# Causal Robot Learning for Manipulation

*Thesis*

Tabitha Edith Lee
CMU-RI-TR-24-25
July 5, 2024


The Robotics Institute
School of Computer Science
Carnegie Mellon University
Pittsburgh, Pennsylvania

**Thesis Committee:**

| | |
|---|---|
| Prof. Oliver Kroemer, *Chair* | The Robotics Institute<br>Carnegie Mellon University |
| Prof. Shubham Tulsiani | The Robotics Institute<br>Carnegie Mellon University |
| Prof. Kun Zhang | Dept. of Philosophy &<br>Machine Learning Dept.<br>Carnegie Mellon University |
| Prof. Jonas Martin Peters | Dept. of Mathematics<br>ETH Zürich |

*Submitted in partial fulfillment of the requirements*
*for the degree of Doctor of Philosophy in Robotics.*

**Abstract**

Two decades into the third age of artificial intelligence, the rise of deep learning has yielded two seemingly disparate realities. In one, massive accomplishments have been achieved in deep reinforcement learning, protein folding, and large language models. Yet, in the other, the promises of deep learning to empower robots that operate robustly in real-world environments have yet remained unfulfilled. Vast diversity of objects, distribution shifts, long-tailed phenomena: outside of laboratories, real-world environments challenge modern statistical learning assumptions of the data.

Although such environments have generally been referred to as "unstructured," this terminology belies their nature. Real-world environments are not "unstructured," but arise *because of* structure: the underlying causal processes that generate the observed data. In this view, robots should not only reason and learn with respect to data, but also the data generating processes. Such processes can be formalized by the language of causality. Therefore, to learn and leverage the structure of these "open-world" environments, new causal-based robot learning algorithms are needed.

Towards this goal, this thesis explores a diversity of robot learning problems, from perception to control. First, we explore how perception models can be learned using sim-to-real transfer from synthetic data (DREAM, FormNet). In these works, relevant features were learned through domain randomization, leading to insights into how structure can be learned more directly using causality. To this end, we introduce structural sim-to-real transfer, where simulation can serve as a causal reasoning engine for the robot to select the relevant features for a control policy (CREST) or skill (SCALE). By construction, these policies are robust to irrelevant distribution shifts that would otherwise stymie correlation-based deep learning. Next, the rich interplay between control, dynamical systems, and causality is explored through the Learning By Doing (LBD) competition and LMeshNet, a methodology for constructing hybrid causal world models that integrate both latent and semantic information. Lastly, we examine curriculum learning. In ACL, we explore the commonalities and differences between human and agent curriculum learning. Then, we employ these lessons learned for CURATE: how agents can manipulate the sequencing of training data to efficiently learn a control policy to solve a desired, difficult task.

The advantages of adopting the principles of causal inference have been witnessed to date in fields such as biomedical sciences, economics, and genomics. In the machine learning community, it has recently been argued that such principles should be integrated to harness deep learning, towards causal learning of representations. Analogously, this thesis forwards that the robot learning community stands to gain by leveraging the principles of causality. In so doing, this new paradigm holds promise for robots to learn and leverage structure within the open world through ***causal robot learning for manipulation***.

*To the mountains we climb,*
*the people who uplift us,*
*and the wisdom we gain at the summit.*

# Acknowledgments

"You don't climb mountains without a team..." (Sen. Mark Udall)

While climbing mountains can be seen as an individual endeavor, for every successful mountain summit is a host of people who contributed along the way. This thesis would not have happened without the countless and innumerable contributions of my colleagues, collaborators, friends, and family. Therefore, I dedicate this thesis to you. Before I continue my acknowledgments, please allow me to first reflect upon this journey to contextualize the profound impact that others have made.

> "May your dreams be larger than mountains and may you have the courage to scale their summits." (Harley King)

Ten years ago, I decided to climb a mountain. This was not a literal mountain, but a metaphorical one: the pursuit of a PhD in embodied artificial intelligence. Robotics, long a curiosity, became a passion that propelled my dreams. And, in 2018, I was given an opportunity: after completing my MS at Carnegie Mellon University, I started my ascent in the PhD program. Yet, in order to climb one mountain, I had to summit two other mountains first.

> "Mountains know secrets we need to learn. That it might take time, it might be hard, but if you just hold on long enough, you will find strength to rise up." (Tyler Knott Gregson)

In 2020, a worldwide pandemic left an indelible mark upon us all: for some of us more than others. During the pandemic, lifelong impostor syndrome and feelings of misalignment roared to an inescapable din. I felt stranded upon the mountain, unable to make progress in my ascent while deeply vulnerable within a storm of doubt. Having no other choice than to confront my incongruence, I realized that to finish this climb, I needed to conquer another mountain first. Halfway through my PhD, I made my first summit: I fully embraced my gender identity and transitioned to exist as my most authentic self.

Feeling like I had been reborn, I embarked on my ascent towards the PhD once again, now with a tailwind aiding my climb. Yet, nearly one year later, my life profoundly changed once again. An accident led to the loss of my leg. Immediately, a second mountain loomed. During this time, I grappled with fear, self-doubt, and an overwhelming grief. But, through the support of family, friends, and faith, I recovered. Slowly but surely, my fear gave way to hope. Self-doubt gave way to belief. Finally, grief gave way to acceptance. I relearned to walk with a prosthesis and made my second summit. What had been lost, has now been regained many times over.

"The summit is what drives us, but the climb itself is what matters."
(Conrad Anker)

This thesis represents my third summit, conquering the mountain I had sought to climb ten years ago. My steps may be slower now: more careful, more thoughtful. But, they are also more intentional, more determined. Most important of all, they are more grateful. Because, if there is one constant throughout this incredible, life-changing journey, it has been the profound importance of people. Climbing these mountains — becoming the researcher and person I am today — simply would not have been possible without the immeasurable contributions of others.

I owe these summits to you.

~

Thus, I will now begin my attempt to acknowledge every person who has made this thesis a reality. For those who I have not mentioned in these acknowledgments, please forgive my oversight. Please know that you remain in my heart.

First, to my PhD thesis advisor, Prof. Oliver Kroemer: I am forever grateful for your steadfast and unwavering support throughout my journey. You have remained a source of wisdom and guidance that I can always rely on. You are also a shining beacon of kindness, an example you set not just within our department, but the robotics field as a whole. I fondly remember our meeting during the pandemic where I made a pitch for making causality a focus of my PhD thesis. The field had captured my imagination, and to this day, I remain steadfast in my conviction that it is a critical need for robot learning. That meeting, and all the ones after it, transformed what were only ideas into tangible contributions. This thesis would not have been possible without your support. I cannot express in words how much I am grateful for all that you have done for me during this journey. Thank you.

To my thesis committee, Prof. Oliver Kroemer, Prof. Shubham Tulsiani, Prof. Kun Zhang, and Prof. Jonas Peters: I have learned and grown so much from your guidance and expertise. I am deeply grateful for your thoughtful comments and wisdom in shaping this thesis to as strong as it can be. Each of you were vital in developing this thesis through your unique and important perspectives. Undoubtedly, this thesis is stronger because of you. Thank you.

To my funding agencies and sponsors, the U.S. Army Research Laboratory, Lockheed Martin Corporation, NVIDIA, and the U.S. Office of Naval Research: Our ideas and dreams came to fruition because of your generous support of our research. For this, I am dearly grateful. Thank you.

To the former and current members of the Intelligent Autonomous Manipulation Lab (Pat Callaghan, Samuel Clarke, Siddharth Girdhar, Prof. Oliver Kroemer, Alex LaGrassa, Jacky Liang, Janice Lee, Mark Lee, Steven Lee, Pragna Mannam, Tetsuya Narita, Rohan Pandya, Sarvesh Patil, Ami Sawhney, Saumya Saxena, Yunus Seker, Mohit Sharma, Zilin Si, Maximilian Sieb, Jeff Tan, Shivam Vats, Austin Wang, Xinyu

Wang, Vicky Zeng, Erin Zhang, Kevin Zhang, Wuming Zhang, and Alan Zhao): What a wonderful home the IAM Lab has been these past six years! I will never forget the many good memories I have of visiting the lab, working with robots, and, importantly, fostering friendships. I could not have asked for a more welcoming and inclusive home. Each of you has touched my life and guided me in all those times I sought advice. And, in those times when I was the one providing wisdom, I dearly hope it was helpful. With Prof. Kroemer's leadership, you are well-positioned to continue advancing the state-of-the-art in robot manipulation. Thank you.

To the members of the Resilient Intelligent Systems Lab (Curtis Boirum, Ellen Cappo, Micah Corah, Mosam Dabhi, Arjav Desai, Vishnu Desaraju, Aditya Dhawale, Logan Ellis, Vibhav Ganesh, Kshitij Goel, Mike Lee, Lauren Lieu, Prof. Nathan Michael, Derek Mitchell, Cormac O'Meadhra, Shaurya Shankar, Alex Spitzer, Shobhit Srivastava, Prof. Wennie Tabib, Xuning Yang, John Yao): My journey to the PhD started here as a MS student in 2015, where I began robotics research and got my feet wet (metaphorically and literally, working on an underwater perception project). I will look back at these years fondly. Not only were they instrumental in my growth as a researcher, my first friendships in Pittsburgh were forged here. In particular, I extend my heartfelt thanks to Prof. Nathan Michael for bringing me in to the lab and advising my MS thesis, and to my research sponsor, Westinghouse Electric Company (Lyman Petrosky and Nicholas Bhai), for supporting my MS research. I also wish to thank my MS thesis committee (Prof. Nathan Michael, Prof. Stephen Nuske, Eric Westman) for guiding my MS thesis. Thank you.

To the staff and my colleagues of the Robotics Institute (Arpit Agarwal, Prof. Chris Atkeson, Cornelia and Dominik Bauer, Ankit Bhatia, Abhijat Biswas, Rachel Burcin, Allie Chang, Nadine Chang, Arkadeep Chaudhury, Xianyi Cheng, Kiyn Chin, Maggie Collier, Achal Dave, Victoria Dean, Akshay Dharmavaram, Allie Del Giorno, Bart Duisterhof, B.J. Fecich, Carloz Gil, Rick Goldstein, Jaskaran Grover, Swami Gurumurthy, Jean Harpley, Prof. David Held, Cherie Ho, Prof. George Kantor, Azarakhsh Keipour, Terri Kent, Leo Keselman, Ashwin Khadke, Prof. Kris Kitani, Prof. Oliver Kroemer, Tushar Kusnur, Ashlyn Lacovara, Ryan Lee, Haohong Lin, Prof. Changliu Liu, Cecilia Morales, Tim Mueller-Sim, Shohin Mukherjee, Adithya Murali, Suzanne Muth, Ram Natarajan, Prof. Jean Oh, Brian Okorn, Dinesh Reddy, Suhail Saleem, Dhruv Saxena, Rosario Scalise, Prof. Jeff Schneider, Tanmay Shankar, Kate Shih, Sam Speer, Prof. Aaron Steinfeld, Sudharshan Suresh, Gokul Swamy, Zhi Tan, Ada Taylor, Prof. Zeynep Temel, Anirudh Vemula, Mrinal Verghese, Thomas Weng, Michelle Zhao, Rachel Zheng, and Wenxuan Zhou): What a wonderful, collaborative community with state-of-the-art robotics research and expertise! I am so grateful for the wonderful friendships that have formed over these nearly ten years. The Robotics Institute is truly a special place. In particular, I offer my thanks to my research qualifier committee (Prof. Oliver Kroemer, Prof. Chris Atkeson, Prof. Jeff Schneider, Brian Okorn) for shaping my research ideas as I began my journey. Additionally, I wish to thank B.J. Fecich and Suzanne Muth for their efforts in coordinating

the MS and PhD programs, respectively. Thank you.

To the CMU JEDI team (including Bailey Flanigan, Ananya Joshi, Pallavi Koppol, Sara McAllister, Samantha Reig, Catalina Vajiac), the SCS Dean's PhD Advisory Committee (including Leo Chen, Kiyn Chin, Victoria Dean, Kalil Anderson Garrett, Helen Zhou), and the SCS DEI Seminar Series team (Rachel Burcin, Prof. Ken Holstein, Gael Hyppolite, Prof. Queenie Kravitz): I am deeply grateful for the opportunity to work with you and endeavor towards making the School of Computer Science a more welcoming and inclusive place for everyone. I dearly wish that the seeds and ideals we planted will blossom beyond SCS and Pittsburgh, towards making the entire world a better place. Thank you.

To my internship colleagues at NVIDIA (Stan Birchfield, Jia Cheng, Prof. Dieter Fox, Terry Mosier, Thang To, Jonathan Tremblay) and Lockheed Martin (Chris Debrunner, Eric Dixon, Saman Fahandezhsaadi, Joseph Gleason, Daniel Kolosa, Shruti Mahadevan, John Steinbis): You have provided me wonderful, unique opportunities to pursue research with impactful applications. My internships occurred at pivotal moments in my PhD journey. In the summer of my first year, NVIDIA was my first opportunity with deep learning, allowing me to understand its powerful impact and planting research ideas that would later grow into my thesis. During my last year, Lockheed Martin offered a rare opportunity to pursue causality research for robotic applications, with similar conviction towards the technological capabilities of a robot learning system that is grounded in causality. I am a better researcher because of your guidance and what I have learned through these opportunities. Thank you.

To my workshop co-organizers for the Causality for Robotics workshop at IROS 2023 (Caleb Chuck, Jiaheng Hu, Prof. Oliver Kroemer, Sarvesh Patil, Zizhao Wang, Prof. Yuke Zhu) and the Causal-HRI workshop at HRI 2024 (Jiaee Cheong, Nikhil Churamani, Luke Guerdan, Prof. Hatice Gunes, Prof. Zhao Han): I am deeply fortunate for the opportunity to work with you and advocate for the importance of causality to the robotics community. I hope that our efforts led to important questions, conversations, and collaborations that will yield new research and directions in this nascent intersection of causality and robotics. I also wish to thank the speakers, contributors, and attendees of these workshops for your important contributions. Thank you.

To the organizers and hosts of my external talks with the University of Toronto (Prof. Igor Gilitschenski, Reinhard Grassmann, Claas Voelcker), University of Michigan GENDiR (Wami Ogunbi, Emily Sheetz, Andrea Sipos), and Talking Robotics (Sooraj Krishna, Riddhiman Laha, Nitin Ragothaman, Prof. Silvia Tulli): Thank you for such wonderful opportunities to share our research and the lessons we learned along the way. I hope that I effectively communicated our work and sparked interest in causal robot learning algorithms. I look back fondly on these talks with deep gratitude. Thank you.

To my colleagues and collaborators of the Learning By Doing competition (Dominik Baumann, Prof. Isabelle Guyon, Prof. Oliver Kroemer, Søren Wengel Mo-

gensen, Prof. Jonas Peters, Prof. Niklas Pfister, Prof. Sebastian Trimpe, Sebastian Weichwald, and the competition participants), Automated Curriculum Learning (Annya Dahmani, Prof. Alison Gopnik, Nan Rosemary Ke, Prof. Oliver Kroemer, Eunice Yiu), and at the University of Maryland (Prof. Yiannis Aloimonos, Pavan Mantripragada, Yantian Zha, Bomin Zhang): You have given your time, your efforts, and your attention to pursue important and novel research together. These collaborations have greatly enriched my PhD, and I hope they are just the start of continued collaboration for many years to come. Thank you.

