}} = \{G_i\}_{i=1...t}$
Time horizon T
Output: Controls $\hat{u}_{t:T}$, predicted next time step $\hat{G}_{t+1}$

Update A by descending with the gradients $\nabla_A \mathcal{L}_{\text{state}}(\hat{G}_t, G_t)$
Forward simulation using the current graph $\hat{G}_{t+1} \leftarrow \Phi(G_t)$
Make a buffer for storing the simulation results $\mathcal{G} \leftarrow \bar{\mathcal{G}} \cup \hat{G}_{t+1}$
for $i = t+1, ..., T-1$ **do**
Forward simulation: $\hat{G}_{i+1} \leftarrow \Phi(\hat{G}_i)$; $\mathcal{G} \leftarrow \mathcal{G} \cup \hat{G}_{i+1}$
end for
Update $\hat{u}_{t:T}$ by descending with the gradients $\nabla_{\hat{u}_{t:T}} \mathcal{L}_{\text{goal}}(\mathcal{G}, \mathcal{G}_g)$
Return $\hat{u}_{t:T}$ and $\hat{G}_{t+1} \leftarrow \Phi(G_t)$

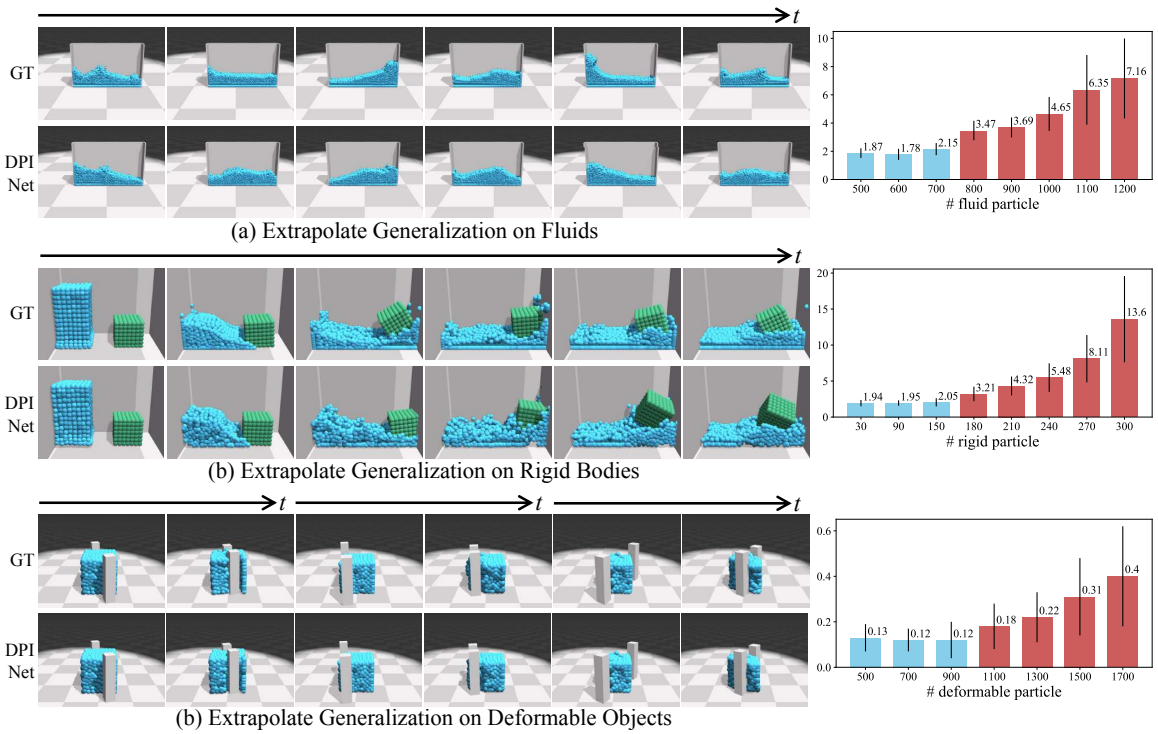


Figure B-1: **Extrapolate generalization on fluids, rigid bodies, and deformable objects.** The performance is evaluated by the MSE ($\times 10^{-2}$) between the ground truth and rollouts from DPI-Nets. The blue bars denote the range of particle numbers that have been seen during training, which indicate interpolation performance. The red bars indicate extrapolation performance that our model can generalize to cases containing two times more particles than cases it has been trained on.

B.2 Generalization on Extrapolation

We show our model's performance on fluids, rigid bodies, and deformable objects with a larger number of particles than they have in the training set. Figure B-1 shows qualitative and quantitative results. Our model scales up well to larger objects.

B.3 Data Generation

The data is generated using NVIDIA FleX. We have developed a Python interface for the ease of generating and interacting with different environments. We will release the code upon publication.

FluidFall. We generated 3,000 rollouts over 120 time steps. The two drops of fluids contain 64 and 125 particles individually, where the initial position of one of the drop in the 3

dimensional coordinates is uniformly sampled between (0.15, 0.55, 0.05) and (0.25, 0.7, 0.15), while the other drop is uniformly sampled between (0.15, 0.1, 0.05) and (0.25, 0.25, 0.15).

BoxBath. We generated 3,000 rollouts over 150 time steps. There are 960 fluid particles and the rigid cube consist particles ranging between 27 and 150. The fluid particle block is initialized at (0, 0, 0), and the initial position of the rigid cube is randomly initialized between (0.45, -0.0155, 0.02) to (1.2, -0.0155, 0.4).

FluidShake. We generated 2,000 rollouts over 300 time steps. The height of the box is 1.0 and the thickness of the wall is 0.025. For the initial fluid cuboid, the number of fluid particles is uniformly sampled between 10 and 12 in the x direction, between 15 and 20 in the y direction, 3 in the z direction. The box is fixed in the y and z direction, and is moving freely in the x direction. We randomly place the initial x position between -0.2 to 0.2. The sampling of the speed is implemented as $\mathrm{v} = \mathrm{v} + \mathrm{rand}(-0.15, 0.15) - 0.1\mathrm{x}$, in order to encourage motion smoothness and moving back to origin, where speed v is initialized as 0.

RiceGrip. We generated 5,000 rollouts over 30 time steps. We randomize the size of the initial "rice" cuboid, where the length of the three sides is uniformly sampled between 8.0 and 10.0. The material property parameters `clusterStiffness` is uniformly sampled between 0.3 and 0.7, `clusterPlasticThreshold` is uniformly sampled between 1e-5 and 5e-4, and `clusterPlasticCreep` is uniformly sampled between 0.1 and 0.3. The position of the gripper is randomly sampled within a circle of radius 0.5. The orientation of the gripper is always perpendicular to the line connecting the origin to the center of the gripper and the close distance is uniformly sampled between 0.7 to 1.0.

Of all the generated data, 90% of the rollouts are used for training, and the rest 10% are used for validation.

B.4 Training Details

The models are implemented in PyTorch, and are trained using Adam optimizer [Kingma and Ba, 2015] with a learning rate of 0.0001. The number of particles and relations might be different at each time step, hence we use a batch size of 1, and we update the weights of the networks once every 2 forward rounds.

The neighborhood d is set as 0.08, and the propagation step L is set as 2 for all four environments. For hierarchical modeling, it does not make sense to propagate more than one time between leaves and roots as they are disjoint particle sets, and each propagation stage between them only involves one-way edges; hence $\phi_{\text{LeafToLeaf}}$ uses $L = 2$. $\phi_{\text{LeafToRoot}}$ uses $L = 1$. $\phi_{\text{RootToRoot}}$ uses $L = 2$, and $\phi_{\text{RootToLeaf}}$ uses $L = 1$.

For all propagation networks used below, the object encoder f_O^{enc} is an MLP with two hidden layers of size 200, and outputs a feature map of size 200. The relation encoder f_R^{enc} is an MLP with three hidden layers of size 300, and outputs a feature map of size 200. The propagator f_O and f_R are both MLP with one hidden layer of size 200, in which a residual connection is used to better propagate the effects, and outputs a feature map of size 200. The propagators are shared within each stage of propagation. The motion predictor f_O^{output} is an MLP with two hidden layers of size 200, and output the state of required dimension. ReLU is used as the activation function.

FluidFall. The model is trained for 13 epochs. The output of the model is the 3 dimensional velocity, which is multiplied by Δt and added to the current position to do rollouts.

BoxBath. In this environment, four propagation networks are used due to the hierarchical modeling and the number of roots for the rigid cube is set as 8. We have two separate motion predictor for fluids and rigid body, where the fluid predictor output velocity for each fluid particle, while the rigid predictor takes the mean of the signals over all its rigid particles as input, and output a rigid transformation (rotation and translation). The model is trained for 5 epochs.

FluidShake. Only one propagation network is used in this environment, and the model is trained for 5 epochs.

RiceGrip. Four propagation networks are used due to the hierarchical modeling, and the number of roots for the "rice" is set as 30. The model is trained for 20 epochs.

B.5 Control Details

N_{sample} is chosen as 20 for all three cases, where we sample 20 random control sequences, and choose the best performing one as evaluated using our learned model. The evaluation is based on the Chamfer distance between the controlling result and the target configuration.

FluidShake. In this environment, the control sequence is the speed of the box along the x axis. The method to sample the candidate control sequence is the same as when generating training data of this environment. After selected the best performing control sequence, we first use RMSprop optimizer to optimize the control inputs for 10 iterations using a learning rate of 0.003. We then use model-predictive control to apply the control sequence to the FleX physics engine using Algorithm 1.

RiceGrip. In this environment, we need to come up with a sequence of grip configurations, where each grip contains positions, orientation, and closing distance. The method to sample the candidate control sequence is the same as when generating training data of this environment. After selected the best performing control sequence, we first use RMSprop optimizer to optimize the control inputs for 20 iterations using a learning rate of 0.003. We then use model-predictive control to apply the control sequence to the FleX physics engine using Algorithm 1.

RiceGrip in Real World. In this environment, we need to come up with a sequence of grip configurations, where each grip contains positions, orientation, and closing distance. The method to sample the candidate control sequence is the same as when generating training data of **RiceGrip**, and N_{fill} is chosen as 768. Different from the previous case, the physical parameters are always unknown and has to be estimated online. After selected the best performing control sequence, we first use RMSprop optimizer to optimize the control inputs for 20 iterations using a learning rate of 0.003. We then use model-predictive control to apply the control sequence to the real world using Algorithm 1.

Appendix C

Learning Latent-Space Dynamics with Neural Radiance Field

C.1 Model Details

In the decoder model, we use a similar network architecture as the NeRF paper [Mildenhall et al., 2020]. As shown in Figure C-1, we send a 3D point $\boldsymbol{x} \in \mathbb{R}^3$, a camera ray $\boldsymbol{d} \in \mathbb{R}^3$, and a state feature representation $\mathbf{s}_t$ into a fully-connected network and output the corresponding density σ_t and RGB color $\boldsymbol{c}_t$.

C.2 Environment Details

In the **FluidPour** environment, we generated $1,000$ trajectories for training. Each trajectory has 300 frames with 20 camera views sampled around the objects with a fixed distance towards the world origin. The action space for the control task is the position and tilting angle of the cup, which are randomly generated when constructing the training set.

In the **FluidShake** environment, we generated $1,000$ trajectories for training. Each trajectory has 300 frames with 20 camera views sampled around the objects with a fixed distance towards the world origin. The action space for the control task is the 2D location of the container in the world coordinate, which is also randomly generated when constructing the training set.

Figure C-2 shows some example visual observations for FluidPour and FluidShake used during training. We also include some example images from viewpoints outside the training

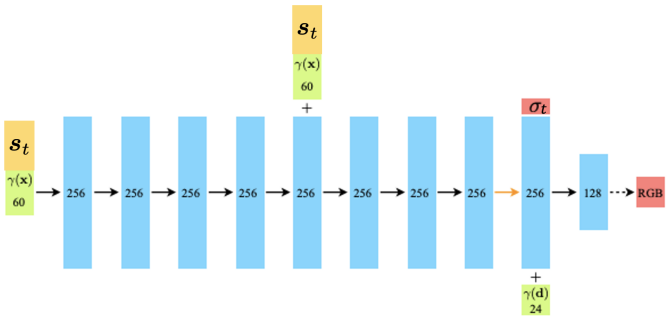


Figure C-1: **A visualization of our decoder network architecture.** All layers are standard fully-connected layers, where the black arrows indicate layers with ReLU activations, orange arrows indicate layers with no activation, dashed black arrows indicate layers with sigmoid activation, and "+" denotes vector concatenation. We concatenate the positional encoding of the input location $\gamma(\boldsymbol{x})$ and our learned state representation $\boldsymbol{s}_t$, and pass them through 8 fully-connected ReLU layers, each with 256 channels. We follow the NeRF [Mildenhall et al., 2020] architecture and include a skip connection that concatenates this input to the fifth layer's activation. An additional layer outputs the volume density σ_t (which is rectified using a ReLU to ensure that the output volume density is nonnegative) and a 256-dimensional feature vector. This feature vector is concatenated with the positional encoding of the input viewing direction $\gamma(\boldsymbol{d})$, and is processed by an additional fully-connected ReLU layer with 128 channels. A final layer (with a sigmoid activation) outputs the emitted RGB radiance at position $\boldsymbol{x}$, as viewed by a ray with direction $\boldsymbol{d}$, given the current 3D state representation $\boldsymbol{s}_t$.