To those whom I have had the privilege of learning from during my PhD (Montserrat Gonzalez Arenas, Yevgen Chebotar, Alexander D'Amour, and the CMU Eberly Center, including Alexis Adams, Sophie le Blanc, Phoebe Cook, Steven Moore, Patrick Walsh): Thank you for your advice, guidance, mentorship, and time. I am fortunate and grateful to be able to learn and grow from our discussions, from research ideas to teaching methods. Thank you.

To my medical, care, and support team: I cannot thank you enough for your profound and selfless care. You helped me embrace my best self, uplifted me when I was at my lowest, cared for me, gave me the gift of mobility, and taught me how to walk again. I owe you a debt of gratitude that I could never repay. Thank you.

To Gettysburg Foursquare Church, Metropolitan Community Church of Toronto, and Transmission Ministry Collective: I am grateful for your direction and wisdom, serving as a place where I could explore and strengthen my faith. Thank you.

To my friends, both in Pittsburgh and beyond (Rohit Acharya, Cara Bloom, Blake Buchanan, Chris and Lisa Cheok, Bryan Chu, Amanda and Jimmy Creegan, Dale, Jackson Dehn, Lena Dickinson, Jeanne, Janet and Jim, Kate, Kate and Nick Kleinschmidt, Grace and John Leatherman, Jill Marshall, Tanya Marwah, Dave Mitchell, Janice Morgan, Niusha and Tony, Kate Pepper, Luke Perry, Scott Perrygo, Sara Peterson, Katie Polak, Ralph, Samir, Heidi Stark, David Stebbins, Gail Stebbins, Jamila Sykes, Uttara, Pras Velagapudi, Ariel Wrigley, Srujana Rao Yarasi, Yuriko): Thank you for being there for me during this journey. I cherish the wonderful experiences we have had together, and I am so grateful for our friendship. I look forward to growing together with you for many years to come. Thank you.

To Mom, Dad, Becki, Noel White, Pastor Lisa Arrington, Rachel Nelson, Matt Ferry, the Ferry family, the Humphrey family, the Lee family, the Nelson family, the White family, and my extended family: I am forever grateful for your steadfast love and support not just during the PhD, but for all the years of my life. You are all so immeasurably important and special to me in ways that I cannot fully express in words. I love you. Thank you.

Lastly, to the source of my faith, God: Thank you for never leaving or forsaking me, for exemplifying perfect love, for leading me to joy (Isaiah 55:12). You have lifted me from the depths of my grief through a promise that all things work for good (Romans 8:28). Thank you. Soli Deo Gloria.

# Contents

# List of Tables

# List of Figures

# I

# Introduction

# 1 Introduction

## 1.1 Motivation

### 1.1.1 Two Realities of Deep Learning

Two decades into the third age of artificial intelligence (AI) [1], the rise of deep learning and massive neural networks has yielded two seemingly disparate realities. In one reality, deep learning has empowered massive accomplishments: solving chess, shogi, and Go with AlphaZero [2]; predicting protein structures with AlphaFold [3]; generating human-like language with GPT-4 and other large language models [4, 5]. In the other reality, OpenAI recently disbanded its robotics team, citing a lack of rich data.[1] Technology companies are resetting consumer expectations for less autonomous capabilities as the fully autonomous self-driving car remains yet out of reach [6]. The best-selling household robot remains a vacuum cleaner, which only recently ventured into (vision-based) machine learning [7]. Even manufacturing robots in factories struggle with integrating deep learning to move beyond simple, precise, and repetitive tasks [8, 9]. Even in these controlled environments, deep learning may be insufficiently robust when faced with seemingly inconsequential environmental changes such as lighting [8]. If a robot can learn dexterous manipulation for solving a Rubik's cube in the first reality [10], then why are there no robot butlers in the second [11]?

[1] Announced by Wojciech Zaremba, co-founder of OpenAI, on the June 3, 2021 episode of the Gradient Dissent: Conversations on AI podcast. Full quote: "I disbanded the robotics team... There [are] actually plenty of domains that are very, very rich with data [for unsupervised and reinforcement learning]. And ultimately, that was holding us back in the case of robotics. This decision was quite hard for me... But from the perspective of what we want to achieve, which is to build [Artificial General Intelligence], I think there was actually some components missing." [Link]

### 1.1.2 In an Open World, There Are No Unstructured Environments

A common motivation of roboticists to pursue deep learning is the powerful capability of representation learning. With representation learning, the hope is to overcome the challenges and costs of imbuing robots with human domain expertise via non-learning approaches, as well as achieve better generalization for problems in environments where it is challenging, or even impossible, for a human to even characterize a suitable representation. Representative examples of such environments include homes, restaurants, and public places, where there is high impact potential for useful automation work to improve human quality-of-life. Such environments have commonly been described within the robotics community as "unstructured environments." However, this terminology belies the true nature of these environments. Moreover, it forwards a notion that modern correlation-based, structure-agnostic deep learning may eventually — with enough data — take hold in this domain, much like advances elsewhere, in the first reality of deep learning.

This thesis forwards a different view: *there are no unstructured environments.* Adopting terminology from the vision community [12–14], these settings are best referred to as *open-world* environments.[2] As argued in a recent CVPR workshop [14], open-world settings pose tremendous challenges for learning approaches. Data follow a long-tailed distribution. Due to real-world novelty and progression, distribution shifts will occur, so test data may fall outside the training distribution. Even collecting training data in the first place is costly and, if not done safely, poses risk to the safety of nearby humans, the environment, and the robot itself. Indeed, in these domains, robot learning faces fundamentally different challenges than other, data-rich domains [15], leading to questions whether significant advancements can be achieved by applying modern machine learning techniques alone. Perhaps this view provides some explanation for the bifurcation of deep learning realities.

Yet, an explanation is not a path forward. If open-world environments are not unstructured, what are they? In the unstructured view of the real world, observed data arise through correlations. The key insight is that the observed data in these environments arise not despite structure, but *because* of it.

### 1.1.3 Causality: The Structure of Data

We now formalize this view through structural causal models using the principles of causality [16]. Our objective is to provide sufficient understanding of the language of causal inference for this thesis. To this end, we provide the following primer, which has been paraphrased from Peters, Janzing, and Schölkopf [17]. We include

[2] In some sense, the fact that this term arose from the vision community is unsurprising. Their domain is even more challenging due to the limited ability of real-world interaction. Our work posits to use just that — manipulation — to overcome these challenges.

occasional examples of these principles specifically for the robotics domain, which we hope aids both roboticists as well as causal inference experts who may be unfamiliar with robots. There are many important aspects of causal inference that are not covered in this primer, such as faithfulness, confounders, structural minimality, and so forth. For more information, we refer the reader to foundational texts in this area, such as those by Pearl [16, 18]; Spirtes, Glymour, and Scheines [19]; Imbens and Rubin [20]; Pearl, Glymour, and Jewell [21]; Peters, Janzing, and Schölkopf [17]; and Pearl and Mackenzie [22]. Additionally, we note that the formalism of structural causal models is but one framework out of others that may exist (e.g., potential outcomes [23, 24]).

**Primer: A Crash Course on Causality.** A structural causal model (SCM) $\mathfrak{C} \coloneqq (\mathbf{S}, P_N)$ is defined as a system of $d$ *structural assignments* (also referred to as "structural equations"):

$$X_j \coloneqq f_j(\mathbf{PA}_j, N_j), \quad j = 1, ..., d$$

where $\mathbf{X} = (X_1, ..., X_d)$ are random variables with distribution $P_{\mathbf{X}}$, the variables $\mathbf{PA}_j \subseteq \{X_1, ..., X_d\} \setminus \{X_j\}$ are parent variables of $X_j$, $f_j$ is the function (or process) that generates $X_j$ based on $\mathbf{PA}_j$, and $N_j$ are noise variables. Note that these variables are not "noise" in the sense of probabilistic estimates of uncertainty, e.g., a sensor reading having some zero-mean Gaussian distribution noise. Rather, they are exogenous variables, whose value is required and imposed upon the system. Importantly, we require the noise variables to all be independent. The joint distribution of the noise variables is $P_N$. The noise variables "drive" the system, as they are required to be instantiated first, before the calculation of structural assignments. This collection of structural assignments can be expressed as a directed acyclic graph (DAG), where the variables are nodes of this graph. The use of ':=' denotes these equations should *not* generally be treated as an equality, but rather, an assignment of the output of $f_j$ to $X_j$. It is for this reason that the terminology "structural assignments" may be preferred in some examples, to indicate that this is not an equation in the algebraic sense that can be rearranged.

From this SCM, a probability distribution of $\mathbf{X}$ is entailed. Such a distribution is referred to as the *observational distribution*. Learning the SCM model from data samples is known as *causal discovery*, *structure learning*, or *structure discovery*. In general, assumptions may be required to uniquely recover the SCM from observational data, depending on the problem. If the SCM can be uniquely determined, it is said to be *identifiable*. Depending on the assumptions made, however, sometimes the SCM can only be learned up to a certain DAG equivalence class.

With the current formulation, we can now introduce the concept of *interventions* as replacing a structural assignment with a different one. The distribution entailed from a SCM under intervention is called the *interventional distribution*. We usually

refer to the intervention in notation with the *do* operator, e.g., $do(X = 1)$. For example, intervening on a robot tactile sensor by touching and applying force would cause a corresponding measurement signal. However, intervening on the measurement signal — say, setting it to some value — would *not* cause a force to press the tactile sensor. In this way, conducting an intervention may provide a strong indicator for the structure of the underlying SCM, even if it is not known completely. This is the principle by which randomized control trials are conducted. Unsurprisingly, interventions can also be viewed as types of *experiments*.

Interventions can facilitate greater learning and reasoning capabilities than from observational data alone. Indeed, learning from observational data alone (without considering that data are generated by some causal process) is essentially correlation-based learning. However, interventions can elicit different patterns, depending on the relationship between variables, which provides a stronger "learning signal." Importantly, these interventions allow the robot to break spurious correlations — data patterns that are thought to be causal, but are not. Variables that are irrelevant to the problem at hand will induce spurious correlations, and interventions can assist the robot with determining which variables to use and which can be neglected.

Lastly, we introduce the concept of *counterfactuals*. Counterfactuals are generally intuitive for humans to pose. For example, "Would the robot have detected the cereal box correctly, if the lighting was dimmer?" is a counterfactual. Counterfactuals can be best explained as an intervention upon the SCM where some variables have already been observed (i.e., some of the noise variables have been instantiated). In the case of the cereal box, it requires some understanding of the scene to know how the cereal box would have behaved in this specific context (i.e., observational data), in order to make the interventional query of how that would change due to the effect of another variable (lighting).

### 1.1.4 From Correlational Learning to Causal Learning

In the causal perspective, data arise from data generating processes. Learning with respect to not only the data, but the processes that generated the data, holds promise for addressing many outstanding questions with most modern, correlation-based machine learning, such as how best to achieve out-of-distribution generalization, handle robustness to distribution shifts, and how to learn re-usable mechanisms that can transfer to new problems [25]. As recently argued by Schölkopf et al.,

> "...the majority of current successes of machine learning boil down to large scale pattern recognition on suitably collected independent and identically distributed (i.i.d.) data." (Schölkopf et al. [25])

Schölkopf et al. posit a path forward for the machine learning community through harnessing deep learning through the principles of graph-based causality and causal inference, thereby introducing *causal representation learning*.

In robot learning, such sentiments of the limitations of statistical learning have also been forwarded by Roy et al. [15]: "conventional machine learning and artificial intelligence do not adequately address the needs of an embodied agent [i.e., robot] that learns."

We believe these challenges can be addressed through causality. The advantages of adopting the principles of causal inference have been witnessed to date in fields such as biomedical sciences, economics, and genomics [26–28]. This thesis argues that causal inference is not only important for robot application, but more strongly, the principles of causal learning, in fact, empower robot learning.

Analogously to Schölkopf et al.'s appeal to the machine learning community [25], this thesis forwards that the robot learning community stands to gain by leveraging the insights of causal inference and causal representation learning through collaboration and cross-pollination. In so doing, this new paradigm holds promise for robots to learn and leverage structure within the open world through *causal robot learning for manipulation*.

## 1.2 Thesis Statement: Learning and Leveraging Structure within the Open World

We now introduce the statement of this thesis that frames the broader vision of this body of work (Fig. 1.1).

> *This thesis posits that, for robots to succeed in the open world, robot learning algorithms should learn and reason with respect to not only the observed data, but the processes that generate the data. Towards this end, the objective of this thesis is to develop robot learning algorithms that learn and leverage the causal structure of manipulation tasks. Consequentially, this thesis will synthesize the principles of causal inference and causal representation learning for* ***causal robot learning for manipulation****.*

## 1.3 Thesis Research Questions

The completed work of this thesis addresses the following research questions that entail this thesis statement.

R1. What are the relevant variables for a particular task or model? Where do these variables come from, and what is the relationship between them?

R2. What are the implications of distribution and domain shifts for learning? How do these shifts manifest in relevant variables? How can the sim-to-real gap be bridged?


Figure 1.1: The vision of this thesis is to develop learning algorithms that empower robots to perform useful automation work to improve human quality-of-life when deployed into the real world. Such real-world settings include (in column-wise order): residential kitchens and living rooms, assisted living centers, restaurants, community centers, and libraries, to name a few. We specifically refer to these settings as *open-world* environments. In the open world, environments are characterized by a rich diversity of objects, a sparsity of relevant objects for completing any particular task, and distribution shifts owing to yet-unseen settings, mechanisms, and long-tailed phenomena. These distinguishing factors stymie modern correlation-based deep learning. Yet, there exists common structure across these environments: books along the same row all sit at the same height of a shelf; chairs are stationary until they are pushed in; water flows from a kitchen faucet when the handle is rotated; prismatic cabinet drawers are constrained to move linearly along one axis; light is projected one-way into a camera to generate an image, not vice versa. Indeed, the open world is not "unstructured," as commonly described, but arises *because* of structure — from underlying causal processes that generate observed data for the robot to use. Therefore, our learning algorithms would benefit from learning and leveraging this structure towards realizing our vision of open-world robots. (Images licensed from Shutterstock.)

R3. How can dynamical systems be better understood in order to achieve greater control and understanding, including for both physical processes and learning processes?

R4. How will the robot know when learning is complete? Which tasks should be considered for bootstrapping the learning of a specific, target task?

R5. Can useful perception models arise from correlation-based learning with sufficient domain randomization?

## 1.4 Thesis Contributions

In total, this body of work will contain eight contributions, each of which are distinct papers and chapters of this thesis. Mappings of the research questions onto these contributions are provided below. These contributions are approximately chronologically ordered.

*Correlation-Based Transfer Learning for Perception:*

C1. **DREAM** [29], an approach for camera-to-robot pose estimation through synthetic sim-to-real transfer.
→ Chapter 2, *Research Questions: R2, R5*

C2. **FormNet** [30], an approach for visual identification of articulated objects through synthetic sim-to-real transfer.
→ Chapter 3, *Research Questions: R2, R5*

*Structural Sim-to-Real Transfer:*

C3. **CREST** [31], an algorithm for sim-to-real causal feature selection for manipulation policies using an internal model of a task.
→ Chapter 4, *Research Questions: R1, R2, R3*

C4. **SCALE** [32], an algorithm for causal learning of manipulation skills using simulation that captures the underlying data generating processes.
→ Chapter 5, *Research Questions: R1, R2, R3*

*Causality and Dynamical Systems:*

C5. **Learning By Doing (LBD)** [33], a competition to foster research in the intersection of causality, control theory, and reinforcement learning for controlling dynamical systems.
→ Chapter 6, *Research Questions: R1, R3*

C6. **LMeshNet**, a methodology for creating a hybrid causal world model that integrates semantic information and provides greater interpretability of the world model's latent space.
→ Chapter 7, *Research Questions: R1, R3*

*Curriculum Learning:*

C7. **ACL** [34–36], an investigation of the commonalities and differences in curriculum learning between humans and reinforcement learning agents. We note that the agent study is the specific thesis contribution in this work, although the human study has richly informed other contributions of the thesis.
→ Chapter 8, *Research Questions: R2, R3, R4*

C8. **CURATE**, an algorithm for curriculum learning for reinforcement learning agents and robots based on sample-based evaluations.
→ Chapter 9, *Research Questions: R2, R3, R4*

## 1.5 Thesis Conclusions and Takeaways

As a consequence of these contributions, we introduce four conclusions and takeaways of this thesis. These conclusions extend across multiple contributions and support the forwarded position of this thesis.

T1. We introduce structural sim-to-real transfer, extending prior use of simulators as statistical data generators currently used for sim-to-real transfer. In so doing, we demonstrate that simulation can serve as a causal reasoning engine that the robot can leverage in the open world. Through experimental demonstrations on real-world robots, we also demonstrate that the principles of causality can extend to robot applications in practice.
→ *Contributions: C3, C4*

T2. We introduce methods to identify, understand, and leverage the causal structure of a particular task or model, to better equip the robot with techniques to operate in the open world with its vast diversity of objects, features, and percepts.
→ *Contributions: C3, C4, C5, C6*

T3. We demonstrate various methodologies for transfer learning, particularly through distribution and domain shifts, such as in bridging the sim-to-real gap or curriculum learning.
→ *Contributions: C1, C2, C3, C4, C7, C8*

T4. We argue the position of integrating the principles of causal inference and causal representation learning into robot learning to specifically address the challenges

of open-world manipulation. Although success may be found for certain problems with approaches that are agnostic to the underlying causal structure, such approaches may be either too brittle (due to distribution shifts or spurious correlations) or, for more complex problems, require intractable amounts of data or domain randomization. Our contributions demonstrate that methods that learn and leverage the causal structure of tasks can endow robots with richer capabilities that should lead to greater performance and success in the open world. In so doing, we argue towards the adoption of *causal robot learning for manipulation* through the eight contributions of the thesis.
→ *Contributions: C1, C2, C3, C4, C5, C6, C7, C8*

## 1.6 Thesis Outline

The contributions of this thesis are structured in four parts, where each part consists of two works.

**Part II**, Correlation-Based Transfer Learning for Perception, pertains to initial work in this thesis for learning perception models using sim-to-real transfer learning. Notably, these works explored bridging the sim-to-real gap with only synthetic observations with domain randomization. The first work is DREAM (**Chapter 2**), where sim-to-real transfer learning enabled camera-to-robot pose estimation for the camera calibration problem. Our related work, FormNet (**Chapter 3**), is the "spiritual successor" of DREAM, applying these ideas for visual identification of articulation mechanisms in real-world environments. This part serves as a reminder that correlation-based techniques may yield successful models, although their robustness, sample efficiency, and utility may still be improved through training objectives and approaches that enforce greater causal structure. Moreover, training models primarily through vast domain randomization may be intractable for problems with more complex data generating processes, necessitating approaches that emphasize structure, rather than being agnostic to it.

**Part III**, Structural Sim-to-Real Transfer, primarily considers the eponymous methodology that learns how to construct the causal structure of a policy or skill in simulation, and then transfer this structure to reality. The works in this part leverage the concept of a simulator being more than a statistical data generator, as commonly used in the sim-to-real community. Instead, simulation functions as a causal reasoning engine for the robot to use in the open-world setting. These works include causal feature selection (CREST) in **Chapter 4** and causal learning of skills (SCALE) in **Chapter 5**.

**Part IV**, Causality and Dynamical Systems, concerns the rich overlap between causality and dynamical processes. Learning By Doing (**Chapter 6**) makes clear the commonalities between causality, control theory, and reinforcement learning by examining control of a chemical process as well as a robot manipulator. LMeshNet

(**Chapter 7**) introduces an approach for hybridization of a world model (i.e., a model of environment dynamics) that integrates and synchronizes latent and semantic information. For hybrid causal world models, by leveraging the favorable disentanglement properties of the latent space, causal discovery can be used to provide greater interpretability of the latent space based on a provided semantic space.

**Part V**, Curriculum Learning, contains the final two thesis works. These works explore curriculum learning as supported by insights from human cognition, towards our goal of causal curriculum learning. ACL (**Chapter 8**) investigates the similarities and contrasts in curriculum learning between agents and humans. Humans need not completely solve a task before advancing to a more difficult task, seemingly due to maintaining an internal measure of progress achieved in the task. On the other hand, agents may experience catastrophic forgetting due to the inherent distribution shifts of curriculum learning, but such training divergence may be mitigated (or, in some cases, recovered from) using a human-inspired auxiliary reward based on progress. These insights into human cognition have been key towards the development of CURATE (**Chapter 9**), our curriculum learning algorithm. We plan on demonstrating the generality of CURATE for a variety of agent domains, including a robot insertion task. Moreover, we are currently investigating ways to extend CURATE with greater causal reasoning and learning capabilities, towards a causal curriculum learning algorithm. Such a capability for automated curriculum learning would be useful to boost the generalization and sample efficiency of learning how to complete a particular target task that could arise in the open-world setting.

Following the thesis contributions, the thesis concludes in **Chapter 10**. In this chapter, we summarize and reflect on the themes of the thesis and provide future directions that may emerge from the thesis works. Finally, we conclude by leaving a vision for the thesis reader for the broader implications of causal robot learning as it pertains to causal embodied intelligence.

## 1.7 A Note on Terminology

*What does it mean for a method to be "causal"?* For certain learning problems, the framing of causality is apparent. This would be the case for many environmental relationships, such as learning that turning a switch on a kitchen stove causes a burner on top of the stove to ignite. Yet, what about robot actions in the environment, such as grasping and moving a frying pan on top of the burner? These actions are also causal, but what is the advantage of such a framing, and how does it compare to existing methods for learning actions? To readers who are new to causality, we now provide a short discussion on terminology to allay confusion before discussing the thesis contributions in detail in later chapters.

Consider the field of reinforcement learning. As discussed in Learning By Doing (Chapter 6), reinforcement learning agents learn to perform actions, which can

also be seen as causal interventions within the environment. Therefore, is reinforcement learning already "causal"? Kaddour et al. argues that although the answer is technically yes, the approaches developed by the reinforcement learning and causality communities have different underlying goals [37]. The reinforcement learning field emphasizes the maximization of rewards, whereas causality literature focuses on learning and leveraging causal structure through identifiability and inference [37]. For a specific example of a causal reinforcement learning method, in Causal Dynamics Learning for Task-Independent State Abstraction, a causal dynamics model that learns the relationships between state variables can provide better generalization and sample efficiency for model-based reinforcement learning [38]. Meanwhile, a non-causal (i.e., correlation-based) approach for reinforcement learning would usually be agnostic to any structure that may exist within the environment that the agent could have otherwise used for richer learning outcomes.

We believe the key to dispelling confusion is to refer to approaches as "causal" if they explicitly relate to *causal structure*: the underlying data generating processes of the problem. Structure-based approaches seek to understand causal variables, create causal representations, learn structural causal models, assess whether a causal effect is identifiable, provide causal inferences, and offer model-based explanations. The fact that causality can tie in broadly to many problems, although occasionally a source of terminology confusion, is a testament to the generality of causality for reasoning, learning, and structuring information.

# II

# Correlation-Based Transfer Learning for Perception

# 2 DREAM: Camera-to-Robot Pose Estimation from a Single Image through Synthetic Sim-to-Real Transfer

This section presents DREAM (**D**eep **R**obot-to-camera **E**xtrinsics for **A**rticulated **M**anipulators), a correlation-based, sim-to-real transfer learning approach for learning a perception model. Specifically, DREAM's key capability is a deep neural network that predicts robot keypoints from a single RGB image. This network is trained with only synthetic data that contain domain randomizations, such as robot joint configuration, camera pose, scene lighting, and distractor objects. The predicted keypoints of the network are used as input to a geometric computer vision algorithm that predicts the camera-to-robot pose. The accuracy of DREAM (approx. 2cm in robot workspace accuracy) is sufficient to solve the camera calibration problem for manipulation of objects that do not require fine precision. Moreover, DREAM matches the state-of-the-art (hand-eye calibration) in accuracy, while also significantly lowering the operational costs of camera calibration. Instead of carefully collecting several images with a fiducial marker for hand-eye calibration, DREAM only needs a minimum of one RGB image of the robot — no fiducial marker needed.

Although this work did not specifically pertain to causal learning, we offer DREAM as a case study and point of comparison. DREAM is a representative methodology for correlation-based, sim-to-real transfer learning, which was the state-of-the-art when this work was conducted in 2018. The strategy for bridging the sim-to-real gap with the DREAM network is through large quantities of domain-randomized synthetic data. With a sufficiently large number of synthetic images (on the order of 50,000 to 100,000 images), DREAM's neural network learns a representation that transfers successfully to real camera images. A similar strategy was employed in FormNet (Ch. 3), a follow-up work to DREAM that estimates the articulation properties of real-world household objects using one RGB-D observation frame.

In contrast to causal learning, which is focused on learning with respect to an underlying data generating process, DREAM's correlation-based approach requires enough data variations to break spurious correlations so that the underlying process is learned. A version of DREAM that uses causal learning would likely be capable of training the network with greater sample efficiency and robustness to broader distributions beyond the limited laboratory environments used to evaluate DREAM. However, we leave "Causal DREAM" for future work.

*Sections of the remainder of this chapter first appeared in [29]. This work was completed during an internship at NVIDIA and was presented at the 2020 IEEE International Conference on Robotics and Automation (ICRA 2020). We thank our collaborators on this work: Jonathan Tremblay, Thang To, Jia Cheng, Terry Mosier, Prof. Oliver Kroemer, Prof. Dieter Fox, and Stan Birchfield.*

## 2.1 Summary

*We present an approach for estimating the pose of an external camera with respect to a robot using a single RGB image of the robot. The image is processed by a deep neural network to detect 2D projections of keypoints (such as joints) associated with the robot. The network is trained entirely on simulated data using domain randomization to bridge the reality gap. Perspective-n-point (PnP) is then used to recover the camera extrinsics, assuming that the camera intrinsics and joint configuration of the robot manipulator are known. Unlike classic hand-eye calibration systems, our method does not require an off-line calibration step. Rather, it is capable of computing the camera extrinsics from a single frame, thus opening the possibility of on-line calibration. We show experimental results for three different robots and camera sensors, demonstrating that our approach is able to achieve accuracy with a single frame that is comparable to that of classic off-line hand-eye calibration using multiple frames. With additional frames from a static pose, accuracy improves even further. Code, datasets, and pretrained models for three widely-used robot manipulators are made available:* `https://research.nvidia.com/publication/2020-03_DREAM`

## 2.2 Introduction

Determining the pose of an externally mounted camera is a fundamental problem for robot manipulation, because it is necessary to transform measurements made in camera space to the robot's task space. For robots operating in unstructured, dynamic environments—performing tasks such as object grasping and manipulation, human-robot interaction, and collision detection and avoidance—such a transformation allows visual observations to be readily used for control.

The classic approach to calibrating an externally mounted camera is to fix a fiducial marker (*e.g.*, ArUco [39], ARTag [40], or AprilTag [41]) to the end effector, collect several images, then solve a homogeneous linear system for the unknown transformation [42]. This approach is still widely used due to its generality, flexibility, and the availability of open-source implementations in the Robot Operating System (ROS). However, such an approach requires the somewhat cumbersome procedure of physically modifying the end effector, moving the robot to multiple joint configurations to collect a set of images (assuming the marker-to-wrist transformation is not known), running an off-line calibration procedure, and (optionally) removing the fiducial marker. Such an approach is not amenable to online calibration, because if the camera moves with respect to the robot, the entire calibration procedure must be repeated from scratch.

A more recent alternative is to avoid directly computing the camera-to-robot transform altogether, and instead to rely on an implicit mapping that is learned for the task at hand. For example, Tobin *et al.* [43] use deep learning to map RGB images to world coordinates on a table, assuming that the table at test time has the same dimensions as the one used during training. Similarly, Levine *et al.* [44] learn hand-eye coordination for grasping a door handle, using a large-scale setup of real robots for collecting training data. In these approaches the learned mapping is implicit and specific to the task/environment, thus preventing the mapping from being applied to new tasks or environments without retraining.

We believe there is a need for a general-purpose tool that performs online camera-to-robot calibration without markers. With such a tool, a researcher could set up a camera (*e.g.*, on a tripod), and then immediately use object detections or measurements from image space for real-world robot control in a task-independent manner, without a separate offline calibration step. Moreover, if the camera subsequently moved for some reason (*e.g.*, the tripod were bumped accidentally), there would be no need to redo the calibration step, because the online calibration process would automatically handle such disturbances.

In this paper, we take a step in this direction by presenting a system for solving camera pose estimation from a single RGB image. We refer to our framework as *DREAM* (for Deep Robot-to-camera Extrinsics for Articulated Manipulators). We train a robot-specific deep neural network to estimate the 2D projections of prespecified keypoints (such as the robot's joints) in the image. Combined with camera intrinsics, the robot joint configuration, and forward kinematics, the camera extrinsics are then estimated using Perspective-$n$-Point (P$n$P) [45]. The network is trained entirely on synthetic images, relying on domain randomization [43] to bridge the reality gap. To generate these images, we augmented our open-source tool, NDDS [46], to allow for scripting robotic joint controls and to export metadata about specific 3D locations on a 3D mesh.