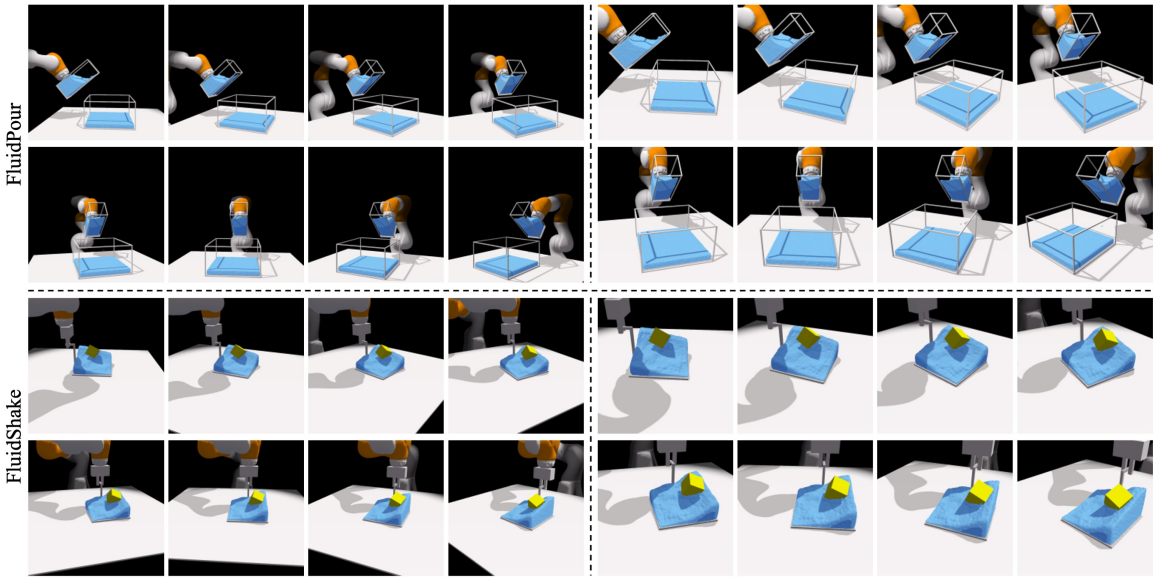
distribution, which are then used to evaluate our model's extrapolated generalization ability.

In **RigidStack**, we generated 800 trajectories for training. Each trajectory has 80 frames with 20 camera views sampled around the objects with a fixed distance towards the world origin.

In **RigidDrop**, we generated 1,000 trajectories for training. Each trajectory has 50 frames with 20 camera views sampled around the objects with a fixed distance towards the world origin.

C.3 Training Details

For the encoder and decoder model described in Figure 4-2, we use the Adam optimizer with the initial learning rate $5e^{-4}$ and decreased to $5e^{-5}$ for all the experiments. The batch size is 2. The hyperparameters in the decoder are the same as the original NeRF model [Mildenhall et al., 2020] except the near and far distance between the objects and cameras are different in our environments. In our FluidPour environment, we have near = 2.0 and far = 9.5. In our FluidShake environment, we have near = 2.0 and far = 7.0. In our



(a) Example viewpoints used for training

(b) Example viewpoints outside training distribution used for goal specification in control

Figure C-2: **Comparison between the viewpoints used for training and the subsequent viewpoint extrapolation experiments.** (a) We show some example images that the model used during training. For both environments, the camera is placed from a fixed distance and facing towards the world origin. (b) To evaluate the model's ability for viewpoint extrapolation, i.e., processing visual observations from viewpoints that are outside the training distribution, we generate another set of viewpoints that are closer, higher, and facing more downwards. It is clear from the figure that the images from viewpoints used during training are very different from the ones used for viewpoint extrapolation when measured using pixel difference. Therefore, it is essential to build a model that can directly reason over 3D to provide the desired extrapolation generalization ability. Although the model has access to visual observations from multiple cameras during training, it can only observe the environment from one camera when performing the downstream control task.

RigidStack environment, we have near $= 2.0$ and far $= 7.0$. In our FluidPour environment, we have near $= 2.0$ and far $= 6.0$.

C.4 Control Details

As discussed in Section 4.4.1, we use model-predictive path integral (MPPI) [Williams et al., 2015a] to solve the MPC problem. MPPI is a sampling-based, gradient-free optimizer that considers temporal coordination between time steps when sampling action trajectories. At time t, the algorithm first samples M action sequences based on the current actions $\boldsymbol{a}_t, \ldots, \boldsymbol{a}_{T-1}$ via $\hat{\boldsymbol{a}}_h^k = \boldsymbol{a}_h + \boldsymbol{n}_h^k, k \in \{1, \ldots, M\}, h \in \{t, \ldots, T-1\}$. Each noise sample $\boldsymbol{n}_h^k$, denoting the noise value at the h^{th} time step of the k^{th} trajectory, is generated using filtering

coefficient β as the following:

$$\begin{aligned} &\boldsymbol{u}_h^k \sim \mathcal{N}(0, \Sigma) \quad \forall k \in \{1, \ldots, M\}, h \in \{t, \ldots, T-1\} \\ &\boldsymbol{n}_h^k = \beta \cdot \boldsymbol{u}_h^k + (1-\beta) \cdot \boldsymbol{n}_{h-1}^k, \text{where } \boldsymbol{n}_{h<t}^k = 0. \end{aligned} \tag{C.1}$$

We then roll them out in parallel using the learned model on the GPU to derive $\hat{\boldsymbol{s}}_T^k, k \in \{1, \ldots, M\}$, and then re-weight the trajectories according to the reward to update the action sequence using a reward-weighting factor γ: $\boldsymbol{a}_h = (\sum_{k=1}^M \exp(\gamma \cdot R^k) \cdot \hat{\boldsymbol{a}}_h^k) / (\sum_{k=1}^M \exp(\gamma \cdot R^k)), h \in \{t, \ldots, T-1\}$, where $R^k = -\|\hat{\boldsymbol{s}}_T^k - \boldsymbol{s}^{\text{goal}}\|_2^2$. This procedure is repeated for L iterations at which point the best action sequence is selected.

The number of updating iteration for the auto-decoding test-time optimization K is 500. The number of sampled trajectories M during MPPI optimization is set to $1,000$. The number of iterations L for updating the action sequence is set to 100 for the first time step, and 10 for the subsequent control steps to maintain a better trade-off between efficiency and effectiveness. The reward-weighting factor γ is set to 50 and the filtering coefficient β is specified as 0.7. The control horizon T is set as 80 both for FluidPour and FluidShake. The hyperparameters are the same for all compared methods.

C.5 Additional Experimental Results

C.5.1 Auto-Decoder Test-Time Optimization for Viewpoint Extrapolation

Figure C-3 shows the qualitative results on auto-decoding test-time optimization. When encountering an image from a viewpoint outside the training distribution, this mechanism can help us derive a better representation of the scene that holds a more accurate description of the 3D contents. The obtained representation after the optimization can then be used as the goal embedding $\boldsymbol{s}^{\text{goal}}$ in Equation 4.4 that the agent needs to achieve.

C.5.2 Validation of the Dynamics Prediction on Real-World Data

We further evaluate our model's dynamics prediction ability by conducting experiments on real-world data. As shown in Figure C-4, we use four D415 RGBD cameras to record a human subject pouring water from one cup to another. The cameras are calibrated and synchronized with the frequency of 15 Hz. We recorded 50 pouring episodes of length 15

seconds, resulting in a dataset of 43,400 frames. We use the first 45 episodes for training and the remaining for testing.

To obtain the action, i.e., the movement of the cup that pours water out, we attached an AprilTag [Olson, 2011] on the cup to obtain the 6 DoF pose of the cup at each time step. From the supplementary video, you can see that our model can make open-loop future predictions on the testing trajectories in the representations space, i.e., given a latent scene representation of the current time step, the subsequent action sequence, our dynamic model can accurately predict the evolution of the latent scene representation and render the corresponding frames that closely resemble the ground truth.

C.5.3 Comparison With PID That Only Matches the Robot's State

To show the necessity of modeling the fluid dynamics in our task, we include an additional baseline that uses PID control [Franklin et al., 2002] to reach the robot state in the target image without worrying about the fluid. Note that this baseline uses additional information, including the ground truth state of the robot in both the current and the target image.

Our supplementary video shows a controlling trial where the goal is to leave a small amount of fluid in the container at the end of the control episode. Naively matching the container's position and orientation will not work, as shown in the PID baseline that the top container did not pour any fluid into the bottom container. In contrast, our model first learns to pour a certain amount of fluid out and then tilt the container back to match the target configuration. We further use the Chamfer distance to measure the models' performance in matching the fluid shape in the 3D point space, where our method significantly outperforms the PID baseline (Ours: 0.048237 vs. PID: 0.085727).

C.5.4 Nearest Neighbor Search Using the Learned Representations

When the goal image is specified from a viewpoint different from the agent's view, to ensure the planning problem defined in Equation 4.4 still work, it is essential that the distance in the learned feature space reflects the distance in the actual 3D space, i.e., scenes that are more similar in the real 3D space should be closer in the learned feature space, even if the visual observations are captured from different viewpoints. We visualize the nearest neighbor results in Figure C-5. Given a query image, we search its nearest neighbors based on their state representation $\boldsymbol{s}$ (introduced in Section 4.3.1). Even if the images look quite

Env	Ours	NN in pixel space	Random
FluidPour	**1.772718**	147.971993	99.903261
FluidShake	**2.584928**	132.467998	99.646617

Table C.1: **Quantitative results on nearest neighbor (NN) search from different viewpoints.** We calculate the accuracy of finding NN using the L1 distance between the ground truth and retrieved time indexes. Our method measures the distance in the learned representation space, which delivers the best performance, and the retrieved frame is, on average, around two time steps away from the ground truth.

different from each other when measured in pixel difference, the learned 3D-aware scene representations can retrieve reasonable neighborhood images that share similar 3D contents, indicating that the learned 3D-aware scene representations hold a good understanding of the real 3D scene and are invariant to viewpoint variations.

We conducted additional experiments to quantitatively measure the accuracy in finding the nearest neighbors (NN) across viewpoints of our proposed method. The results are shown in Table C.1. Specifically, for each query frame, the model is asked to find the closest frame from a randomly selected viewpoint from a trajectory with 300 frames. We measure the accuracy using the L1 distance between the time indexes. Randomly selecting the closest frame leads to an averaged distance of 100 times steps between the selected frame and the ground truth frame. Selecting based on the pixel difference is even worse. Our method can accurately find the nearest neighbor. The average time step difference between the selected frame and the ground truth frame is 1.772718 in FluidPour and 2.584928 in FluidShake.

C.6 Limitations and Future Works

Similar to many other data-driven methods, our model can deliver reasonable performance in regions well-supported by the data. However, for cases that our model has never seen before, e.g., more containers than during training, we wouldn't expect our model to generalize. Potential solutions may include (1) adding the examples and increasing the diversity of the training set or (2) using a more structured/compositional representation space instead of a single vector as in our model, which we leave for future works.

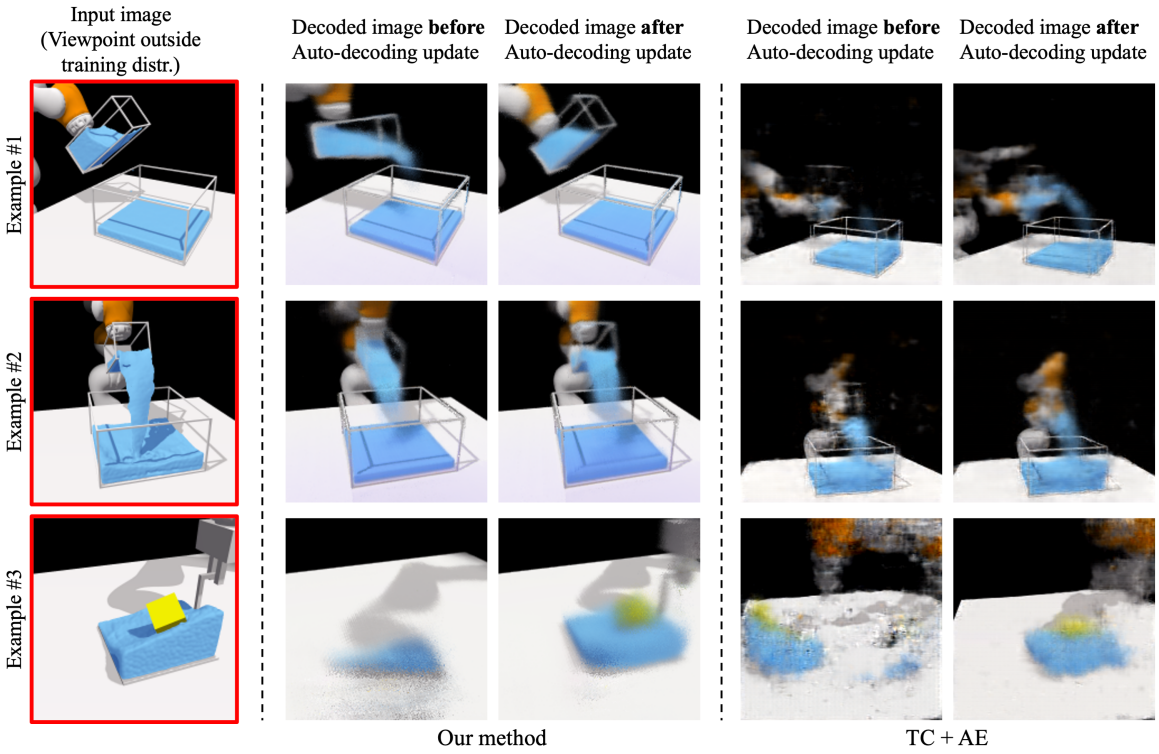
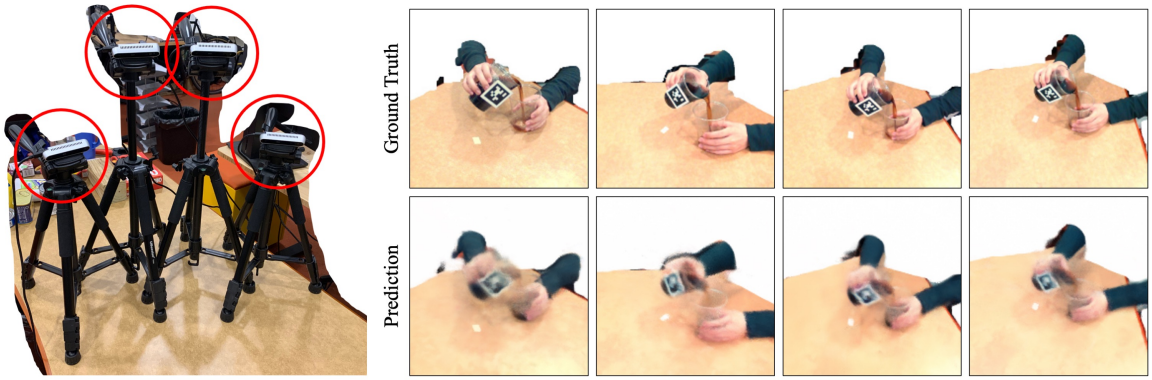


Figure C-3: **Qualitative results on auto-decoding test-time optimization.** Following from the pipeline illustrated in Figure 4-3b, if the input image I_t is outside the training distribution as shown on the left column, the encoder won't be able to generate the most accurate state representation. When passing the predicted state embedding $\boldsymbol{s}_t$ and the same viewpoint as the input to the decoder, the generated image does not match the underlying scene as shown in the second column. We then calculate the L2 distance of the pixels between the generated image and true observation, backpropagating the gradient until the state representation and making updates to $\boldsymbol{s}_t$ using SGD. As discussed in Section 4.4.2, the translational equivariance nature of the decoder allows it to effectively optimize the latent representation to make it better reflect the 3D contents in the scene. After the optimization, the generated visual observation is much closer to the ground truth, as shown in the third column. On the contrary, the vanilla autoencoder that uses a CNN-based decoder won't be able to capture the underlying scene even with test-time auto-decoding optimization, as shown on the right.



(a) Data capture setup (four D415 cameras) (b) Comparison between our model's prediction and the ground truth

Figure C-4: **Dynamics prediction using real-world data.** (a) We build a data recording set up containing four D415 RGBD cameras to record a human subject pouring water from one cup to another and then use the recorded data to evaluate our model's dynamics prediction ability. (b) We show a side-by-side comparison between the ground truth data and our model's prediction when predicting the future from the four camera views. Our model correctly identifies when the fluids pour out and start to fill the bottom container. Please see our supplementary video for better visualizations.

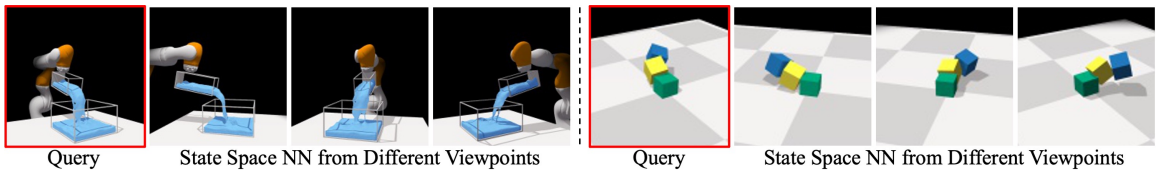


Figure C-5: **Nearest neighbor (NN) results using our learned state representation.** Given a query image (red boundary), we search its nearest neighbors based on their state representation. Our learned scene representations can retrieve reasonable neighbor images, indicating that our state representations retain a good estimation of the contents inside the 3D scene and are invariant to camera poses.

Appendix D

Learned Structured World Models for Causal Discovery From Videos

D.1 Model Details

D.1.1 Unsupervised Keypoint Detection From Videos

The perception module maps the input images into a set of keypoints in an unsupervised way. Any unsupervised keypoint detection methods that can track the components consistently overtime should suit our use case, and there have been many recently-proposed methods that could serve this purpose [Suwajanakorn et al., 2018, Jakab et al., 2018, Zhang et al., 2018, Manuelli et al., 2021]. In this work, we use the technique developed by Kulkarni et al. [2019].

As described in Section 6.2.1 of the main paper, we use reconstruction loss over the pixels to encourage the keypoints to spread over the foreground of the image. During training, it takes in a source image I^{src} and a target image I^{tgt} sampled from the dataset, and passes them through a feature extractor $f_\omega^{\mathcal{V}}$ and a keypoint detector $f_\theta^{\mathcal{V}}$. The model then uses an operation called *transport* to construct a new feature map using a set of local features indicated by the detected keypoints:

$$\Phi(I^{\text{src}}, I^{\text{tgt}}) \triangleq (1 - \mathcal{H}_{f_\theta^{\mathcal{V}}(I^{\text{src}})}) \cdot (1 - \mathcal{H}_{f_\theta^{\mathcal{V}}(I^{\text{tgt}})}) \cdot f_\omega^{\mathcal{V}}(I^{\text{src}}) + \mathcal{H}_{f_\theta^{\mathcal{V}}(I^{\text{tgt}})} \cdot f_\omega^{\mathcal{V}}(I^{\text{tgt}}), \tag{D.1}$$

where $\mathcal{H}$ is a heatmap image containing fixed-variance isotropic Gaussians around each of the

N points specified by $f_\theta^{\mathcal{V}}$ (Figure D-1). The model then passes the feature map $\Phi(I^{\text{src}}, I^{\text{tgt}})$ through a refiner network to get the reconstruction, $\hat{I}^{\text{tgt}}$. We optimize the parameters in the feature extractor, keypoint detector and refiner by minimizing a pixel-wise L_2 loss, $\mathcal{L}_{\text{rec}} = \|I^{\text{tgt}} - \hat{I}^{\text{tgt}}\|$, using stochastic gradient descent.

D.1.2 Graph Neural Networks As the Spatial Encoder

Graph neural networks act as a building block in our model to capture the interactions between different keypoints and generate object- and relation-centric embeddings. Here, we describe the specific formulation of the graph neural network we used in our inference and dynamics modules.

For a set of N keypoints, we construct a directed graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where vertices $\mathcal{V} = \{\boldsymbol{o}_i\}$ represent the information on the keypoints and edges $\mathcal{E} = \{(\boldsymbol{o}_i, \boldsymbol{o}_j, \boldsymbol{g}_{ij})\}$ represent the directed relation pointing from j to i. $\boldsymbol{g}_{ij}$ is the associated edge attributes.

Our graph neural network employs a similar structure as the Interaction Networks (IN) [Battaglia et al., 2016] to generate the embeddings for the objects and the relations:

$$\boldsymbol{h}_{ij} = f^{\text{rel}}(\boldsymbol{o}_i, \boldsymbol{o}_j, \boldsymbol{g}_{ij}) \qquad \text{for each edge } (\boldsymbol{o}_i, \boldsymbol{o}_j, \boldsymbol{g}_{ij}) \in \mathcal{E}, \tag{D.2}$$

$$\boldsymbol{h}_i = f^{\text{obj}}(\boldsymbol{o}_i, \sum_{j \in \mathcal{N}_i} \boldsymbol{h}_{ij}) \qquad \text{for each node } \boldsymbol{o}_i \in \mathcal{V}, \tag{D.3}$$

where f^{obj} and f^{rel} are object and relation encoders respectively. $\mathcal{N}_i$ denotes all vertices that have an edge pointing to object i. $\{\boldsymbol{h}_i\}$ and $\{\boldsymbol{h}_{ij}\}$ are the derived object and relation embeddings individually. In practice, we usually propagate the node and edge information over the graph multiple times to improve the expressiveness of the model [Sanchez-Gonzalez et al., 2018, Li et al., 2019b].

The graph neural network, denoted as ϕ, aggregates the spatial information spanned by the keypoints, passes the information along the edges, and outputs embeddings for the nodes and edges, i.e., $(\{\boldsymbol{h}_i\}, \{\boldsymbol{h}_{i,j}\}) = \phi(\mathcal{V}, \mathcal{E})$. Please see our main paper for how we instantiate ϕ as a submodule in the inference and the dynamics modules.

D.2 Environment Details

D.2.1 Multi-Body Interaction

We use the Pymunk simulator to generate $5,000$ episodes of 500 frames, among which 250 episodes are reserved for testing, and the remaining goes to the training set. At the beginning of each episode, we randomly assign the balls in different positions. For each pair of balls, there is a one-third probability that they are connected by nothing, rigid rod, and spring. The stiffness of the spring relation is set to 20, and we randomly sample the rest length between $[20, 120]$. For the rigid relation, we allow the connected two balls to move freely in a small fixed window on their opposing direction, e.g., if the rigid relationship is of length 50, the distance between the two balls can vary between 45 to 55. This treatment will force the model to infer the length of the rigid relation instead of naively exploiting the distance between the two balls.

D.2.2 Fabric Manipulation

We generate $2,000$ episodes of 300 frames using the NVIDIA FleX simulator [Macklin et al., 2014]. Similar to the Multi-Body environment, we reserve 200 episodes for testing and use the remaining for training our model. As shown in Figure D-1, we build fabrics of three different shapes: a shirt, pants, and a towel, where we also vary the shape of the fabrics like the length of the pant leg or the height and width of the towel. To deform the fabrics and move them around, we apply forces on the contour of the fabric.

D.3 Implementation Details

Our implementation is based on PyTorch [Paszke et al., 2019], and each instance of the model is trained using one NVIDIA TITAN Xp graphics card.

D.3.1 Unsupervised Keypoint Detection

We employ a similar encoder-decoder structure as described in Kulkarni et al. [2019]. Both the keypoint detector, $f_\theta^{\mathcal{V}}$, and the feature extractor, $f_\omega^{\mathcal{V}}$, have 5 blocks of convolutional layers that reduce the height and width of the image into a quarter of their original size. The output of the keypoint detector has N channels, representation the confidence map of

the N keypoints, over which $f_\theta^{\mathcal{V}}$ computes the exact location of each keypoint via spatial softmax. We use the operation describe in Equation D.1 to get the feature maps $\Phi(I^{\text{src}}, I^{\text{tgt}})$. The refiner network, consisting of a few transpose convolutional operators, transforms the features map back to the original size of the target image.