This paper makes the following contributions:

- We demonstrate the feasibility of computing the camera-to-robot transforma-

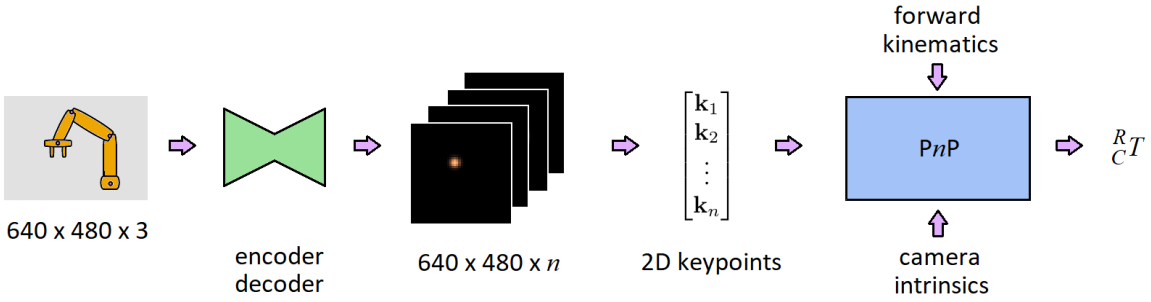



Figure 2.1: The DREAM framework. A deep encoder-decoder neural network takes as input an RGB image of the robot from an externally-mounted camera, and it outputs $n$ belief maps (one per keypoint). The 2D peak of each belief map is then extracted and used by P$n$P, along with the forward kinematics and camera intrinsics, to estimate the camera-to-robot pose, ${}^{R}_{C}T$.

tion from a single image of the robot, *without fiducial markers*, using a deep neural network trained only on synthetic data.

- We show that the resulting accuracy with a single real image frame is comparable to that of classic hand-eye calibration using multiple frames. For a static camera, accuracy further improves with multiple image frames.
- Quantitative and qualitative results are shown for three different robot manipulators (Franka Emika's Panda, Kuka's LBR iiwa 7 R800, and Rethink Robotics' Baxter) and a variety of cameras.

The simplicity of our approach enables real-time pose estimation on a desktop computer with an NVIDIA Titan Xp GPU, without manual optimization.

## 2.3 Approach

An externally mounted camera observes $n$ keypoints $\mathbf{p}_i \in \mathbb{R}^3$ on various robot links. These keypoints project onto the image as $\mathbf{k}_i \in \mathbb{R}^2$, $i = 1, \ldots, n$. Some of these projections may be inside the camera frustum, whereas others may be outside. We consider the latter to be invisible/inaccessible, whereas the former are visible, regardless of occlusion.[1] The intrinsic parameters of the camera are assumed known.

Our proposed two-stage process for solving the problem of camera-to-robot pose estimation from a single RGB image frame is illustrated in Fig. 2.1. First, an encoder-decoder neural network processes the image to produce a set of $n$ belief maps, one

[1]The network learns to estimate the positions of occluded keypoints from the surrounding context; technically, since the keypoints are the robot joints (which are inside the robot links), they are always occluded.

per keypoint. Then, Perspective-$n$-Point (P$n$P) [45] uses the peaks of these 2D belief maps, along with the forward kinematics of the robot and the camera intrinsics, to compute the camera-to-robot pose, ${}^{R}_{C}T$. Note that the network training depends only on the images, not the camera; therefore, after training, the system can be applied to any camera with known intrinsics. We restrict $n \geq 4$ for stable P$n$P results.

### 2.3.1 Network Architecture

Inspired by recent work on object pose estimation [47–49], we use an auto-encoder network to detect the keypoints. The neural network takes as input an RGB image of size $w \times h \times 3$, and it outputs an $\alpha w \times \alpha h \times n$ tensor, where $w = 640$, $h = 480$, and $\alpha \in \{1, \frac{1}{2}, \frac{1}{4}\}$, depending on the output resolution used. This output captures a 2D belief map for each keypoint, where pixel values represent the likelihood that the keypoint is projected onto that pixel.

The encoder consists of the convolutional layers of VGG-19 [50] pretrained on ImageNet. The decoder (upsampling) component is composed of four 2D transpose convolutional layers (stride = 2, padding = 1, output padding = 1), and each layer is followed by a normal $3 \times 3$ convolutional layer and ReLU activation layer. We also experimented with a ResNet-based encoder, *viz.*, our reimplementation of [51], with the same batch normalization, upsampling layers, and so forth as described in the paper.

The output head is composed of 3 convolutional layers ($3 \times 3$, stride = 1, padding = 1) with ReLU activations with 64, 32, and $n$ channels, respectively. There is no activation layer after the final convolutional layer. The network is trained using an L2 loss function comparing the output belief maps with ground truth belief maps, where the latter are generated using $\sigma = 2$ pixels to smooth the peaks.

### 2.3.2 Pose Estimation

Given the 2D keypoint coordinates, robot joint configuration with forward kinematics, and camera intrinsics, P$n$P [45] is used to retrieve the pose of the robot, similar to [47–49, 52, 53]. The keypoint coordinates are calculated as a weighted average of values near thresholded peaks in the output belief maps (threshold = 0.03), after first applying Gaussian smoothing to the belief maps to reduce the effects of noise. The weighted average allows for subpixel precision.

### 2.3.3 Data Generation

The network is trained using only synthetic data with domain randomization (DR) and image augmentation [54]. Despite the traditional challenges with bridging the reality gap, we find that our network generalizes well to real-world images, as we will show in the experimental results. To generate the data we used our open-source

NDDS tool [46], which is a plugin for the UE4 game engine. We augmented NDDS to export 3D/2D keypoint locations and robot joint angles, as well as to allow control of the robot joints.

The synthetic robot was placed in a simple virtual 3D scene in UE4, viewed by a virtual camera. Various randomizations were applied: 1) The robot's joint angles were randomized within the joint limits. 2) The camera was positioned freely in a somewhat truncated hemispherical shell around the robot, with azimuth ranging from $-135°$ to $+135°$ (excluding the back of the robot), elevation from $-10°$ to $75°$, and distance from 75 cm to 120 cm; the optical axis was also randomized within a small cone. 3) Three scene lights were positioned and oriented freely while randomizing both intensity and color. 4) The scene background was randomly selected from the COCO dataset [55]. 5) 3D objects from the YCB dataset [56] were randomly placed, similar to flying distractors [57]. 6) Random color was applied to the robot mesh.

Figure 2.2 shows representative images from synthetic datasets generated by our approach. Domain randomized (DR) datasets were used for training and testing; datasets without domain randomization (non-DR) were used for testing to assess sim-to-sim generalization. The DR training data was generated using the publicly available robot models, whereas the non-DR test data used models that were artistically improved to be more photorealistic.

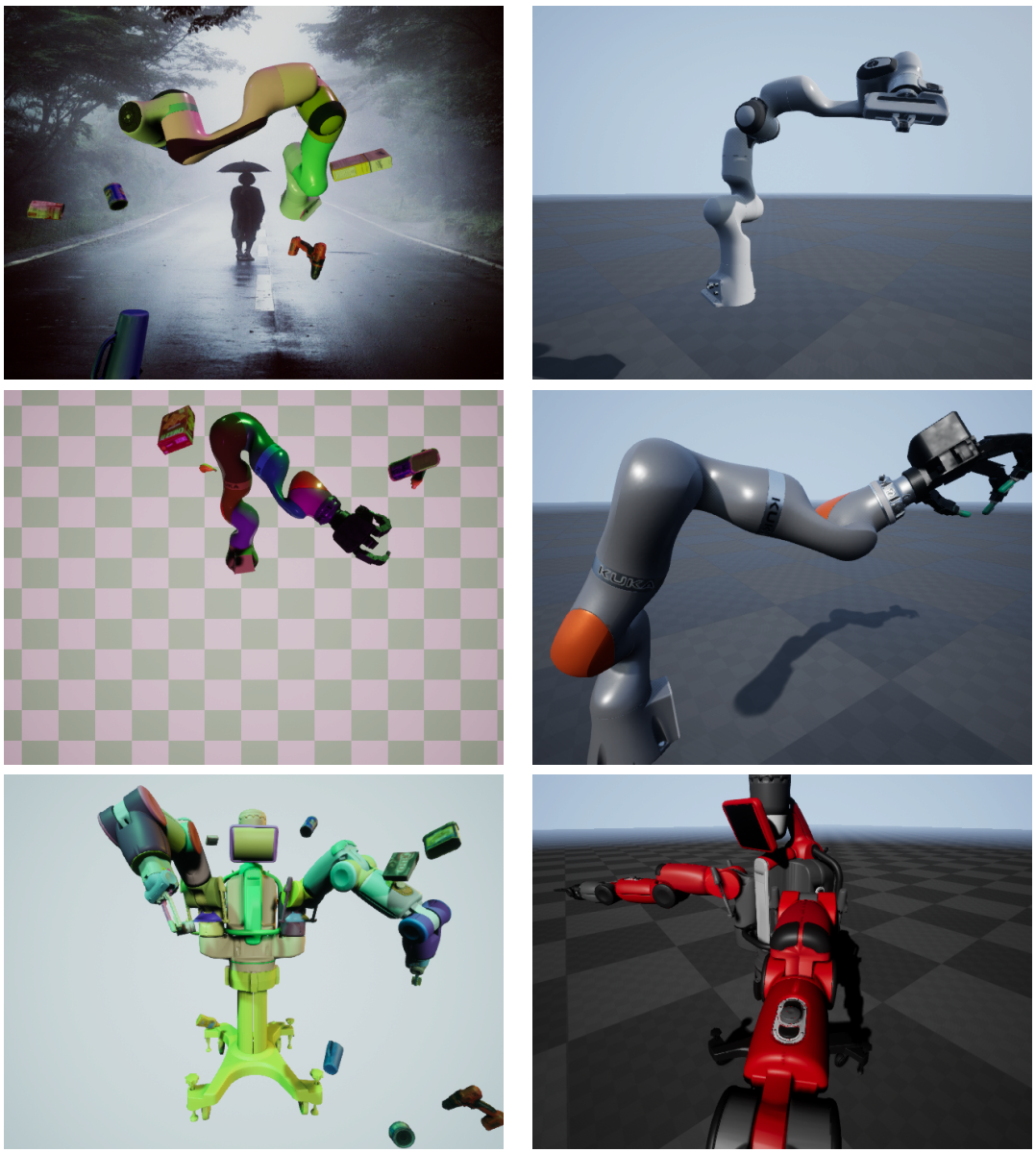
## 2.4 Experimental Results

In this section we evaluate our DREAM method both in simulation and in the real world. We seek to answer the following questions: 1) How well does the synthetic-only training paradigm transfer to real-world data? 2) What is the impact of various network architectures? 3) What accuracy can be achieved with our system, and how does it compare to traditional hand-eye calibration?

### 2.4.1 Datasets

We collected several real-world datasets in our lab using various RGBD sensors. Since our DREAM algorithm processes only RGB images, the depth images were captured solely to facilitate ground truth camera poses via DART [58], a depth-based articulated tracking algorithm. DART was initialized manually and monitored in real-time during data collection to ensure tracking remained stable and correct, by viewing the projection of the robot coordinate frames onto the camera's RGB images via the ROS visualization tool (RViz).

**Panda-3Cam.** For this dataset, the camera was placed on a tripod aimed at the Franka Emika Panda robot arm. The robot was moved to five different joint configurations, at which the camera collected data for approximately five seconds each, followed by manual teleoperation to provide representative end effector trajectories



domain-randomized (DR) non-DR

Figure 2.2: Synthetic training images for the three robot models: Franka Panda (top), Kuka LBR with Allegro hand (middle), and Rethink Baxter (bottom).

along cardinal axes, as well as along a representative reaching trajectory. During data collection, neither the robot base nor the camera moved. The entire capture was repeated using *three different cameras* utilizing different modalities: Microsoft XBox 360 Kinect (structured light), Intel RealSense D415 (active stereo), and Microsoft Azure Kinect (time-of-flight). All images were recorded natively at $640 \times 480$ resolution at 30 fps, except for the Azure Kinect, which was collected at $1280 \times 720$ and downsampled to $640 \times 480$ via bicubic interpolation. This dataset consists of 17k total image frames divided approximately equally between the three cameras.

**Panda-Orb.** To evaluate the method's ability to handle a variety of camera poses, additional data was captured of the Panda robot. The RealSense camera was again placed on a tripod, but this time at 27 different positions in a roughly *orbital* motion around the robot (namely, 9 different azimuths, ranging from roughly -180° to +180°, and for each azimuth two different elevations approximately 30° and 45°, along with a slightly closer depth at 30°). For each camera pose, the robot was commanded using Riemannian Motion Policies (RMPs) [59, 60] to perform the same joint motion sequence of navigating between 10 waypoints defined in both task and joint space. The dataset consists of approximately 40k image frames.

### 2.4.2 Metrics

Metrics were computed on both 2D and 3D. For the 2D evaluation, the percentage of correct keypoints (PCK) [61] below a certain threshold was calculated, as the threshold varied. All keypoints whose ground truth was within the camera's frustum were considered, regardless of whether they were occluded. For 3D evaluation of the accuracy of the final camera-to-robot pose, the average distance (ADD) [62, 63] was calculated, which is the average Euclidean distance of all 3D keypoints to their transformed versions, using the estimated camera pose as the transform. ADD is a principled way, based upon Euclidean geometry, to combine rotation and translation errors without introducing an arbitrary weighting between them. As with PCK, the percentage of keypoints with ADD lower than a threshold was calculated, as the threshold varied. For ADD, all keypoints were considered. In both cases, averages were computed over keypoints over all image frames.

### 2.4.3 Training and Simulation Experiments

For comparison, we trained three versions of our DREAM network. As described earlier, the architecture uses either a VGG- or ResNet-based encoder, and the decoder outputs either full (F), half (H), or quarter (Q) resolution. Each neural network was trained for 50 epochs using Adam [64] with 1.5e-4 learning rate and 0.9 momentum. Training for each robot used approximately 100k synthetic DR images. The best-performing weights were selected according to a synthetic validation set.

As a baseline, Fig. 2.3 compares these versions on two simulated datasets, one with

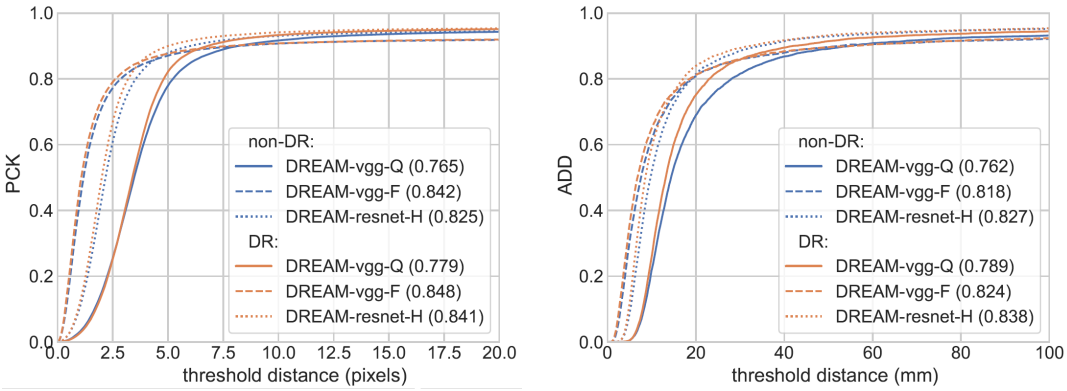



Figure 2.3: PCK (left) and ADD (right) results for three different variants of our DREAM network on the two simulated datasets. The numbers in parentheses are the area under the curve (AUC).

domain-randomization (DR) and the other without (non-DR). The improvement due to increasing resolution is clear, but different architectures have only minimal impact for most scenarios.