We optimize $\mathcal{L}_{\text{rec}}$ using Adam optimizer [Kingma and Ba, 2015] with a learning rate of 0.001 for about 240k iterations.

D.3.2 Predicting the Directed Edge Set Using the Inference Module

We use simple multilayer perceptron (MLP) to instantiate the object encoder, f^{obj}, and the relation encoder, f^{rel}. To aggregate the temporal information, we use three blocks of convolutional layers for CNN^{obj} and CNN^{rel}. The use of convolutional operators allows the model to handle time series of different lengths, and the output of the CNNs is fed through a max-pooling layer to compute a fixed-dimensional feature vector.

D.3.3 Joint Optimization of the Inference Module and the Dynamics Module

We train the inference module, $f_\phi^{\mathcal{E}}$, and the dynamics module, $f_\psi^{\mathcal{D}}$, jointly by optimizing the loss function defined in Section 6.2.5 using stochastic gradient descent via Adam optimizer with a learning rate of 0.0001 in the Multi-Body environment and 0.0005 in the Fabrics environment for about 300k iterations.

For the exact network architecture and more details in the training procedures of the individual modules, please refer to our released code.

D.4 Additional Experimental Results

D.4.1 Unsupervised Keypoint Detection

The combination of the keypoint-based bottleneck layer and the downstream reconstruction task allows the perception module to extract temporally-consistent keypoints dispersing over the images' foreground. The model accurately tracks the movement of the objects and can naturally handle deformable objects. Figure D-1 shows some more qualitative examples of our perception module in both the Multi-Body and the Fabric environments.

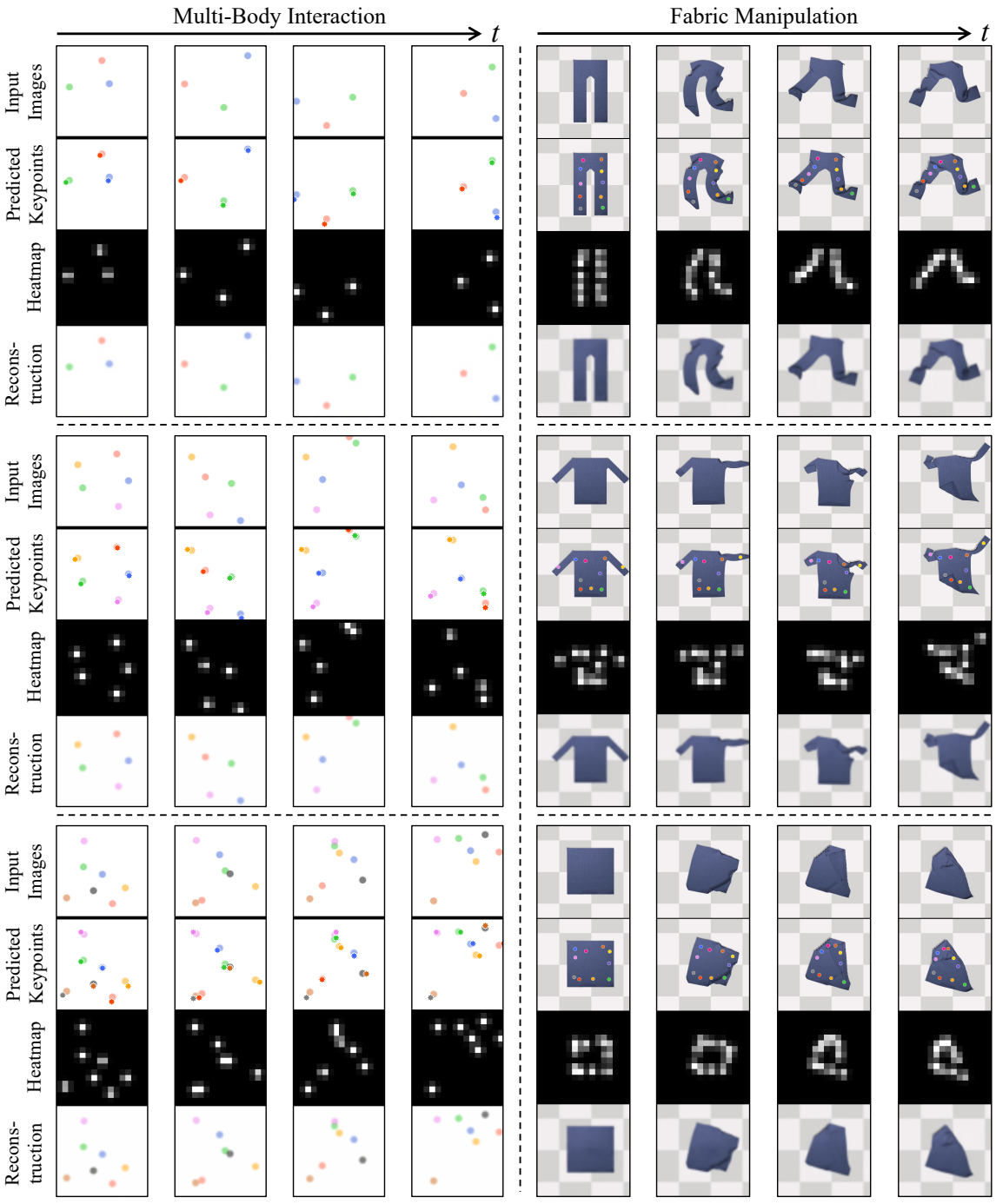


Figure D-1: **Unsupervised keypoint detection.** We show some more qualitative results of our perception module and visualize the intermediate results. In each block, the first row shows the input images, and the second row illustrates an overlay between the predicted keypoints and the image. The third and the fourth row show the intermediate results - heatmap spanned by the keypoints and the reconstructed target image.

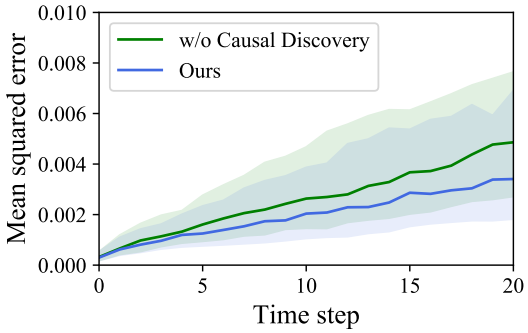


Figure D-2: **Future prediction in the Fabrics environment.** When making long-term predictions into the future, our method outperforms the baseline that does not perform causal discovery.

D.4.2 Future Prediction in the Fabric Environment

Figure D-2 shows a comparison between our model and the baseline, which is the same as our model except that it does not contain an inference module to perform causal discovery. Our model can make more accurate future predictions, indicating the importance of an accurate modeling of the causal mechanisms in the underlying physical system.

Bibliography

Ian Abraham, Gerardo De La Torre, and Todd D Murphey. Model-based control using koopman operators. In *RSS*, 2017.

Jack A Adams. A closed-loop theory of motor learning. *Journal of motor behavior*, 3(2): 111–150, 1971.

Pulkit Agrawal, Ashvin V Nair, Pieter Abbeel, Jitendra Malik, and Sergey Levine. Learning to poke by poking: Experiential learning of intuitive physics. In *Advances in Neural Information Processing Systems*, pages 5074–5082, 2016.

Bok Y Ahn, Eric B Duoss, Michael J Motala, Xiaoying Guo, Sang-Il Park, Yujie Xiong, Jongseung Yoon, Ralph G Nuzzo, John A Rogers, and Jennifer A Lewis. Omnidirectional printing of flexible, stretchable, and spanning silver microelectrodes. *Science*, 323(5921): 1590–1593, 2009.

Ferran Alet, Erica Weng, Tomás Lozano-Pérez, and Leslie Pack Kaelbling. Neural relational inference with fast modular meta-learning. In *Advances in Neural Information Processing Systems*, pages 11804–11815, 2019.

Hassan Arbabi and Igor Mezic. Ergodic theory, dynamic mode decomposition, and computation of spectral properties of the koopman operator. *SIAM Journal on Applied Dynamical Systems*, 16(4):2096–2126, 2017.

Hassan Arbabi, Milan Korda, and Igor Mezic. A data-driven koopman model predictive control framework for nonlinear flows. In *CDC*, 2018.

Mohammad Babaeizadeh, Chelsea Finn, Dumitru Erhan, Roy H. Campbell, and Sergey Levine. Stochastic variational video prediction. In *ICLR*, 2018.

Talis Bachmann. Identification of spatially quantised tachistoscopic images of faces: How many pixels does it take to carry identity? *European Journal of Cognitive Psychology*, 3 (1):87–103, 1991.

Shiv S Baishya and Berthold Bäuml. Robust material classification with a tactile skin using deep learning. In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 8–15. IEEE, 2016.

Chiara Bartolozzi, Lorenzo Natale, Francesco Nori, and Giorgio Metta. Robots with a sense of touch. *Nature materials*, 15(9):921–925, 2016.

Christopher J Bates, Ilker Yildirim, Joshua B Tenenbaum, and Peter Battaglia. Modeling human intuitions about liquid flow with particle-based simulation. *PLoS Computational Biology*, 15(7):e1007210, 2019.

Peter W Battaglia, Jessica B Hamrick, and Joshua B Tenenbaum. Simulation as an engine of physical scene understanding. *PNAS*, 110(45):18327–18332, 2013.

Peter W. Battaglia, Razvan Pascanu, Matthew Lai, Danilo Rezende, and Koray Kavukcuoglu. Interaction networks for learning about objects, relations and physics. In *NIPS*, 2016.

Peter W Battaglia, Jessica B Hamrick, Victor Bapst, Alvaro Sanchez-Gonzalez, Vinicius Zambaldi, Mateusz Malinowski, Andrea Tacchetti, David Raposo, Adam Santoro, Ryan Faulkner, et al. Relational inductive biases, deep learning, and graph networks. *arXiv:1806.01261*, 2018.

David Bau, Bolei Zhou, Aditya Khosla, Aude Oliva, and Antonio Torralba. Network dissection: Quantifying interpretability of deep visual representations. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 6541–6549, 2017.

Catherine E Bauby and Arthur D Kuo. Active control of lateral balance in human walking. *Journal of biomechanics*, 33(11):1433–1440, 2000.

Maria Bauza, Francois R Hogan, and Alberto Rodriguez. A data-efficient approach to precise and controlled pushing. In *Conference on Robot Learning*, pages 336–345. PMLR, 2018.

Daniel Bear, Chaofei Fan, Damian Mrowca, Yunzhu Li, Seth Alter, Aran Nayebi, Jeremy Schwartz, Li F Fei-Fei, Jiajun Wu, Josh Tenenbaum, et al. Learning physical graph representations from visual scenes. *Advances in Neural Information Processing Systems*, 33:6027–6039, 2020.

Marc G Bellemare, Yavar Naddaf, Joel Veness, and Michael Bowling. The arcade learning environment: An evaluation platform for general agents. *Journal of Artificial Intelligence Research*, 47:253–279, 2013.

James Bradbury, Stephen Merity, Caiming Xiong, and Richard Socher. Quasi-recurrent neural networks. In *ICLR*, 2017.

Alejandro L Briseno, Stefan CB Mannsfeld, Mang M Ling, Shuhong Liu, Ricky J Tseng, Colin Reese, Mark E Roberts, Yang Yang, Fred Wudl, and Zhenan Bao. Patterning organic single-crystal transistor arrays. *Nature*, 444(7121):913–917, 2006.

Eric E Brodie and Helen E Ross. Sensorimotor mechanisms in weight discrimination. *Perception & Psychophysics*, 36(5):477–481, 1984.

Daniel Bruder, Brent Gillespie, C David Remy, and Ram Vasudevan. Modeling and control of soft robots using the koopman operator and model predictive control. In *RSS*, 2019a.

Daniel Bruder, C David Remy, and Ram Vasudevan. Nonlinear system identification of soft robot dynamics using koopman operator theory. In *ICRA*, 2019b.

Steven L Brunton, Bingni W Brunton, Joshua L Proctor, and J Nathan Kutz. Koopman invariant subspaces and finite linear representations of nonlinear dynamical systems for control. *PloS one*, 11(2):e0150171, 2016.

Berk Calli, Arjun Singh, James Bruce, Aaron Walsman, Kurt Konolige, Siddhartha Srinivasa, Pieter Abbeel, and Aaron M Dollar. Yale-cmu-berkeley dataset for robotic manipulation research. *The International Journal of Robotics Research*, 36(3):261–268, 2017. doi: 10.1177/0278364917700714. URL https://doi.org/10.1177/0278364917700714.

Eduardo F Camacho and Carlos Bordons Alba. *Model predictive control*. Springer Science & Business Media, 2013.

Giorgio Cannata, Marco Maggiali, Giorgio Metta, and Giulio Sandini. An embedded artificial skin for humanoid robots. In *2008 IEEE International conference on multisensor fusion and integration for intelligent systems*, pages 434–438. IEEE, 2008.

Michael B Chang, Tomer Ullman, Antonio Torralba, and Joshua B Tenenbaum. A compositional object-based approach to learning physical dynamics. In *ICLR*, 2017.

Sheng Chen, SA Billings, and PM Grant. Non-linear system identification using neural networks. *International Journal of Control*, 51(6):1191–1214, 1990.

Gordon Cheng, Emmanuel Dean-Leon, Florian Bergner, Julio Rogelio Guadarrama Olvera, Quentin Leboutet, and Philipp Mittendorfer. A comprehensive realization of robot skin: Sensors, sensing, control, and applications. *Proceedings of the IEEE*, 107(10):2034–2051, 2019.

David Maxwell Chickering. Optimal structure identification with greedy search. *Journal of machine learning research*, 3(Nov):507–554, 2002.

Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*, 2014.

Alex Chortos, Jia Liu, and Zhenan Bao. Pursuing prosthetic electronic skin. *Nature materials*, 15(9):937–950, 2016.

Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. In *NIPS Workshop*, 2014.

Diego Colombo, Marloes H Maathuis, Markus Kalisch, and Thomas S Richardson. Learning high-dimensional dags with latent and selection variables. In *UAI*, page 850, 2011.

Brian Curless and Marc Levoy. A volumetric method for building complex models from range images. In *SIGGRAPH*, 1996.

Ravinder S Dahiya, Giorgio Metta, Maurizio Valle, and Giulio Sandini. Tactile sensing—from humans to humanoids. *IEEE transactions on robotics*, 26(1):1–20, 2009.

Tommaso D'Alessio. Measurement errors in the scanning of piezoresistive sensors arrays. *Sensors and Actuators A: Physical*, 72(1):71–76, 1999.

Filipe de Avila Belbute-Peres, Kevin A Smith, Kelsey Allen, Joshua B Tenenbaum, and J Zico Kolter. End-to-end differentiable physics for learning and control. In *Neural Information Processing Systems*, 2018.

Jonas Degrave, Michiel Hermans, Joni Dambre, et al. A differentiable physics engine for deep learning in robotics. *Frontiers in Neurorobotics*, 13, 2019.

Yilun Du, Yinan Zhang, Hong-Xing Yu, Joshua B Tenenbaum, and Jiajun Wu. Neural radiance flow for 4d view synthesis and video processing. *arXiv e-prints*, pages arXiv–2012, 2020.

Aysegul Dundar, Kevin J Shih, Animesh Garg, Robert Pottorf, Andrew Tao, and Bryan Catanzaro. Unsupervised disentanglement of pose, appearance and background from images and videos. *arXiv preprint arXiv:2001.09518*, 2020.

Frederik Ebert, Chelsea Finn, Alex X Lee, and Sergey Levine. Self-supervised visual planning with temporal skip connections. *arXiv preprint arXiv:1710.05268*, 2017.

Frederik Ebert, Chelsea Finn, Sudeep Dasari, Annie Xie, Alex Lee, and Sergey Levine. Visual foresight: Model-based deep reinforcement learning for vision-based robotic control. *arXiv preprint arXiv:1812.00568*, 2018.

Sebastien Ehrhardt, Aron Monszpart, Niloy Mitra, and Andrea Vedaldi. Taking visual motion prediction to new heightfields. *arXiv:1712.09448*, 2017.

Adam J Engler, Shamik Sen, H Lee Sweeney, and Dennis E Discher. Matrix elasticity directs stem cell lineage specification. *Cell*, 126(4):677–689, 2006.

Doris Entner and Patrik O Hoyer. On causal discovery from time series data using fci. *Probabilistic graphical models*, pages 121–128, 2010.

SM Ali Eslami, Danilo Jimenez Rezende, Frederic Besse, Fabio Viola, Ari S Morcos, Marta Garnelo, Avraham Ruderman, Andrei A Rusu, Ivo Danihelka, Karol Gregor, et al. Neural scene representation and rendering. *Science*, 360(6394):1204–1210, 2018.

Haoqiang Fan, Hao Su, and Leonidas J Guibas. A point set generation network for 3d object reconstruction from a single image. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 605–613, 2017.

Gregory Farquhar, Tim Rocktäschel, Maximilian Igl, and Shimon Whiteson. Treeqn and atreec: Differentiable tree planning for deep reinforcement learning. In *ICLR*, 2018.

Thomas Feix, Tracy L Kivell, Emmanuelle Pouydebat, and Aaron M Dollar. Estimating thumb–index finger precision grip and manipulation potential in extant and fossil primates. *Journal of the Royal Society Interface*, 12(106):20150176, 2015a.

Thomas Feix, Javier Romero, Heinz-Bodo Schmiedmayer, Aaron M Dollar, and Danica Kragic. The grasp taxonomy of human grasp types. *IEEE Transactions on human-machine systems*, 46(1):66–77, 2015b.

Chelsea Finn and Sergey Levine. Deep visual foresight for planning robot motion. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2786–2793. IEEE, 2017.

Chelsea Finn, Ian Goodfellow, and Sergey Levine. Unsupervised learning for physical interaction through video prediction. In *Advances in neural information processing systems*, pages 64–72, 2016.

J Randall Flanagan and Christina A Bandomir. Coming to grips with weight perception: effects of grasp configuration on perceived heaviness. *Perception & Psychophysics*, 62(6): 1204–1219, 2000.

Peter Florence. Dense visual learning for robot manipulation. In *PhD Thesis*, 2019.

Peter Florence, Lucas Manuelli, and Russ Tedrake. Self-supervised correspondence in visuomotor policy learning. *IEEE Robotics and Automation Letters*, 2019.

Peter R Florence, Lucas Manuelli, and Russ Tedrake. Dense object nets: Learning dense visual object descriptors by and for robotic manipulation. *Conference on Robot Learning (CoRL)*, 2018.

Katerina Fragkiadaki, Pulkit Agrawal, Sergey Levine, and Jitendra Malik. Learning visual predictive models of physics for playing billiards. In *ICLR*, 2016.

Gene F Franklin, J David Powell, Abbas Emami-Naeini, and J David Powell. *Feedback control of dynamic systems*, volume 4. Prentice hall Upper Saddle River, NJ, 2002.

Wei Gao and Russ Tedrake. kpam 2.0: Feedback control for category-level robotic manipulation. *arXiv preprint arXiv:2102.06279*, 2021.

Yang Gao, Lisa Anne Hendricks, Katherine J Kuchenbecker, and Trevor Darrell. Deep learning for tactile understanding from visual and haptic data. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 536–543. IEEE, 2016.

AmirEmad Ghassami, Saber Salehkaleybar, Negar Kiyavash, and Elias Bareinboim. Budgeted experiment design for causal structure learning. In *International Conference on Machine Learning*, pages 1724–1733, 2018.

Clark Glymour, Kun Zhang, and Peter Spirtes. Review of causal discovery methods based on graphical models. *Frontiers in Genetics*, 10, 2019.

Mingming Gong, Kun Zhang, Bernhard Schölkopf, Clark Glymour, and Dacheng Tao. Causal discovery from temporally aggregated time series. In *Uncertainty in artificial intelligence: proceedings of the... conference. Conference on Uncertainty in Artificial Intelligence*, volume 2017. NIH Public Access, 2017.

Olivier Goudet, Diviyan Kalainathan, Philippe Caillou, Isabelle Guyon, David Lopez-Paz, and Michèle Sebag. Causal generative neural networks. *arXiv preprint arXiv:1711.08936*, 2017.

Anirudh Goyal, Alex Lamb, Jordan Hoffmann, Shagun Sodhani, Sergey Levine, Yoshua Bengio, and Bernhard Schölkopf. Recurrent independent mechanisms. *arXiv preprint arXiv:1909.10893*, 2019.

Shixiang Gu, Timothy Lillicrap, Ilya Sutskever, and Sergey Levine. Continuous deep q-learning with model-based acceleration. In *ICML*, 2016.

David Ha and Jürgen Schmidhuber. World models. *arXiv preprint arXiv:1803.10122*, 2018.

Danijar Hafner, Timothy Lillicrap, Jimmy Ba, and Mohammad Norouzi. Dream to control: Learning behaviors by latent imagination. *arXiv preprint arXiv:1912.01603*, 2019a.

Danijar Hafner, Timothy Lillicrap, Ian Fischer, Ruben Villegas, David Ha, Honglak Lee, and James Davidson. Learning latent dynamics for planning from pixels. In *International Conference on Machine Learning*, pages 2555–2565. PMLR, 2019b.

Jessica B Hamrick, Peter W Battaglia, Thomas L Griffiths, and Joshua B Tenenbaum. Inferring mass in complex scenes by mental simulation. *Cognition*, 157:61–76, 2016.

Jessica B Hamrick, Andrew J Ballard, Razvan Pascanu, Oriol Vinyals, Nicolas Heess, and Peter W Battaglia. Metacontrol for adaptive imagination-based optimization. In *ICLR*, 2017.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

Bertrand Higy, Carlo Ciliberto, Lorenzo Rosasco, and Lorenzo Natale. Combining sensory modalities and exploratory procedures to improve haptic object recognition in robotics. In *2016 IEEE-RAS 16th International Conference on Humanoid Robots (Humanoids)*, pages 117–124. IEEE, 2016.

Francois R Hogan, Maria Bauza, and Alberto Rodriguez. A data-efficient approach to precise and controlled pushing. *arXiv preprint arXiv:1807.09904*, 2018.

François Robert Hogan and Alberto Rodriguez. Feedback control of the pusher-slider system: A story of hybrid and underactuated contact dynamics. *arXiv preprint arXiv:1611.08268*, 2016.

Jun-Ting Hsieh, Bingbin Liu, De-An Huang, Li F Fei-Fei, and Juan Carlos Niebles. Learning to decompose and disentangle representations for video prediction. In *Advances in Neural Information Processing Systems*, pages 517–526, 2018.

Yuanming Hu, Luke Anderson, Tzu-Mao Li, Qi Sun, Nathan Carr, Jonathan Ragan-Kelley, and Fredo Durand. Difftaichi: Differentiable programming for physical simulation. In *International Conference on Learning Representations (ICLR)*, 2019a.

Yuanming Hu, Jiancheng Liu, Andrew Spielberg, Joshua B Tenenbaum, William T Freeman, Jiajun Wu, Daniela Rus, and Wojciech Matusik. Chainqueen: A real-time differentiable physical simulator for soft robotics. In *ICRA*, 2019b.

Antti Hyttinen, Frederick Eberhardt, and Patrik O Hoyer. Experiment selection for causal discovery. *The Journal of Machine Learning Research*, 14(1):3041–3071, 2013.

Tomas Jakab, Ankush Gupta, Hakan Bilen, and Andrea Vedaldi. Unsupervised learning of object landmarks through conditional image generation. In *Advances in Neural Information Processing Systems*, pages 4016–4027, 2018.

Eric Jang, Shixiang Gu, and Ben Poole. Categorical reparameterization with gumbel-softmax. *arXiv preprint arXiv:1611.01144*, 2016.

Michael Janner, Sergey Levine, William T. Freeman, Joshua B. Tenenbaum, Chelsea Finn, and Jiajun Wu. Reasoning about physical interactions with object-centric models. In *International Conference on Learning Representations*, 2019.

Roland S Johansson and J Randall Flanagan. Coding and use of tactile signals from the fingertips in object manipulation tasks. *Nature Reviews Neuroscience*, 10(5):345–359, 2009.

Eurika Kaiser, J Nathan Kutz, and Steven L Brunton. Data-driven discovery of koopman eigenfunctions for control. *arXiv:1707.01146*, 2017.

Diviyan Kalainathan, Olivier Goudet, Isabelle Guyon, David Lopez-Paz, and Michèle Sebag. Sam: Structural agnostic model, causal discovery and penalized adversarial learning. *arXiv preprint arXiv:1803.04929*, 2018.