### 2.4.4 Real-world Experiments

Results comparing the same three networks on the Panda-3Cam dataset are shown in Fig. 2.4. Encouragingly, these results show that our training procedure is able to bridge the reality gap: There is only a modest difference between the best performing network on simulated and real data.

For 3D, it is worth noting that the standard deviation of ground truth camera pose from DART was 1.6 mm, 2.2 mm, and 2.9 mm, respectively, for the XBox 360 Kinect (XK), RealSense (RS), and Azure Kinect (AK) cameras, respectively. The degraded results for the Azure Kinect are due to DART's sensitivity to noise in the time-of-flight-based depth, rather than to DREAM itself. On the other hand, the degraded results for XBox 360 Kinect are due to the poor RGB image quality from that sensor.

Ultimately, the goal is to be able to place the camera at an arbitrary pose aimed at a robot, and calibrate automatically. To measure DREAM's ability to achieve this goal, we evaluated the system on the Panda-Orb dataset containing multiple camera poses. These results, alongside those of the previous experiments, are shown in Tab. 2.1. For this table we used DREAM-vgg-F, since the other architectures performed similarly as before.

We also trained DREAM on the Kuka LBR iiwa and Baxter robots. While the

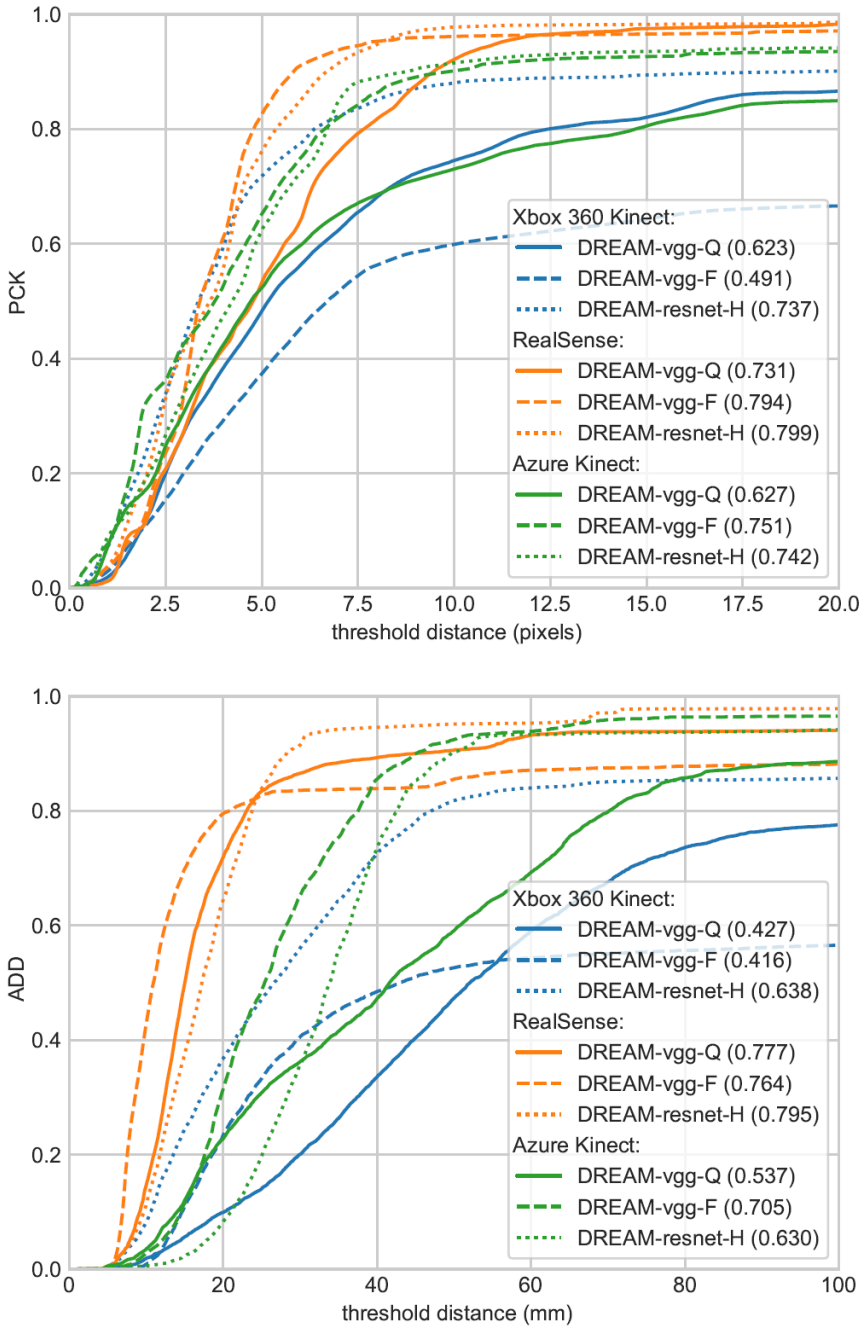



Figure 2.4: PCK (top) and ADD (bottom) results on the real Panda-3Cam dataset.

Table 2.1: Results at different thresholds of the same network (DREAM-vgg-F) on various datasets (and camera sensors). All but the last row are taken from Figs. 2.3 and 2.4.

| | PCK @ (pix) | | | ADD @ (mm) | | |
|---|---|---|---|---|---|---|
| Dataset | 2.5 | 5.0 | 10.0 | 20 | 40 | 60 |
| Sim. DR | 0.79 | 0.88 | 0.90 | 0.81 | 0.88 | 0.90 |
| Sim. non-DR | 0.77 | 0.87 | 0.90 | 0.80 | 0.88 | 0.90 |
| Panda-3Cam (XK) | 0.15 | 0.37 | 0.59 | 0.23 | 0.48 | 0.54 |
| Panda-3Cam (RS) | 0.24 | 0.83 | 0.96 | 0.80 | 0.83 | 0.87 |
| Panda-3Cam (AK) | 0.36 | 0.65 | 0.90 | 0.32 | 0.86 | 0.94 |
| Panda-Orb (RS) | 0.28 | 0.67 | 0.83 | 0.57 | 0.77 | 0.80 |

former is similar to the Panda, the latter is more difficult due to symmetry that causes the two arms to be easily confused with one another. To alleviate this problem, we had to restrict the azimuth range from -45° to +45°. Although we did not perform quantitative analysis on either robot, qualitatively we found that the approach works about the same for these robots as it does for the Panda. The detected keypoints overlaid on images of the three robots, using three different RGB cameras, are shown in Fig. 2.5. Although in principle the keypoints could be placed anywhere on the robot, we simply assign keypoints to the joints according to each robot's URDF (Unified Robot Description Format).

### 2.4.5 Comparison with Hand-Eye Calibration

The goal of our next experiment was to assess the accuracy of DREAM versus traditional hand-eye calibration (HEC). For the latter, we used the `easy_handeye` ROS package[2] to track an ArUco fiducial marker [39] attached to the Panda robot hand.

The XBox 360 Kinect was mounted on a tripod, and the robot arm was moved to a sequence of $M = 18$ different joint configurations, stopping at each configuration for one second to collect data. Neither the camera nor the base of the robot moved. The fiducial marker was then removed from the hand, and the robot arm was moved to a different sequence of $M$ joint configurations. The joint configurations were selected favorably to ensure that the fiducial markers and keypoints, respectively, were detected in the two sets of images. As before, DART with manual initialization was used for ground truth.

Although our DREAM approach works with just a single RGB image, it can potentially achieve greater accuracy with multiple images by simply feeding all detected keypoints (from multiple frames) to a single P$n$P call. Thus, to facilitate a direct

[2] `https://github.com/IFL-CAMP/easy_handeye`

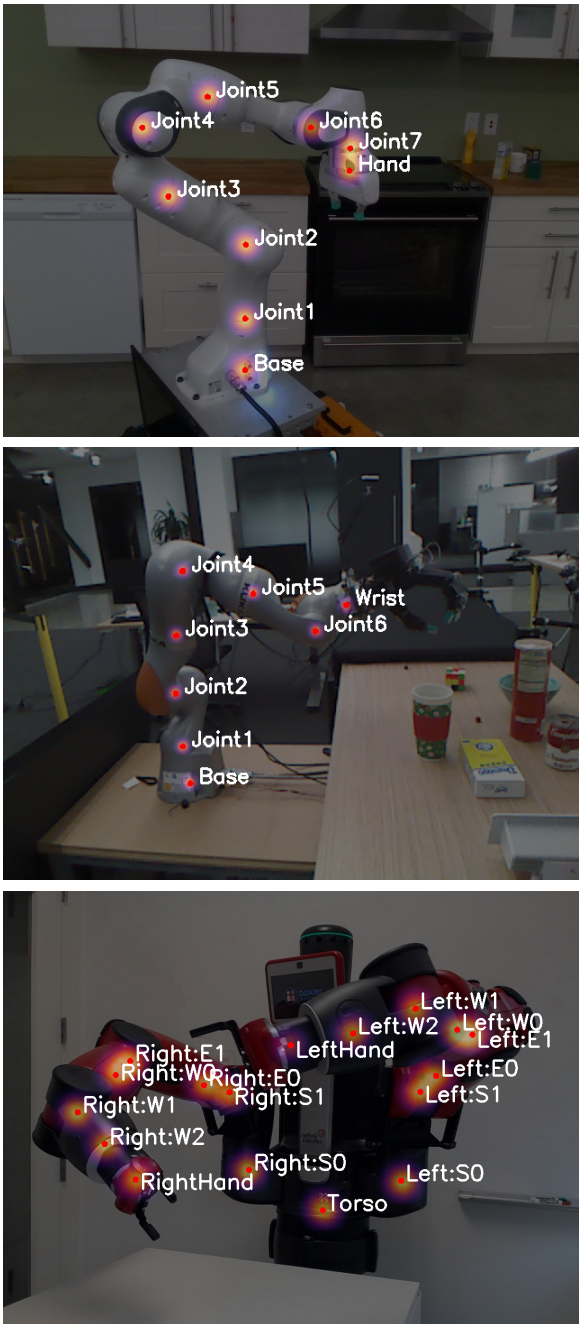



Figure 2.5: Keypoint belief maps (red dots indicate peaks) detected by DREAM in RGB images of three different robots (taken by three different cameras). From left to right: Franka Emika Panda (Intel RealSense D415), Kuka LBR iiwa (Logitech C920 webcam), and Rethink Baxter (cell phone camera).

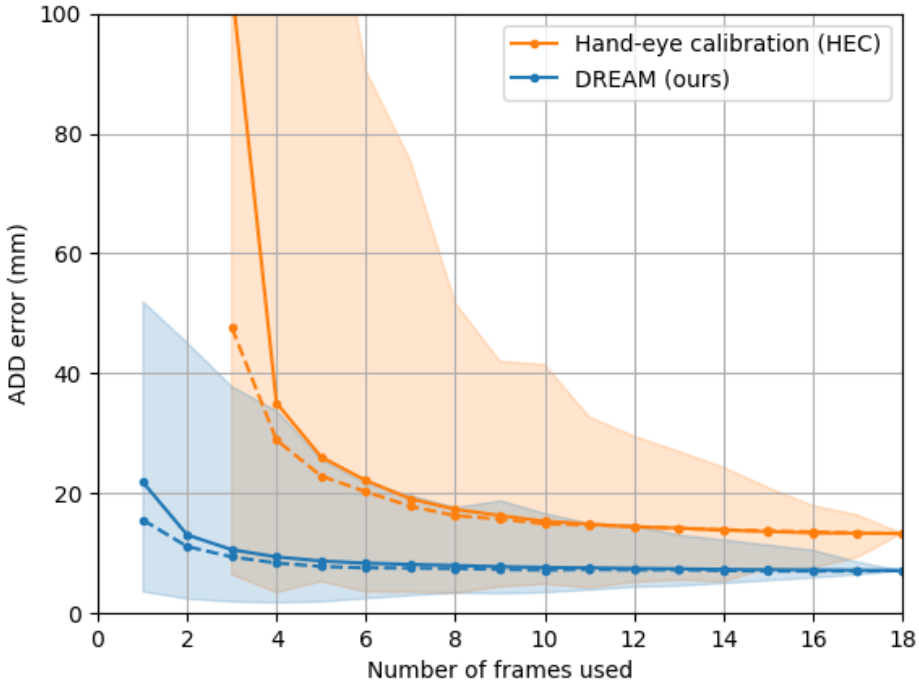



Figure 2.6: DREAM vs. HEC, measured by ADD as a function of the number of image frames used for calibration. Shown are the mean (solid line), median (dashed line), and min/max (shaded area), computed over different image combinations. DREAM requires only a single image frame but achieves greater accuracy with more images.

comparison with HEC, we applied DREAM to $m \geq 1$ images from the set of $M$ images that were collected. Similarly, we applied HEC to $m \geq 3$ images from the set. Both algorithms were then evaluated by comparing the estimated pose with the ground truth pose via ADD. For both algorithms, we selected $\binom{M}{m}$ possible combinations when evaluating the algorithm on $m$ images, to allow the mean, median, min, and max to be computed. To avoid unnecessary combinatorial explosion, whenever this number exceeded $N = 2500$, we randomly selected $N$ combinations rather than exhaustive selection.

Results of this head-to-head comparison are shown in Fig. 2.6. Note that HEC is unable to estimate the camera pose when $m < 3$, whereas DREAM works with just a single image. As the number of images increases, the estimated pose from both DREAM and HEC improves, depending somewhat on the robot configurations used. In all cases, DREAM performs as well or better than HEC. (Note, however, that HEC results would likely improve from manually, rather than randomly, selecting image frames.)

Table 2.2: Euclidean error between the robot's actual reached position and the commanded position, using the camera pose estimated by each of the three methods.

| | DART | HEC | DREAM (ours) |
|---|---|---|---|
| camera | depth | RGB | RGB |
| no. frames | 1 | 10 | 1 |
| min error (mm) | 10.1 | **9.4** | 20.2 |
| max error (mm) | 44.3 | 51.3 | **34.7** |
| mean error (mm) | **21.4** | 28.2 | 27.4 |
| std error (mm) | 12.3 | 14.2 | **4.7** |

### 2.4.6 Measuring Workspace Accuracy

In the final experiment we evaluated the accuracy of DREAM's output with respect to the workspace of the robot. The RealSense camera was placed on a tripod facing the Panda robot reaching toward an adjustable-height table on which were placed five fiducial markers. A head-to-head comparison of the camera poses computed by DART, DREAM, and HEC was conducted by transforming each fiducial marker's pose from the camera's frame to the robot's frame by applying each algorithm's camera-to-robot pose estimate. The robot was then commanded to move the end effector to a target position defined 10 cm directly above the marker, to avoid potential collision. This process was repeated for ten target positions (5 markers, 2 table heights). The Euclidean distance between the end effector's position in 3D was measured for each algorithm. Note that in this experiment DART was not considered to be ground truth, but rather was compared against the other methods.