Zhanat Kappassov, Juan-Antonio Corrales, and Véronique Perdereau. Tactile sensing in dexterous robot hands. *Robotics and Autonomous Systems*, 74:195–220, 2015.

Nan Rosemary Ke, Olexa Bilaniuk, Anirudh Goyal, Stefan Bauer, Hugo Larochelle, Chris Pal, and Yoshua Bengio. Learning neural causal models from unknown interventions. *arXiv preprint arXiv:1910.01075*, 2019.

Dahl-Young Khang, Hanqing Jiang, Young Huang, and John A Rogers. A stretchable form of single-crystal silicon for high-performance electronics on rubber substrates. *Science*, 311 (5758):208–212, 2006.

Dae-Hyeong Kim, Jong-Hyun Ahn, Won Mook Choi, Hoon-Sik Kim, Tae-Ho Kim, Jizhou Song, Yonggang Y Huang, Zhuangjian Liu, Chun Lu, and John A Rogers. Stretchable and foldable silicon integrated circuits. *Science*, 320(5875):507–511, 2008.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *ICLR*, 2015.

Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

Thomas N Kipf, Ethan Fetaya, Kuan-Chieh Wang, Max Welling, and Richard S Zemel. Neural relational inference for interacting systems. *arXiv:1802.04687*, 2018.

Roberta L Klatzky, Susan J Lederman, and Victoria A Metzger. Identifying objects by touch: An "expert system". *Perception & psychophysics*, 37(4):299–302, 1985.

Junghyuk Ko, Sukhwinder Bhullar, Yonghyun Cho, Patrick C Lee, and Martin Byung-Guk Jun. Design and fabrication of auxetic stretchable force sensor for hand rehabilitation. *Smart Materials and Structures*, 24(7):075027, 2015.

Murat Kocaoglu, Karthikeyan Shanmugam, and Elias Bareinboim. Experimental design for learning causal graphs with latent variables. In *Advances in Neural Information Processing Systems*, pages 7018–7028, 2017.

Bernard O Koopman. Hamiltonian systems and transformation in hilbert space. *Proceedings of the national academy of sciences of the united states of america*, 17(5):315, 1931.

BO Koopman and J v Neumann. Dynamical systems of continuous spectra. *PNAS*, 18(3): 255, 1932.

Milan Korda and Igor Mezić. Linear predictors for nonlinear dynamical systems: Koopman operator meets model predictive control. *Automatica*, 93:149–160, 2018.

Martin Krzywinski, Jacqueline Schein, Inanc Birol, Joseph Connors, Randy Gascoyne, Doug Horsman, Steven J Jones, and Marco A Marra. Circos: an information aesthetic for comparative genomics. *Genome research*, 19(9):1639–1645, 2009.

Tejas D Kulkarni, Ankush Gupta, Catalin Ionescu, Sebastian Borgeaud, Malcolm Reynolds, Andrew Zisserman, and Volodymyr Mnih. Unsupervised learning of object keypoints for perception and control. In *Advances in Neural Information Processing Systems*, pages 10723–10733, 2019.

Manoj Kumar, Mohammad Babaeizadeh, Dumitru Erhan, Chelsea Finn, Sergey Levine, Laurent Dinh, and Durk Kingma. Videoflow: A conditional flow-based model for stochastic video generation. *arXiv preprint arXiv:1903.01434*, 2019.

Brenden M Lake, Tomer D Ullman, Joshua B Tenenbaum, and Samuel J Gershman. Building machines that learn and think like people. *Behavioral and brain sciences*, 40, 2017.

Roberto Lazzarini, R Magni, and Paolo Dario. A tactile array sensor layered in an artificial skin. In *Proceedings 1995 IEEE/RSJ International Conference on Intelligent Robots and Systems. Human Robot Interaction and Cooperative Robots*, volume 3, pages 114–119. IEEE, 1995.

Susan J Lederman and Roberta L Klatzky. Hand movements: A window into haptic object recognition. *Cognitive psychology*, 19(3):342–368, 1987.

Susan J Lederman and Roberta L Klatzky. Haptic perception: A tutorial. *Attention, Perception, & Psychophysics*, 71(7):1439–1459, 2009.

Tao Lei and Yu Zhang. Training rnns as fast as cnns. *arXiv preprint arXiv:1709.02755*, 2017.

Ian Lenz, Ross A Knepper, and Ashutosh Saxena. Deepmpc: Learning deep latent features for model predictive control. In *RSS*, 2015.

Sergey Levine, Chelsea Finn, Trevor Darrell, and Pieter Abbeel. End-to-end training of deep visuomotor policies. *The Journal of Machine Learning Research*, 17(1):1334–1373, 2016.

Rui Li, Robert Platt, Wenzhen Yuan, Andreas ten Pas, Nathan Roscup, Mandayam A Srinivasan, and Edward Adelson. Localization and manipulation of small parts using gelsight tactile sensing. In *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3988–3993. IEEE, 2014.

Tianye Li, Mira Slavcheva, Michael Zollhoefer, Simon Green, Christoph Lassner, Changil Kim, Tanner Schmidt, Steven Lovegrove, Michael Goesele, and Zhaoyang Lv. Neural 3d video synthesis. *arXiv preprint arXiv:2103.02597*, 2021.

Yunzhu Li, Jiajun Wu, Russ Tedrake, Joshua B Tenenbaum, and Antonio Torralba. Learning particle dynamics for manipulating rigid bodies, deformable objects, and fluids. *arXiv preprint arXiv:1810.01566*, 2018.

Yunzhu Li, Hao He, Jiajun Wu, Dina Katabi, and Antonio Torralba. Learning compositional koopman operators for model-based control. *arXiv preprint arXiv:1910.08264*, 2019a.

Yunzhu Li, Jiajun Wu, Jun-Yan Zhu, Joshua B Tenenbaum, Antonio Torralba, and Russ Tedrake. Propagation networks for model-based control under partial observation. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 1205–1211. IEEE, 2019b.

Yunzhu Li, Toru Lin, Kexin Yi, Daniel Bear, Daniel Yamins, Jiajun Wu, Joshua Tenenbaum, and Antonio Torralba. Visual grounding of learned physical models. In *International conference on machine learning*, pages 5927–5936. PMLR, 2020a.

Yunzhu Li, Antonio Torralba, Anima Anandkumar, Dieter Fox, and Animesh Garg. Causal discovery in physical systems from videos. *Advances in Neural Information Processing Systems*, 33:9180–9192, 2020b.

Yunzhu Li, Shuang Li, Vincent Sitzmann, Pulkit Agrawal, and Antonio Torralba. 3d neural scene representations for visuomotor control. In *Conference on Robot Learning*, pages 112–123. PMLR, 2022.

Zhengqi Li, Simon Niklaus, Noah Snavely, and Oliver Wang. Neural scene flow fields for space-time view synthesis of dynamic scenes. *arXiv preprint arXiv:2011.13084*, 2020c.

Junbang Liang, Ming Lin, and Vladlen Koltun. Differentiable cloth simulation for inverse problems. In *NIPS*, 2019.

Ce Liu et al. *Beyond pixels: exploring new representations and applications for motion analysis*. PhD thesis, Massachusetts Institute of Technology, 2009.

Lennart Ljung. System identification. *Wiley Encyclopedia of Electrical and Electronics Engineering*, 2001.

Stephen Lombardi, Tomas Simon, Jason Saragih, Gabriel Schwartz, Andreas Lehrmann, and Yaser Sheikh. Neural volumes: Learning dynamic renderable volumes from images. *arXiv preprint arXiv:1906.07751*, 2019.

Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3431–3440, 2015.

Yiyue Luo, Yunzhu Li, Michael Foshey, Wan Shou, Pratyusha Sharma, Tomás Palacios, Antonio Torralba, and Wojciech Matusik. Intelligent carpet: Inferring 3d human pose from tactile signals. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11255–11265, 2021a.

Yiyue Luo, Yunzhu Li, Pratyusha Sharma, Wan Shou, Kui Wu, Michael Foshey, Beichen Li, Tomás Palacios, Antonio Torralba, and Wojciech Matusik. Learning human–environment interactions using conformal tactile textiles. *Nature Electronics*, 4(3):193–201, 2021b.

Bethany Lusch, J Nathan Kutz, and Steven L Brunton. Deep learning for universal linear embeddings of nonlinear dynamics. *Nature Communications*, 9(1):4950, 2018.

Miles Macklin and Matthias Müller. Position based fluids. *ACM TOG*, 32(4):104, 2013.

Miles Macklin, Matthias Müller, Nuttapong Chentanez, and Tae-Yong Kim. Unified particle physics for real-time applications. *ACM TOG*, 33(4):153, 2014.

Chris J Maddison, Andriy Mnih, and Yee Whye Teh. The concrete distribution: A continuous relaxation of discrete random variables. *arXiv preprint arXiv:1611.00712*, 2016.

Jeffrey Mahler, Matthew Matl, Vishal Satish, Michael Danielczuk, Bill DeRose, Stephen McKinley, and Ken Goldberg. Learning ambidextrous robot grasping policies. *Science Robotics*, 4(26):eaau4984, 2019.

Giorgos Mamakoukas, Maria Castano, Xiaobo Tan, and Todd Murphey. Local koopman operators for data-driven control of robotic systems. In *RSS*, 2019.

Lucas Manuelli, Wei Gao, Peter Florence, and Russ Tedrake. kpam: Keypoint affordances for category-level robotic manipulation. *arXiv preprint arXiv:1903.06684*, 2019.

Lucas Manuelli, Yunzhu Li, Pete Florence, and Russ Tedrake. Keypoints into the future: Self-supervised correspondence in model-based reinforcement learning. In *Conference on Robot Learning (CoRL)*, pages 693–710. PMLR, 2021.

Mary W Marzke. Precision grips, hand morphology, and tools. *American Journal of Physical Anthropology: The Official Publication of the American Association of Physical Anthropologists*, 102(1):91–110, 1997.

Alexandre Mauroy and Jorge Goncalves. Linear identification of nonlinear systems: A lifting technique based on the koopman operator. In *CDC*, 2016.

Alexandre Mauroy and Jorge Goncalves. Koopman-based lifting techniques for nonlinear systems identification. *IEEE Transactions on Automatic Control*, 2019.

David Mayne. Nonlinear model predictive control: Challenges and opportunities. In *Nonlinear Model Predictive Control*, pages 23–44. Springer, 2000.

Martin Meier, Guillaume Walck, Robert Haschke, and Helge J Ritter. Distinguishing sliding from slipping during object pushing. In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5579–5584. IEEE, 2016.

Daniel Mellinger, Nathan Michael, and Vijay Kumar. Trajectory generation and control for precise aggressive maneuvers with quadrotors. *The International Journal of Robotics Research*, 31(5):664–674, 2012.

Ben Mildenhall, Pratul P Srinivasan, Matthew Tancik, Jonathan T Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. In *European conference on computer vision*, pages 405–421. Springer, 2020.

Matthias Minderer, Chen Sun, Ruben Villegas, Forrester Cole, Kevin P Murphy, and Honglak Lee. Unsupervised learning of object structure and dynamics from videos. In *NIPS*, 2019.

Thomas B Moeslund, Adrian Hilton, and Volker Krüger. A survey of advances in vision-based human motion capture and analysis. *Computer vision and image understanding*, 104(2-3): 90–126, 2006.

Joseph Moore, Rick Cory, and Russ Tedrake. Robust post-stall perching with a simple fixed-wing glider using lqr-trees. *Bioinspiration & biomimetics*, 9(2):025013, 2014.

Douglas Morrison, Peter Corke, and Jürgen Leitner. Closing the loop for robotic grasping: A real-time, generative grasp synthesis approach. *arXiv preprint arXiv:1804.05172*, 2018.

Jeremy Morton, Freddie D Witherden, Antony Jameson, and Mykel J Kochenderfer. Deep dynamical modeling and control of unsteady fluid flows. In *NIPS*, 2018.

Jeremy Morton, Freddie D Witherden, and Mykel J Kochenderfer. Deep variational koopman models: Inferring koopman observations for uncertainty-aware dynamics modeling and control. In *IJCAI*, 2019.

Damian Mrowca, Chengxu Zhuang, Elias Wang, Nick Haber, Li Fei-Fei, Joshua B Tenenbaum, and Daniel LK Yamins. Flexible neural representation for physics prediction. In *NIPS*, 2018.

Anusha Nagabandi, Gregory Kahn, Ronald S Fearing, and Sergey Levine. Neural network dynamics for model-based deep reinforcement learning with model-free fine-tuning. In *ICRA*, 2018.

Anusha Nagabandi, Ignasi Clavera, Simin Liu, Ronald S Fearing, Pieter Abbeel, Sergey Levine, and Chelsea Finn. Learning to adapt in dynamic, real-world environments through meta-reinforcement learning. In *ICLR*, 2019a.

Anusha Nagabandi, Chelsea Finn, and Sergey Levine. Deep online learning via meta-learning: Continual adaptation for model-based rl. In *ICLR*, 2019b.

Anusha Nagabandi, Kurt Konolige, Sergey Levine, and Vikash Kumar. Deep dynamics models for learning dexterous manipulation. In *Conference on Robot Learning (CoRL)*, pages 1101–1112. PMLR, 2020.

John R Napier. The prehensile movements of the human hand. *The Journal of bone and joint surgery. British volume*, 38(4):902–913, 1956.

Vidya Narayanan, Kui Wu, Cem Yuksel, and James McCann. Visual knitting machine programming. *ACM Transactions on Graphics (TOG)*, 38(4):1–13, 2019.

Fiona N Newell, Marc O Ernst, Bosco S Tjan, and Heinrich H Bülthoff. Viewpoint dependence in visual and haptic object recognition. *Psychological science*, 12(1):37–42, 2001.

Thu Nguyen-Phuoc, Chuan Li, Stephen Balaban, and Yong-Liang Yang. Rendernet: A deep convolutional network for differentiable rendering from 3d shapes. *arXiv preprint arXiv:1806.06575*, 2018.

Michael Niemeyer, Lars Mescheder, Michael Oechsle, and Andreas Geiger. Occupancy flow: 4d reconstruction by learning particle dynamics. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 5379–5389, 2019.

Michael Niemeyer, Lars Mescheder, Michael Oechsle, and Andreas Geiger. Differentiable volumetric rendering: Learning implicit 3d representations without 3d supervision. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3504–3515, 2020.

Wesley A Niewoehner, Aaron Bergstrom, Derrick Eichele, Melissa Zuroff, and Jeffrey T Clark. Manual dexterity in neanderthals. *Nature*, 422(6930):395–395, 2003.

Junhyuk Oh, Satinder Singh, and Honglak Lee. Value prediction network. In *NIPS*, 2017.

Allison M Okamura, Niels Smaby, and Mark R Cutkosky. An overview of dexterous manipulation. In *Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No. 00CH37065)*, volume 1, pages 255–262. IEEE, 2000.

Edwin Olson. Apriltag: A robust and flexible visual fiducial system. In *2011 IEEE International Conference on Robotics and Automation*, pages 3400–3407. IEEE, 2011.

Luke E Osborn, Andrei Dragomir, Joseph L Betthauser, Christopher L Hunt, Harrison H Nguyen, Rahul R Kaliki, and Nitish V Thakor. Prosthesis with neuromorphic multilayered e-dermis perceives touch and pain. *Science robotics*, 3(19):eaat3818, 2018.

Julian Ost, Fahim Mannan, Nils Thuerey, Julian Knodt, and Felix Heide. Neural scene graphs for dynamic scenes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2856–2865, 2021.

Jeong Joon Park, Peter Florence, Julian Straub, Richard Newcombe, and Steven Lovegrove. Deepsdf: Learning continuous signed distance functions for shape representation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 165–174, 2019.

Jonghwa Park, Marie Kim, Youngoh Lee, Heon Sang Lee, and Hyunhyub Ko. Fingertip skin–inspired microstructured ferroelectric skins discriminate static/dynamic pressure and temperature stimuli. *Science advances*, 1(9):e1500661, 2015.

Keunhong Park, Utkarsh Sinha, Jonathan T Barron, Sofien Bouaziz, Dan B Goldman, Steven M Seitz, and Ricardo-Martin Brualla. Deformable neural radiance fields. *arXiv preprint arXiv:2011.12948*, 2020.

Razvan Pascanu, Yujia Li, Oriol Vinyals, Nicolas Heess, Lars Buesing, Sebastien Racanière, David Reichert, Théophane Weber, Daan Wierstra, and Peter Battaglia. Learning model-based planning from scratch. *arXiv:1707.06170*, 2017.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems*, pages 8024–8035, 2019.

Judea Pearl. *Causality*. Cambridge university press, 2009.

Jonas Peters, Peter Bühlmann, and Nicolai Meinshausen. Causal inference by using invariant prediction: identification and confidence intervals. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 78(5):947–1012, 2016.

Jonas Peters, Dominik Janzing, and Bernhard Schölkopf. *Elements of causal inference: foundations and learning algorithms*. MIT press, 2017.

Ivan Poupyrev, Nan-Wei Gong, Shiho Fukuhara, Mustafa Emre Karagozler, Carsten Schwesig, and Karen E Robinson. Project jacquard: interactive digital textiles at scale. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*, pages 4216–4227, 2016.

Joshua L Proctor, Steven L Brunton, and J Nathan Kutz. Generalizing koopman theory to allow for inputs and control. *SIAM Journal on Applied Dynamical Systems*, 17(1):909–930, 2018.

Albert Pumarola, Enric Corona, Gerard Pons-Moll, and Francesc Moreno-Noguer. D-nerf: Neural radiance fields for dynamic scenes. *arXiv preprint arXiv:2011.13961*, 2020.

Sébastien Racanière, Théophane Weber, David Reichert, Lars Buesing, Arthur Guez, Danilo Jimenez Rezende, Adrià Puigdomènech Badia, Oriol Vinyals, Nicolas Heess, Yujia Li, Razvan Pascanu, Peter Battaglia, David Silver, and Daan Wierstra. Imagination-augmented agents for deep reinforcement learning. In *NIPS*, 2017.

Michael Rein, Valentine Dominique Favrod, Chong Hou, Tural Khudiyev, Alexander Stolyarov, Jason Cox, Chia-Chun Chung, Chhea Chhav, Marty Ellis, John Joannopoulos, et al. Diode fibres for fabric-based optical communications. *Nature*, 560(7717):214–218, 2018.

Danilo Jimenez Rezende, SM Eslami, Shakir Mohamed, Peter Battaglia, Max Jaderberg, and Nicolas Heess. Unsupervised learning of 3d structure from images. *arXiv preprint arXiv:1607.00662*, 2016.

John A Rogers, Takao Someya, and Yonggang Huang. Materials and mechanics for stretchable electronics. *science*, 327(5973):1603–1607, 2010.

Joseph M Romano, Kaijen Hsiao, Günter Niemeyer, Sachin Chitta, and Katherine J Kuchenbecker. Human-inspired robotic grasp control with tactile sensing. *IEEE Transactions on Robotics*, 27(6):1067–1079, 2011.

Dominik Rothenhäusler, Christina Heinze, Jonas Peters, and Nicolai Meinshausen. Backshift: Learning causal cyclic graphs from unknown shift interventions. In *Advances in Neural Information Processing Systems*, pages 1513–1521, 2015.

Clarence W Rowley, Igor Mezić, Shervin Bagheri, Philipp Schlatter, and Dan S Henningson. Spectral analysis of nonlinear flows. *Journal of Fluid Mechanics*, 641:115–127, 2009.

Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *International journal of computer vision*, 115(3):211–252, 2015.

Hannes P Saal, Benoit P Delhaye, Brandon C Rayhaun, and Sliman J Bensmaia. Simulating tactile signals from the whole hand with millisecond precision. *Proceedings of the National Academy of Sciences*, 114(28):E5693–E5702, 2017.

Shunsuke Saito, Zeng Huang, Ryota Natsume, Shigeo Morishima, Angjoo Kanazawa, and Hao Li. Pifu: Pixel-aligned implicit function for high-resolution clothed human digitization. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 2304–2314, 2019.

Alvaro Sanchez-Gonzalez, Nicolas Heess, Jost Tobias Springenberg, Josh Merel, Martin Riedmiller, Raia Hadsell, and Peter Battaglia. Graph networks as learnable physics engines for inference and control. In *ICML*, 2018.

Alvaro Sanchez-Gonzalez, Jonathan Godwin, Tobias Pfaff, Rex Ying, Jure Leskovec, and Peter Battaglia. Learning to simulate complex physics with graph networks. In *International Conference on Machine Learning (ICML)*, pages 8459–8468. PMLR, 2020.

Adam Santoro, David Raposo, David G Barrett, Mateusz Malinowski, Razvan Pascanu, Peter Battaglia, and Timothy Lillicrap. A simple neural network module for relational reasoning. In *Advances in neural information processing systems*, pages 4967–4976, 2017.

Giovanni Saponaro, Lorenzo Jamone, Alexandre Bernardino, and Giampiero Salvi. Beyond the self: Using grounded affordances to interpret and describe others' actions. *IEEE Transactions on Cognitive and Developmental Systems*, 12(2):209–221, 2019.

Connor Schenck and Dieter Fox. Reasoning about liquids via closed-loop simulation. *arXiv preprint arXiv:1703.01656*, 2017.

Connor Schenck and Dieter Fox. Perceiving and reasoning about liquids using fully convolutional networks. *The International Journal of Robotics Research*, 37(4-5):452–471, 2018a.

Connor Schenck and Dieter Fox. Spnets: Differentiable fluid dynamics for deep neural networks. In *CoRL*, 2018b.

Peter J Schmid. Dynamic mode decomposition of numerical and experimental data. *Journal of Fluid Mechanics*, 656:5–28, 2010.

Tanner Schmidt, Richard Newcombe, and Dieter Fox. Self-supervised visual descriptor learning for dense correspondence. *IEEE Robotics and Automation Letters*, 2(2):420–427, 2017.

Julian Schrittwieser, Ioannis Antonoglou, Thomas Hubert, Karen Simonyan, Laurent Sifre, Simon Schmitt, Arthur Guez, Edward Lockhart, Demis Hassabis, Thore Graepel, et al. Mastering atari, go, chess and shogi by planning with a learned model. *Nature*, 588(7839): 604–609, 2020.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv:1707.06347*, 2017.

Jesse Scott, Christopher Funk, Bharadwaj Ravichandran, John H Challis, Robert T Collins, and Yanxi Liu. From kinematics to dynamics: Estimating center of pressure and base of support from video frames of human motion. *arXiv preprint arXiv:2001.00657*, 2020.

Shima Seiki. Sds-one apex3, 2011.

Pierre Sermanet, Corey Lynch, Yevgen Chebotar, Jasmine Hsu, Eric Jang, Stefan Schaal, Sergey Levine, and Google Brain. Time-contrastive networks: Self-supervised learning from video. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1134–1141. IEEE, 2018.

Karthikeyan Shanmugam, Murat Kocaoglu, Alexandros G Dimakis, and Sriram Vishwanath. Learning causal graphs with small interventions. In *Advances in Neural Information Processing Systems*, pages 3195–3203, 2015.

Shohei Shimizu. Lingam: Non-gaussian methods for estimating causal structures. *Behaviormetrika*, 41(1):65–98, 2014.

David Silver, Hado van Hasselt, Matteo Hessel, Tom Schaul, Arthur Guez, Tim Harley, Gabriel Dulac-Arnold, David Reichert, Neil Rabinowitz, Andre Barreto, and Thomas Degris. The predictron: End-to-end learning and planning. In *ICML*, 2017.

Tomas Simon, Hanbyul Joo, Iain Matthews, and Yaser Sheikh. Hand keypoint detection in single images using multiview bootstrapping. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 1145–1153, 2017.

Vincent Sitzmann, Justus Thies, Felix Heide, Matthias Nießner, Gordon Wetzstein, and Michael Zollhöfer. Deepvoxels: Learning persistent 3d feature embeddings. In *Proc. CVPR*, 2019a.

Vincent Sitzmann, Michael Zollhöfer, and Gordon Wetzstein. Scene representation networks: Continuous 3d-structure-aware neural scene representations. *arXiv preprint arXiv:1906.01618*, 2019b.

Vincent Sitzmann, Eric R Chan, Richard Tucker, Noah Snavely, and Gordon Wetzstein. Metasdf: Meta-learning signed distance functions. *arXiv preprint arXiv:2006.09662*, 2020.

Takao Someya, Zhenan Bao, and George G Malliaras. The rise of plastic bioelectronics. *Nature*, 540(7633):379–385, 2016.

Peter Spirtes, Clark N Glymour, Richard Scheines, and David Heckerman. *Causation, prediction, and search*. MIT press, 2000.

Aravind Srinivas, Allan Jabri, Pieter Abbeel, Sergey Levine, and Chelsea Finn. Universal planning networks. In *ICML*, 2018.