Results are shown in Tab. 2.2. Even though DREAM is RGB-only, it performs favorably not only to HEC but also to the depth-based DART algorithm. This is partly explained by the fact that the extrinsic calibration between the depth and RGB cameras is not perfect. Note that DREAM's error is similar to that of our previous work [52] upon which it is based, where we showed that an error of approximately 2 cm for object pose estimation from RGB is possible.

## 2.5 Previous Work

Relationship to previous work is considered in this section.

**Object pose estimation.** In robotics applications, it is not uncommon for objects to be detected via fiducial markers [65–67]. Even so, there is growing interest in the problem of markerless object pose estimation in both the robotics and computer vision communities [47–49, 52, 53, 62, 63, 68, 69], building upon work in keypoint detection for human pose estimation [51, 70–73]. Recent leading methods are similar to the approach proposed here: A network is trained to predict object keypoints in

the 2D image, followed by P$n$P [45] to estimate the pose of the object in the camera coordinate frame [48, 49, 52, 53, 74, 75], or alternatively, a deformable shape model is fit to the detect keypoints [76]. Indeed, our approach is inspired by these methods. Our approach builds upon the findings of Peng *et al.* [48], who showed that regressing to keypoints on the object is better than regressing to vertices of an enveloping cuboid. Other methods have regressed directly to the pose [63, 69], but care must be taken not to bake the camera intrinsics into the learned weights.

**Robotic camera extrinsics.** Closely related to the problem of estimating the camera-to-object pose (just described) is that of estimating the camera-to-robot pose. The classic solution to this problem is known as hand-eye calibration, in which a fiducial marker (such as ArUco [39], ARTag [40], AprilTag [41], or otherwise known object) is attached to the end effector and tracked through multiple frames. Using forward kinematics and multiple recorded frames, the algorithm solves a linear system to compute the camera-to-robot transform [77–79]. Similarly, an online calibration method is presented by Pauwels and Kragic [80], in which the 3D position of a known object is tracked from nonlinear optimization over multiple frames.

An alternate approach is to move a small object on a table, command the robot to point to each location in succession, then use forward kinematics to calibrate [81]. However, the accuracy of such an approach degrades as the robot moves away from the table used for calibration. Aalerund *et al.* [82] present a method for calibrating an RGBD network of cameras with respect to each other for robotics applications, but the camera-to-robot transforms are not estimated.

For completeness, we mention that, although our paper addresses the case of an externally mounted camera, another popular configuration is to mount the camera on the wrist [83], for which the classic hand-eye calibration approach applies [80]. Yet another configuration is to mount the camera on the ceiling pointing downward, for which simple 2D correspondences are sufficient [81, 84, 85].

**Robotic pose estimation.** Bohg *et al.* [86] explore the problem of markerless robot arm pose estimation. In this approach, a random decision forest is applied to a depth image to segment the links of the robot, from which the robot joints are estimated. In follow-up work, Widmaier *et al.* [87] address the same problem but obviate the need for segmentation by directly regressing to the robot joint angles. Neither of these approaches estimate the camera-to-robot transform.

The most similar approach to ours is the simultaneous work of Lambrecht and Kästner [88, 89], in which a deep network is also trained to detect projected keypoints, from which the camera-to-robot pose is computed via P$n$P. A key difference is that our network is trained only on synthetic data, whereas theirs requires real and synthetic data. In other recent work, Zuo *et al.* [90] also present a keypoint-based detection network. But rather than use P$n$P, nonlinear optimization directly regresses to the camera pose and the unknown joint angles of a small, low-cost manipulator. The network is trained using synthetic data, with domain adaptation to bridge the reality gap.

## 2.6 Conclusion

We have presented a deep neural network-based approach to compute the extrinsic camera-to-robot pose using a single RGB image. Compared to traditional hand-eye calibration, we showed that our DREAM method achieves comparable accuracy even though it does not use fiducial markers or multiple frames. Nevertheless, with additional frames, our method is able to reduce the error even further. We have presented quantitative results on a robot manipulator using images from three different cameras, and we have shown qualitative results on other robots using other cameras. We believe the proposed method takes a significant step toward robust, online calibration. Future work should be aimed at filtering results over time, computing uncertainty, and incorporating the camera pose into a closed-loop grasping or manipulation task.

## 2.7 Acknowledgments

We gratefully acknowledge Karl van Wyk, Clemens Eppner, Chris Paxton, Ankur Handa, and Erik Leitch for their help. Many thanks also to Kevin Zhang and Mohit Sharma (Carnegie Mellon University).

# 3 FormNet: Visual Identification of Articulated Objects from a Single Image Observation through Synthetic Sim-to-Real Transfer

This section presents FormNet (**F**low of **O**bject **R**esidual **M**otion **Net**work). Like DREAM (Ch. 2), FormNet is also a correlation-based, sim-to-real transfer learning approach for learning a perception model. FormNet identifies the kinematic mechanism between pairs of articulated object parts using only one image observation containing an RGB-D image and segmentation mask. Visual identification of such articulated objects is highly useful for robot manipulation in open-world settings, as many objects in these settings — such as doors, cabinets, drawers, and windows — are articulated. In evaluation for novel object instances in trained categories, FormNet achieves 82.5% accuracy. Importantly, FormNet demonstrated zero-shot sim-to-real transfer for real-world image observations.

In many ways, FormNet is the "spiritual successor" of DREAM. FormNet's architecture was inspired by the DREAM architecture, and FormNet also blends together a model-free technique (estimating object residual motion with a neural network) with a model-based technique (estimating articulation parameters based on the network's prediction of residual motion). It was also trained similarly, using 100,000 synthetic images with domain randomizations of background, texture, lighting, and pose. Through sufficient domain randomization, this methodology yielded a model that could complete this perception task. It is hypothesized that incorporating causal learning into FormNet would also improve the sample efficiency and robustness to broader distributions. However, like "Causal DREAM", we leave such improvements for future work.

*Sections of the remainder of this chapter first appeared in [30]. Additional information is provided in App. A. We thank Vicky Zeng, who was first author and presented this work at the 2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2021). We thank our other collaborators on this work: Jacky Liang and Prof. Oliver Kroemer.*

## 3.1 Summary

*As autonomous robots interact and navigate around real-world environments such as homes, it is useful to reliably identify and manipulate articulated objects, such as doors and cabinets. Many prior works in object articulation identification require manipulation of the object, either by the robot or a human. While recent works have addressed predicting articulation types from visual observations alone, they often assume prior knowledge of category-level kinematic motion models or sequence of observations where the articulated parts are moving according to their kinematic constraints. In this work, we propose FormNet, a neural network that identifies the articulation mechanisms between pairs of object parts from a single frame of an RGB-D image and segmentation masks. The network is trained on 100k synthetic images of 149 articulated objects from 6 categories. Synthetic images are rendered via a photorealistic simulator with domain randomization. Our proposed model predicts motion residual flows of object parts, and these flows are used to determine the articulation type and parameters. The network achieves an articulation type classification accuracy of 82.5% on novel object instances in trained categories. Experiments also show how this method enables generalization to novel categories and can be applied to real-world images without fine-tuning.*

## 3.2 Introduction

Reliable, autonomous robots have many potential applications as assistants to humans in settings such as homes, businesses, and hospitals [11, 91–95]. One prerequisite for these applications is the capability for robots to both recognize and manipulate **articulated** objects: objects that have moving parts that are kinematically linked with each other, such as doors, windows, drawers, caps, dials, buttons, and switches. For example, a robot tasked with fetching medicine must identify and interact with several articulated objects: opening a door to enter a room, searching a cabinet of drawers for a medicine bottle, twisting the bottle cap open, and then delivering the contents. Manually specifying articulation constraints for the vast diversity of objects is intractable, so it is important for a robot to autonomously identify these constraints and their parameters.

Interactive Perception (IP) is a well-known approach to this problem [96]. With IP, the robot interacts with objects in the environment through physical contact,

Figure 3.1: Predicting articulation mechanisms using FormNet. Given the inputs on the left (RGB-D image and segmentation masks of a pair of object parts), a neural network predicts the articulation type (revolute, prismatic, fixed, or unconnected) and appropriate articulation parameters (e.g., location and direction of revolute axis). The network is trained on synthetic data and infers articulation parameters via predicting motion residual flows.

observes the changes, and predicts the underlying kinematic constraints. For example, the robot may pull on a handle, and if the handle's trajectory forms a straight line, then the constraint is prismatic (e.g., drawer); if the handle follows an arc, then the constraint is revolute (e.g., door). Prior IP works often share the limitation of not leveraging the object's *visual* features — either at all [97–100] or only as a contextual prior to exploration [101]. Most articulated objects that humans interact with are designed to visually signify their articulation affordances through pronounced geometry and texture. An elongated bar with one contact at the end to another surface is probably a revolute handle; a cabinet handle with a connection at each end is likely prismatic.

Other approaches consider visual aspects while identifying articulation mechanisms. Existing works in this direction track the relative movements of object parts from videos and predict the constraint types [102–108]. However, these still require the articulated object to be manipulated — an onerous assumption for robots operating in novel environments.

The contribution of our work is FormNet[1]: a neural network model that identifies articulation mechanisms between objects with only single-frame vision observations, no interactions, and no pre-specified object category model (Fig. 3.1). We leverage recent developments in high-quality object mesh datasets[2] that contain information about both object parts and their relative kinematic constraints. Color and depth (RGB-D) images, as well as part segmentation masks, are collected across six object categories found in PartNet-Mobility [109]. These categories include common household objects like doors, windows, and cabinets. Training images are rendered in simulation with domain randomization. To predict articulation type, we represent and encode the displacement of an object part under a given articulation mechanism as motion residual flow. The network uses convolutions to predict this flow and the parts' "connectedness" (i.e., whether two object parts are connected), which we then post-process with RANSAC to form the articulation type prediction. We evaluate the performance of the trained network with ablation studies across multiple object categories, and we also demonstrate that it can predict articulation types of objects in real-world images without further fine-tuning. See supplementary materials at `https://sites.google.com/view/articulated-objects/home`

## 3.3 Related Work

Our work relates to two broad classes of vision-based object perception for (1) articulation constraints and (2) pose estimation in manipulation settings.

**Visual identification of object articulation constraints.** To identify articulation mechanisms via visual observations, the authors of [110] manually labeled a large dataset with motion parameters, such as the location and axis of revolute and prismatic joints. Then, they proposed using motion-driven features and losses to train neural networks that jointly solves for motion-driven part segmentation and motion parameters. Here, motion parameters were encoded as displacement and orientation residuals, corresponding to prismatic and revolute joints. However, this method assumes access to the *complete* point cloud of an object, not a partial or noisy point cloud that would be found with egocentric depth sensors used in most robotic manipulation applications.

Later works relaxed the assumption of complete point clouds, but they leveraged knowledge of a set of predefined articulated object categories and their kinematic models. For example, doors would be one category, and cabinets another. Knowledge of object categories allow these algorithms to fit predefined geometric and kinematic models to the observed visual features, which are often point clouds. For instance, in [111] the authors formed Gaussian mixture models over six predefined kinematic

[1]FormNet stands for **F**low of **O**bject **R**esidual **M**otion **Net**work. The name alludes to the *form*, or shape, of object motion under articulation constraints.

[2]`https://sim2realai.github.io/Synthetic-Datasets-of-Objects-Part-I/`

models, and they trained a neural network to predict parameters of the mixture model from single depth images. The parameters include kinematic model parameters for each category, the object's joint configurations, and geometry parameters (e.g., door length). Training data is generated in simulation, and the model generalizes to novel objects within known categories.

In [112], the authors forgo mixture of Gaussians by proposing the articulation-aware normalized coordinate space hierarchy, a canonical representation for each articulated object category. Within this representation, object scales, orientations, and articulation parameters are normalized, allowing a neural network to directly regress to coordinates in this space. The proposed model uses PointNet++ to process point clouds extracted from depth images, and depth image data for training is also generated in simulation.

The authors of ScrewNet [113] removed the assumption of known object categories or kinematic models. Instead, they represent the relative motion of point clouds as a screw motion: rotation of a body around an axis coupled with a translation in the axis. ScrewNet is a neural network that directly predicts parameters of this screw motion between articulated parts, inferring articulation type without known kinematic models. However, to make this prediction, the network requires a sequence of depth images, with the articulated object parts moving relative to each other.

Procrustes-Lo-RANSAC (PLR) [114, 115] similarly predicts articulation types without *a priori* kinematic models. PLR leverages a geometric vision-based algorithm instead of a neural network, but it requires observations of the object in two distinct articulation configurations.

Like [113–115], our work does not require prior definitions of category-level kinematic models. Instead, our proposed method uses *a single image observation* to predict articulation type; no observations of part motion are required.

**Vision-based object pose estimation for manipulation.** Our work can be viewed as a form of vision-based pose estimation, wherein we infer the constrained poses that the connected parts of an object would move under the predicted articulation kinematics. In this view, our work relates to pose estimation in manipulation settings, which includes both objects [48, 52, 63, 116] and robots [29, 86, 87, 117]. Of these works, PVNet [48] also regresses to a residual (pointing to object keypoints for pose estimation), whereas our motion residual is used to infer the kinematic constraint. For practical manipulation scenarios, we also note the success of DART [58], a depth-based tracking algorithm for articulated models. Our learning-based model facilitates single-image estimation of articulation motion that generalizes without needing a model specification for novel objects, as DART does. Lastly, our system can be integrated into a vision-based manipulation pipeline, such as the one described in [118]. Instead of object pick-and-place, our approach would facilitate object articulated motion, such as opening cabinets and doors.

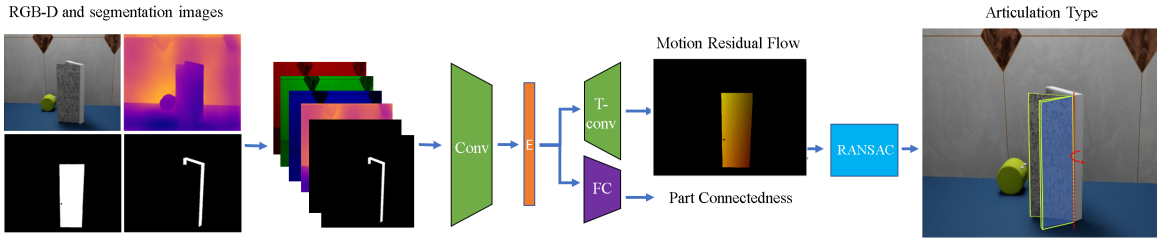



Figure 3.2: Neural network architecture for FormNet. It takes as input the RGB, depth, and segmentation images of two queried object parts. Conv means convolution layers, E means intermediate embedding, T-conv means transposed convolution layers, and FC means fully-connected layers. The network produces two outputs: motion residual flow and binary part connectedness. If the two queried parts are predicted to be connected, plane fitting via RANSAC is used to post-process the motion residual flows to estimate the articulation mechanism's type and parameters.

## 3.4 Method

In this section, we describe the proposed neural network model (FormNet) for vision-based identification of articulations of object parts, how its training data is generated, and the representations of the model's inputs and outputs.

### 3.4.1 Overview

Our approach identifies kinematic constraints between pairs of object parts from a stationary visual observation alone. We construct a neural network model that takes as input a single-view RGB-D image and segmentation masks of two distinct object parts. We focused on designing and training this network, and assume the part segmentation masks are provided from a pre-existing algorithm (e.g., [119]). The neural network provides two outputs for the segmented parts: (1) a parts connectedness classification and (2) the motion residual flow, which is the displacement of the second part relative to the first part if the second part moves under their kinematic constraint. Two parts are connected if they are parts of the same articulated object and are neighbors of one another in the object's kinematic chain. The joint for two connected parts is classified as revolute, prismatic, or fixed, depending on the predicted motion residual flow. Furthermore, we apply RANSAC on the predicted motion residual flow to extract articulation parameters for revolute (rotation axis) and prismatic (direction of movement) joints.

By formulating our network to infer the articulation type between a pair of object parts, our approach works with images containing an arbitrary number of articulated objects. Therefore, predefined categorical models or shared coordinate spaces of articulated objects are not needed; the network does not need to reason about

| | ***Door*** | ***Window*** | ***Faucet*** | ***Dishw.*** | ***Fridge*** | ***Cab.*** |
|---|---|---|---|---|---|---|
| **Objects** | 30 | 28 | 23 | 30 | 20 | 18 |
| **Parts** | 92 | 112 | 115 | 92 | 98 | 109 |
| **Type** | R | P | R | R | R | P,R |

Table 3.1: Dataset Statistics. **Objects** shows the number of object models. **Parts** shows the total number of distinct object parts. **Type** lists the type of articulations that exist in that category, where R = revolute and P = prismatic. **Dishw.** stands for Dishwasher, and **Cab.** stands for Cabinet.

multiple articulation mechanisms belonging to specific object categories. In addition, this formulation also allows our model to generalize to object categories unseen during training. The entire kinematic chain of an articulated object in an image can be recovered by querying the network with all pairs of object parts.

The proposed model is trained via a large dataset of synthetically rendered images of articulated objects. Domain randomization and augmentations are applied to the training images, facilitating a network that is robust and invariant to changes in viewpoint, lighting, textures, occlusions, and object joint configurations. Models of articulated objects came from PartNet-Mobility, from which we filtered and cleaned models to form our training set.

### 3.4.2 Dataset of Articulated Objects

We considered several public datasets of objects with object part information for making our training data, including RBO [120], ShapeNet [121], and PartNet [122]. To train a generalizable articulation identification model, an object dataset is needed that contains a wide variety of object categories, a large number of diverse objects within each category, and labels of articulation types between connected object parts. The RBO dataset is a collection of 14 objects with 358 RGB-D interaction video sequences. While it provides articulation and part segmentation, the relatively small size makes it inadequate as training data. ShapeNet consists of over 3 million 3D CAD models, of which 220K are classified to 3135 categories. PartNet contains roughly 26K models across 24 object categories with good part segmentation. While ShapeNet and PartNet are sufficiently large, both datasets lack articulation information.

In recent works, Shape2Motion [121] and PartNet-Mobility [109] have augmented ShapeNet and PartNet to include articulation information. Shape2Motion is large with over 2.4K objects. However, it is not amenable to simulation and rendering; it lacks joint limit information and object textures. PartNet-Mobility does not face such limitations and has our desired properties: the dataset has over 2.4K objects across 46 categories, object textures, and articulation information with joint limits.

As such, we use PartNet-Mobility to generate the training data for our articulation prediction model (Table 3.1).

We processed meshes from PartNet-Mobility to choose (1) categories with strong visual signals in kinematic constraints and (2) characteristic subsets of objects from each chosen category. The six categories chosen were doors, windows, faucets, dishwashers, refrigerators and cabinets. In each category, we choose a representative subset of objects that maintains intra-category variability. In total, 149 objects were selected. We cleaned the selected mesh models by scaling meshes to realistic sizes and standardizing the orientation of their coordinate frames. The former makes rendered images more realistic (i.e., faucets are typically smaller than refrigerators). The latter ensures that objects of the same categories appear in similar poses when loaded for rendering. Lastly, we removed object parts and articulation connections that were too detailed. For example, the interior racks of dishwasher models were ignored in our dataset.

### 3.4.3 Dataset of Scene Images with Articulated Objects

The network is trained using synthetic data with domain randomization and image augmentation. Data was generated using NVIDIA Isaac Sim[3], a GPU-accelerated robotics simulator that supports photorealistic rendering. Articulated objects were loaded into a clean virtual scene and several randomizations were applied, including camera pose, scene lighting, object pose, size, texture, and distractor objects.

To make the trained articulation prediction model generalizable across a wide variety of scenes, we perform domain randomizations to render the synthetic dataset. The camera is positioned randomly within the front upper hemisphere of the object, a region where a robot is likely to be to perform manipulation. Joint configurations of articulated objects are randomized within their joint limits. Object sizes are also randomly scaled between 0.5 and 2.0 during generation. In addition, we randomize object textures and scene lighting. These randomizations accommodate for intra-category variation. While household items have diverse appearances (i.e., the width, color, and length of doors might differ), these particular differences do not affect the underlying articulation mechanisms. Scaling and changing the visual properties of the objects in the training data makes the model invariant to such details. Lastly, distractor objects consisting of common objects and household items are included to produce natural occlusions of articulated object parts.

In total, approximately 100K image scenes were rendered, each including an RGB-D image of resolution $640 \times 480$, an object part segmentation image, and the articulation information of objects in the scene.

In addition to domain randomization, we apply standard image augmentation techniques during training, including geometric transformations such as random rotations, flips, and crops [54]. For RGB images, we perform random visual transfor-

[3] `https://developer.nvidia.com/isaac-sim`

mations such as contrast and brightness. We also add realistic noises to depth and segmentation masks to make the trained model more robust. For depth images, we apply additive correlated Gaussian noise and multiplicative gamma noise to simulate realistic depth sensor noise [85]. To add realistic noise to the boundaries of the binary segmentation masks, we apply salt and pepper noise followed by a morphological closing operation.

Each data sample during training is generated with a pair of object parts in a rendered image. We first pick a pair of object parts from the segmentation image. The network input then consists of the RGB-D image of the entire scene and two binary segmentation masks of the chosen pair of object parts. The output consists of a binary part-connectedness label of the corresponding pair as well as the motion residual flow of the second part relative to the first part. The motion residual flow is a $W \times H \times 3$ image, where each pixel is only non-zero if it occupies a pixel belonging to the second object part. See Figure 3.4 for some examples. In these non-zero pixels, the values of each pixel correspond to where the corresponding point on the object part in 3D space would be if the object part is moved by a fixed magnitude following its kinematic constraint with the other part. The direction of the motion is expressed in the camera frame. For fixed joints, the motion residual is 0. For revolute joints, the movement magnitude is 30°. For prismatic joints, the movement magnitude is $0.3M$, where $M$ is the maximum joint movement distance provided by PartNet-Mobility. The exact magnitudes of these movement offsets are not important, since the network is not tasked with learning the range of motion of articulated objects, just their articulation types.

### 3.4.4 Network Architecture

The neural network for our approach is an hourglass encoder-decoder architecture similar to the network used for DREAM [29]. As shown in Fig. 3.2, the network takes as input a stacked image observation of size $640 \times 480 \times 6$, with 4 RGB-D channels and 2 part-segmentation mask channels. The network predicts the motion residual flow as a $640 \times 480 \times 3$ image and binary part-connectedness label.

The image encoder consists of the convolutional layers of VGG19 pretrained on ImageNet [123]. The decoder (upsampling) module has four 2D transpose convolutional layers (stride = 2, padding = 1, output padding = 1), and each layer is followed by a normal $3 \times 3$ convolutional layer and ReLU activation layer. The first output head is the part connectedness, consisting of 3 fully connected layers. The second output head for the motion residual flow is composed of 3 convolutional layers ($3 \times 3$, stride = 1, padding = 1) with ReLU activations with 64, 32, and 3 channels, respectively. There is no activation layer after the final convolutional layer.

The network is trained with a Cross Entropy loss on part connectedness and a Mean Square Error loss on the motion residual flow. Let $y_n^c \in \{0, 1\}$ and $y_n^f \in \mathbb{R}^{640\times480\times3}$ respectively denote the binary part-connectedness and motion residual

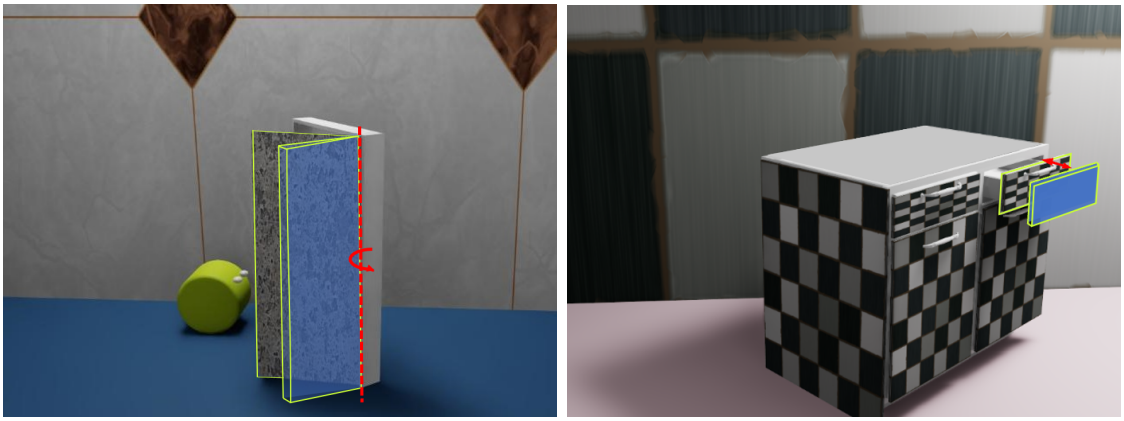


Figure 3.3: Visualizations of estimated pre-motion and post-motion planes. In both images, pre-motion planes are outlined in bright green on the original object part, while the post-motion planes are visualized with a blue infill. In the left image, the red annotations denote the estimated axis of the predicted revolute joint. In the right image, the red arrow denote the direction of the prismatic joint.

flow of the $n$th training sample, and $\hat{y}_n^c, \hat{y}_n^f$ be their estimated counterparts produced by the neural network. The weighted loss function on the two outputs is defined as:

$$\mathbf{SE}(y_n^f, \hat{y}_n^f) = \|y_n^f - \hat{y}_n^f\|^2 \tag{3.1}$$

$$\mathbf{CE}(y_n^c, \hat{y}_n^c) = -y_n^c \log(\hat{y}_n^c) + (1 - y_n^c) \log(1 - \hat{y}_n^c) \tag{3.2}$$

$$\mathcal{L} = \frac{1}{N} \sum_{n=1}^{N} w_{se} y_n^c \mathbf{SE}(y_n^f, \hat{y}_n^f) + w_{ce} \mathbf{CE}(y_n^c, \hat{y}_n^c) \tag{3.3}$$

Note that we only propagate the motion residual squared error loss when the parts are connected. The weights $w_{se} = 0.6$ and $w_{ce} = 0.4$ were chosen after hyperparameter search.

### 3.4.5 Articulation Prediction from Motion Residual Flows

If the network predicts that the two queried object parts are connected, then we process the predicted motion residual flow to robustly estimate the part articulation type and parameters. Algorithm pseudocode can be found in the supplementary materials. First, a plane is fitted on the point cloud of the object part in the input observation. We refer to this as the pre-motion plane. Next, we fit a second plane on the point cloud of the object part, where each point is translated by the predicted motion residual flow. We refer to this as the post-motion plane. Refer to Fig. 3.3 for examples of revolute and prismatic planes.

For fitting both planes, RANdom Sample Consensus (RANSAC) [124] is used to obtain robust estimations given depth and segmentation noise that exists in the

network inputs as well as estimation errors in the network outputs.

The articulation type and parameters can be inferred through comparison of the position and orientation of the pre-motion and post-motion planes. If the planes are sufficiently close together (i.e., the predicted motion residuals are all close to 0), then the predicted articulation type is fixed. Otherwise, if the pre-motion and post-motion planes are sufficiently parallel, then the articulation type is prismatic. In this case, the direction of the prismatic kinematic constraint is the direction of the average motion residual flow. Lastly, if the motion residuals are not all close to 0 and the planes are not parallel, then the predicted articulation type is revolute. In this case, the axis and location of the revolute joint is the line where the pre-motion and post-motion planes intersect. Extracting articulation parameters from motion residual planes in this manner allows the network to learn a single output representation that works for fixed, revolute, and prismatic joints.

## 3.5 Experiments

We evaluate our network on synthetic images and show successful transfer to real-world data. Specifically, we report (1) test accuracy achieved with our network with synthetic images when trained on all object categories, (2) generalization to categories unseen during training in a leave-one-out fashion, and (3) generalization to real-world images. See Fig. 3.4 for representative qualitative results. An ablation study is further conducted to train the network on each single category and test against all others, to analyze knowledge transfer between categories.

### 3.5.1 Network Training

The neural network was implemented with PyTorch and optimized via the Adam optimizers with a learning rate of $1.2 \times 10^{-4}$ and momentum of 0.9. These were tuned via hyperparameter search. The training set consisted of 70k images, and the remaining 30k were used in the test set. The network took 30 epochs to train, taking 32 hours on an NVIDIA Tesla V100 GPU.

### 3.5.2 Network Accuracy (All Object Categories)

To evaluate the accuracy of the classified articulation types, we separate the predictions into 4 classes: prismatic, revolute, fixed, and unconnected. We refer to this as combined accuracy (**CA**). Two additional metrics are evaluated: accuracy over part connectedness (**PC**) and accuracy over connected articulation type (**AT**). The former is the binary classification accuracy of whether or not two parts are connected. The latter is only the articulation type accuracy when the network predicts a true positive part connectedness.