HJ Suh and Russ Tedrake. The surprising effectiveness of linear models for visual foresight in object pile manipulation. *arXiv preprint arXiv:2002.09093*, 2020.

Subramanian Sundaram, Petr Kellnhofer, Yunzhu Li, Jun-Yan Zhu, Antonio Torralba, and Wojciech Matusik. Learning the signatures of the human grasp using a scalable tactile glove. *Nature*, 569(7758):698–702, 2019.

Supasorn Suwajanakorn, Noah Snavely, Jonathan J Tompson, and Mohammad Norouzi. Discovery of latent 3d keypoints via end-to-end geometric reasoning. In *Advances in Neural Information Processing Systems*, pages 2059–2070, 2018.

Naoya Takeishi, Yoshinobu Kawahara, and Takehisa Yairi. Learning koopman invariant subspaces for dynamic mode decomposition. In *NIPS*, 2017.

Maxim Tatarchenko, Alexey Dosovitskiy, and Thomas Brox. Single-view to multi-view: Reconstructing unseen views with a convolutional network. *CoRR abs/1511.06702*, 1(2):2, 2015.

Russ Tedrake. Underactuated robotics: Learning, planning, and control for efficient and agile machines course notes for mit 6.832, 2009.

Russ Tedrake and the Drake Development Team. Drake: Model-based design and verification for robotics, 2019. URL `https://drake.mit.edu`.

Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control. In *IROS*, pages 5026–5033. IEEE, 2012.

Jonathan Tompson, Ross Goroshin, Arjun Jain, Yann LeCun, and Christoph Bregler. Efficient object localization using convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 648–656, 2015.

Antonio Torralba, Rob Fergus, and William T Freeman. 80 million tiny images: A large data set for nonparametric object and scene recognition. *IEEE transactions on pattern analysis and machine intelligence*, 30(11):1958–1970, 2008.

Marc Toussaint, K Allen, K Smith, and J Tenenbaum. Differentiable physics and stable modes for tool-use and manipulation planning. In *RSS*, 2018.

Edgar Tretschk, Ayush Tewari, Vladislav Golyanik, Michael Zollhöfer, Christoph Lassner, and Christian Theobalt. Non-rigid neural radiance fields: Reconstruction and novel view synthesis of a deforming scene from monocular video. *arXiv preprint arXiv:2012.12247*, 2020.

Alex Trevithick and Bo Yang. Grf: Learning a general radiance field for 3d scene representation and rendering. *arXiv preprint arXiv:2010.04595*, 2020.

Ryan L Truby and Jennifer A Lewis. Printing soft matter in three dimensions. *Nature*, 540 (7633):371–378, 2016.

Jonathan H Tu, Clarence W Rowley, Dirk M Luchtenburg, Steven L Brunton, and J Nathan Kutz. On dynamic mode decomposition: Theory and applications. *Journal of Computational Dynamics*, 1(2):391–421, 2014.

Hsiao-Yu Fish Tung, Ricson Cheng, and Katerina Fragkiadaki. Learning spatial common sense with geometry-aware recurrent networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2595–2603, 2019.

Hsiao-Yu Fish Tung, Zhou Xian, Mihir Prabhudesai, Shamit Lal, and Katerina Fragkiadaki. 3d-oes: Viewpoint-invariant object-factorized environment simulators. *arXiv preprint arXiv:2011.06464*, 2020.

Dmitry Ulyanov, Andrea Vedaldi, and Victor Lempitsky. Deep image prior. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 9446–9454, 2018.

Benjamin Ummenhofer, Lukas Prantl, Nils Thuerey, and Vladlen Koltun. Lagrangian fluid simulation with continuous convolutions. In *International Conference on Learning Representations*, 2020.

Laurens Van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(11), 2008.

Sjoerd van Steenkiste, Michael Chang, Klaus Greff, and Jürgen Schmidhuber. Relational neural expectation maximization: Unsupervised discovery of objects and their interactions. In *ICLR*, 2018.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017.

Eric Wan, Antonio Baptista, Magnus Carlsson, Richard Kiebutz, Yinglong Zhang, and Alexander Bogdanov. Model predictive neural control of a high-fidelity helicopter model. In *AIAA Guidance, Navigation, and Control Conference and Exhibit*, page 4164, 2001.

Chen Wang, Roberto Martín-Martín, Danfei Xu, Jun Lv, Cewu Lu, Li Fei-Fei, Silvio Savarese, and Yuke Zhu. 6-pack: Category-level 6d pose tracker with anchor-based keypoints. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 10059–10066. IEEE, 2020.

Sihong Wang, Jin Young Oh, Jie Xu, Helen Tran, and Zhenan Bao. Skin-inspired electronics: an emerging paradigm. *Accounts of chemical research*, 51(5):1033–1045, 2018.

Yuhao Wang, Liam Solus, Karren Yang, and Caroline Uhler. Permutation-based causal inference algorithms with interventions. In *Advances in Neural Information Processing Systems*, pages 5822–5831, 2017a.

Zhihua Wang, Zhaochu Yang, and Tao Dong. A review of wearable technologies for elderly care that can accurately track indoor position, recognize physical activities and monitor vital signs in real time. *Sensors*, 17(2):341, 2017b.

Manuel Watter, Jost Springenberg, Joschka Boedecker, and Martin Riedmiller. Embed to control: A locally linear latent dynamics model for control from raw images. In *Advances in neural information processing systems*, pages 2746–2754, 2015.

Nicholas Watters, Andrea Tacchetti, Theophane Weber, Razvan Pascanu, Peter Battaglia, and Daniel Zoran. Visual interaction networks. In *NIPS*, 2017.

Nicholas Wettels and Gerald E Loeb. Haptic feature extraction from a biomimetic tactile sensor: force, contact location and curvature. In *2011 IEEE International Conference on Robotics and Biomimetics*, pages 2471–2478. IEEE, 2011.

Grady Williams, Andrew Aldrich, and Evangelos Theodorou. Model predictive path integral control using covariance variable importance sampling. *arXiv preprint arXiv:1509.01149*, 2015a.

Matthew O Williams, Ioannis G Kevrekidis, and Clarence W Rowley. A data–driven approximation of the koopman operator: Extending dynamic mode decomposition. *Journal of Nonlinear Science*, 25(6):1307–1346, 2015b.

Matthew O Williams, Maziar S Hemati, Scott TM Dawson, Ioannis G Kevrekidis, and Clarence W Rowley. Extending data-driven koopman analysis to actuated systems. *IFAC-PapersOnLine*, 49(18):704–709, 2016.

David A Winter. Human balance and posture control during standing and walking. *Gait & posture*, 3(4):193–214, 1995.

Daniel E Worrall, Stephan J Garbin, Daniyar Turmukhambetov, and Gabriel J Brostow. Interpretable transformations with encoder-decoder networks. In *Proc. ICCV*, volume 4, 2017.

Jiajun Wu, Ilker Yildirim, Joseph J Lim, William T Freeman, and Joshua B Tenenbaum. Galileo: Perceiving physical object properties by integrating a physics engine with deep learning. In *NIPS*, 2015.

Jiajun Wu, Erika Lu, Pushmeet Kohli, Bill Freeman, and Josh Tenenbaum. Learning to see physics via visual de-animation. In *NIPS*, 2017.

Wenqi Xian, Jia-Bin Huang, Johannes Kopf, and Changil Kim. Space-time neural irradiance fields for free-viewpoint video. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9421–9431, 2021.

Zhenjia Xu, Jiajun Wu, Andy Zeng, Joshua B Tenenbaum, and Shuran Song. Densephysnet: Learning dense physical object representations via multi-step dynamic interactions. In *Robotics: Science and Systems (RSS)*, 2019.

Akihiko Yamaguchi and Christopher G Atkeson. Combining finger vision and optical tactile sensing: Reducing and handling errors while cutting vegetables. In *2016 IEEE-RAS 16th International Conference on Humanoid Robots (Humanoids)*, pages 1045–1051. IEEE, 2016.

Wei Yan, Alexis Page, Tung Nguyen-Dang, Yunpeng Qu, Federica Sordo, Lei Wei, and Fabien Sorin. Advanced multimaterial electronic and optoelectronic fibers and textiles. *Advanced materials*, 31(1):1802348, 2019.

Wilson Yan, Ashwin Vangipuram, Pieter Abbeel, and Lerrel Pinto. Learning predictive representations for deformable objects using contrastive estimation. In *Conference on Robot Learning (CoRL)*, pages 564–574. PMLR, 2021.

Guang-Zhong Yang, Jim Bellingham, Pierre E Dupont, Peer Fischer, Luciano Floridi, Robert Full, Neil Jacobstein, Vijay Kumar, Marcia McNutt, Robert Merrifield, et al. The grand challenges of science robotics. *Science robotics*, 3(14):eaar7650, 2018.

Jeffrey M Yau, Sung Soo Kim, Pramodsingh H Thakur, and Sliman J Bensmaia. Feeling form: the neural basis of haptic shape perception. *Journal of Neurophysiology*, 115(2): 631–642, 2016.

Yufei Ye, Maneesh Singh, Abhinav Gupta, and Shubham Tulsiani. Compositional video prediction. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 10353–10362, 2019.

Lin Yen-Chen, Maria Bauza, and Phillip Isola. Experience-embedded visual foresight. In *Conference on Robot Learning*, pages 1015–1024. PMLR, 2020.

Kexin Yi, Chuang Gan, Yunzhu Li, Pushmeet Kohli, Jiajun Wu, Antonio Torralba, and Joshua B Tenenbaum. Clevrer: Collision events for video representation and reasoning. *arXiv preprint arXiv:1910.01442*, 2019.

Hanna Yousef, Mehdi Boukallel, and Kaspar Althoefer. Tactile sensing for dexterous in-hand manipulation in robotics—a review. *Sensors and Actuators A: physical*, 167(2):171–187, 2011.

Alex Yu, Vickie Ye, Matthew Tancik, and Angjoo Kanazawa. pixelnerf: Neural radiance fields from one or few images. *arXiv preprint arXiv:2012.02190*, 2020.

Andy Zeng, Shuran Song, Stefan Welker, Johnny Lee, Alberto Rodriguez, and Thomas Funkhouser. Learning synergies between pushing and grasping with self-supervised deep reinforcement learning. *arXiv preprint arXiv:1803.09956*, 2018.

Andy Zeng, Shuran Song, Johnny Lee, Alberto Rodriguez, and Thomas Funkhouser. Tossingbot: Learning to throw arbitrary objects with residual physics. *arXiv preprint arXiv:1903.11239*, 2019.

Wei Zeng, Lin Shu, Qiao Li, Song Chen, Fei Wang, and Xiao-Ming Tao. Fiber-based wearable electronics: a review of materials, fabrication, devices, and applications. *Advanced materials*, 26(31):5310–5336, 2014.

K Zhang and A Hyvärinen. On the identifiability of the post-nonlinear causal model. In *25th Conference on Uncertainty in Artificial Intelligence (UAI 2009)*, pages 647–655. AUAI Press, 2009.

Kun Zhang, Mingming Gong, Joseph Ramsey, Kayhan Batmanghelich, Peter Spirtes, and Clark Glymour. Causal discovery in the presence of measurement error: Identifiability conditions. *arXiv preprint arXiv:1706.03768*, 2017.

Qiang Zhang, Yunzhu Li, Yiyue Luo, Wan Shou, Michael Foshey, Junchi Yan, Joshua B Tenenbaum, Wojciech Matusik, and Antonio Torralba. Dynamic modeling of hand-object interactions via tactile sensing. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2874–2881. IEEE, 2021.

Yuting Zhang, Yijie Guo, Yixin Jin, Yijun Luo, Zhiyuan He, and Honglak Lee. Unsupervised discovery of object landmarks as structural representations. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2694–2703, 2018.

Xun Zheng, Bryon Aragam, Pradeep K Ravikumar, and Eric P Xing. Dags with no tears: Continuous optimization for structure learning. In *Advances in Neural Information Processing Systems*, pages 9472–9483, 2018.

Jiaji Zhou, Yifan Hou, and Matthew T Mason. Pushing revisited: Differential flatness, trajectory planning, and stabilization. *The International Journal of Robotics Research*, 38 (12-13):1477–1489, 2019.

Jun-Yan Zhu, Zhoutong Zhang, Chengkai Zhang, Jiajun Wu, Antonio Torralba, Josh Tenenbaum, and Bill Freeman. Visual object networks: image generation with disentangled 3d representations. In *Proc. NIPS*, pages 118–129, 2018.