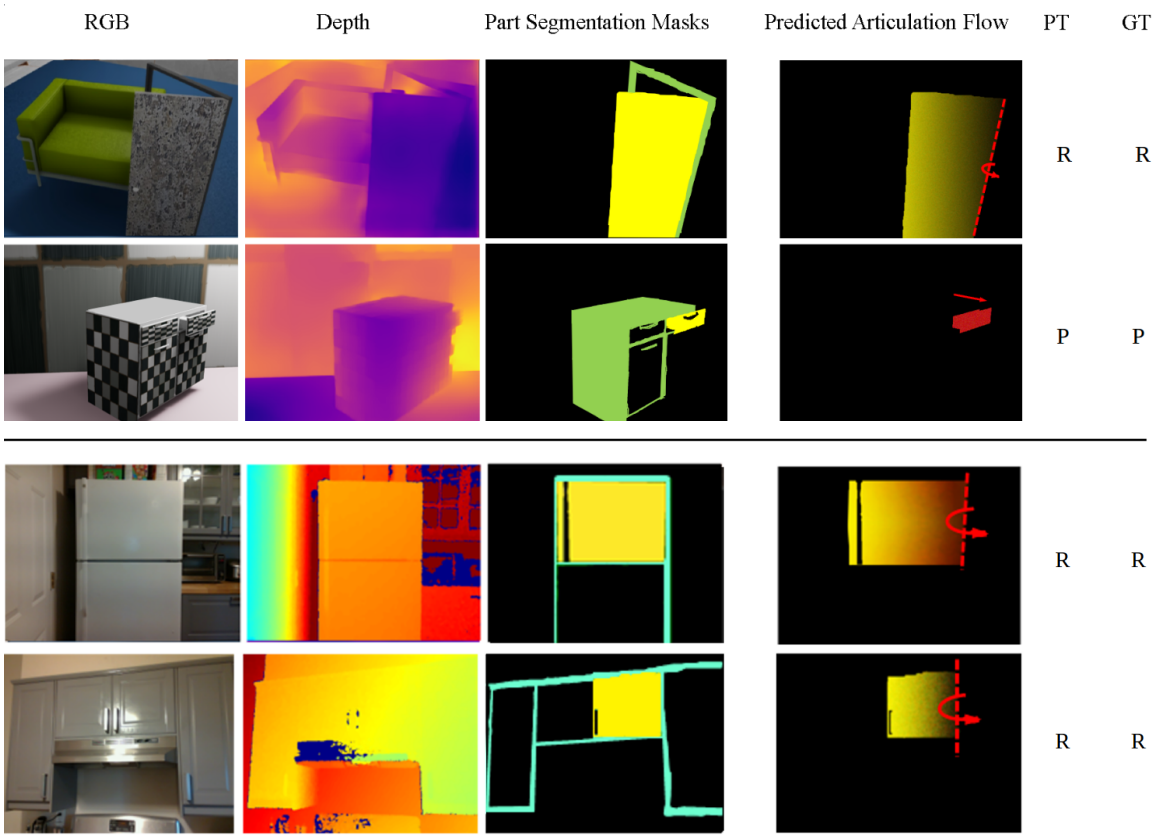



Figure 3.4: Visualization of network inputs and outputs for the Door and Cabinet categories on synthetic data (top two rows) and real-world images (bottom two rows). Additional categories are visualized in supplementary materials. PT means predicted articulation type, and GT ground truth type. Articulation is either revolute (R), prismatic (P), fixed (F), or unconnected (U). The network takes in a stack observation of RGB image, depth image, and two part segmentation masks. The green segmentation mask is the anchor object part and the yellow segmentation mask is the candidate object part. The direction of the motion residual flow is visualized by color gradients, where a prismatic articulation is a single solid color, while a revolute articulation is a gradient towards the axis of rotation.

| | **Object Category** | | | | | | |
|---|---|---|---|---|---|---|---|
| **Metric** | ***Door*** | ***Wind.*** | ***Fauc.*** | ***Dish.*** | ***Frid.*** | ***Cab.*** | ***Avg.*** |
| **AT** | 85.6 | 75.1 | 84.4 | 72.1 | 76.2 | 67.7 | 76.4 |
| **PC** | 97.8 | 91.2 | 94.6 | 95.7 | 95.4 | 92.5 | 94.1 |
| **CA** | 88.7 | 79.6 | 87.5 | 78.6 | 84.2 | 77.5 | 82.5 |
| **B-CA** | 81.5 | 58.7 | 74.3 | 60.1 | 62.7 | 71.2 | 68.0 |

Table 3.2: Accuracy by Category on Test Set. **B-CA** refers to a classification-only baseline method where the output head of the network is performing articulation and connectedness classification directly instead of regressing to motion residual flows. All numbers are shown in percentage.

| | **Object Category** | | | | | |
|---|---|---|---|---|---|---|
| **Metric** | ***Door*** | ***Window*** | ***Faucet*** | ***Dishw.*** | ***Fridge*** | ***Cabinet*** |
| **AT** | 78.8 | 26.5 | 66.2 | 76.3 | 60.3 | 33.5 |
| **PC** | 94.6 | 96.7 | 85.7 | 95.5 | 93.6 | 89.2 |
| **CA** | 82.1 | 47.2 | 73.0 | 81.48 | 73.8 | 58.6 |

Table 3.3: Performance on Novel Object Categories.

Table 3.2 shows the test classification metrics by object category when the network is trained on all object categories. Similar test accuracies are achieved across categories. Fig. 3.5 shows the accuracy of the predicted articulation parameters (location and direction of revolute axes, and direction of prismatic axes).

### 3.5.3 Generalization to Novel Object Categories

We also evaluate how well the proposed method generalizes to novel object categories unseen during training. See Table 3.3 for results. We use a leave-one-out scheme for this evaluation. Specifically, we train six additional models, with the same hyperparameters, such that each model is trained on all categories except a left out category. Each model is then evaluated on the category it was not trained on.

We observe worst performance on windows and cabinets, mediocre performance on faucets and refrigerators, and best performance on doors and dishwashers. The difference between the best and worst performing category is significant, differing by over 50% (windows at 26.5% vs doors at 78.8%). The prediction for windows was worse than chance.

We investigate the abnormally low AT performance of windows in Table 3.3,

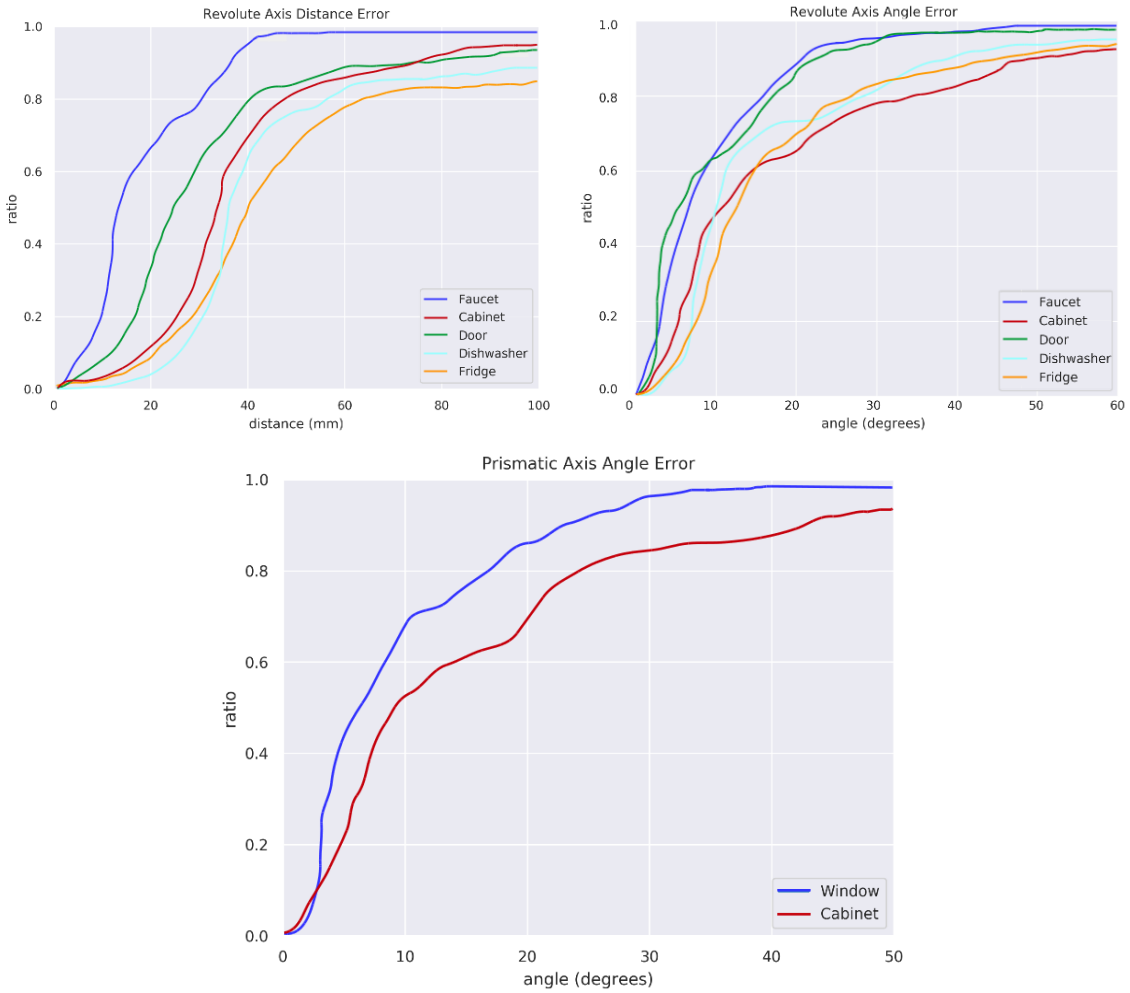



Figure 3.5: Predicted articulation parameter accuracy for revolute axes (top) and prismatic axes (bottom) on test set for all object categories. The revolute axis distance error is the average distance between the points on the ground truth axis to their projections on the predicted axis. The revolute and prismatic angle errors are the angle between the ground truth and the predicted axes. All plots show the percentage of data points that have error below a given threshold.

Figure 3.6: Misleading examples of a window (left) and two doors (right). The shapes of the door panel and window frames, and handles for the window and first door (brown) are similar.

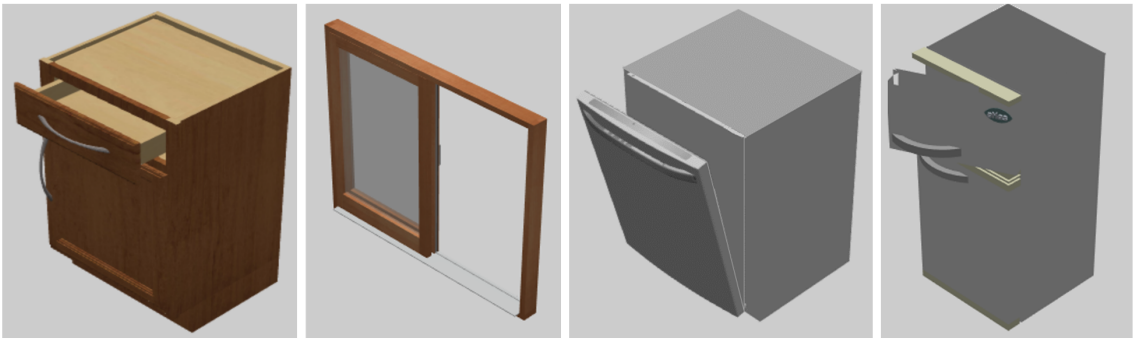


Figure 3.7: Misleading examples of a prismatic cabinet (left) vs other objects (prismatic window, revolute dishwasher, refrigerator). The geometry of the cabinet drawer is different from that of the window, but they both have prismatic joints. By contrast, the handle of the drawer has a similar shape to those of the dishwasher and the refrigerator, even though the latter have revolute joints.

which mainly consists of sliding prismatic joints. High PC across all categories shows that the model identifies connected parts of windows, but it is predicting the wrong articulation types. Analyzing the outputs indicate that most wrong predictions are revolute. We hypothesize two reasons for this: (1) lack of representation of prismatic objects in the training data and (2) misleading revolute objects that share visual similarities with the prismatic windows. Eliminating windows from the training data leaves five categories, of which only one comprises of prismatic-dominant joints.

We also observe similar visual features across windows, doors, and refrigerators, with the latter two as revolute objects. Specifically, they share similar frames, boards

| Train Category | Test Category | | | | | |
|---|---|---|---|---|---|---|
| | ***Door*** | ***Window*** | ***Faucet*** | ***Dishw.*** | ***Fridge*** | ***Cab.*** |
| ***Door*** | 98.7 | 2.5 | 48.7 | 53.7 | 49.6 | 37.3 |
| ***Window*** | 3.2 | 93.2 | 8.9 | 7.8 | 5.6 | 38.2 |
| ***Faucet*** | 33.7 | 8.7 | 88.7 | 4.1 | 4.3 | 17.2 |
| ***Dishw.*** | 34.2 | 12.8 | 3.8 | 98.5 | 52.1 | 34.6 |
| ***Fridge*** | 56.2 | 15.5 | 23.2 | 55.2 | 87.5 | 38.7 |
| ***Cab.*** | 49.6 | 32.5 | 13.5 | 3.4 | 38.7 | 83.2 |

Table 3.4: Performance of Training on One Category. Overall articulation accuracies when trained on one category (each row) and tested on other categories (each column).

and handles, as observed in Fig 3.6.

High visual similarity between windows and revolute objects in the training set may explain the high prediction rate of revolute for windows. The importance of visual similarity is further reflected in cabinets, the other low-performing category. Although cabinets consist of both prismatic and revolute joints, 73% of the errors occur on the prismatic slots. There was only one training category that was prismatic-dominant (windows), and their object parts showed conflicting visual features with the cabinets as seen in Fig 3.7.

### 3.5.4 Generalization from Training on One Category

We formed hypotheses to explain the low transfer for certain unseen categories in the previous section. A lack of or misleading similarity in visual features of the object parts resulted in misclassifications. This could be significant in our understanding of category generalization. What features and representation do we need in our training categories for good generalization? To answer the question and test our hypotheses, we perform an additional ablation study – training networks on just a single category and testing them across all the other five categories. See Table 3.4 for results.

Learning is limited between articulation mechanisms. Different articulation mechanisms (prismatic, revolute) cannot learn well from each other. When trained on a category with only one articulation mechanism, predicting objects with different articulation is low. For instance, a model trained on windows (prismatic) performs below 10% for all but cabinets (the only other category that contains prismatic joints). We also observe relatively high, consistent performance when testing cabinets, regardless of the trained category. Limited learning between articulation mechanisms explain this better performance because only cabinets contain both articulation mechanisms. Therefore, cabinets have partial representation in each trained category. Consequently, cabinets also fail to achieve at least 40% for any trained category, due to

the presence of both articulation mechanisms. Having both articulation mechanisms imply that one of the articulations would be unseen in the train category.

The results also support our hypothesis that misleading similarity in visual features create misclassifications. The lowest transfer occurs between doors and windows. While both performs well on its own category, correct prediction of the other category falls in the $2-3\%$ range. Comparing two sample objects from the two categories show similar handles, boards and frames in Fig. 3.6. However, their ground truth articulation types are different, which explains the low performance. Dishwashers and refrigerators have the highest transfer, and comparing two sample objects show similar handles and physical structure in Fig. 3.7.

### 3.5.5 Real-world Experiments

Despite only being trained using synthetic data, our network also bridges the reality gap when deployed in the real world. To assess how well our model transfers to real-world data, we took 18 RGB-D images of various household items in our homes (comprising of refrigerators, doors, faucets, and cabinets). Part segmentation masks were generated with semi-automatic DEXTR segmentation [125]. Results showed successful transfers, where the model predicts the correct articulation types for 12 of the 18 images. Example visualizations are shown in the bottom rows in Fig 3.4.

## 3.6 Conclusion

We present FormNet, a deep learning approach that predicts articulation mechanisms of object parts from a single image observation without physical interactions and pre-specified categorical kinematic models. Training data is generated with a photorealistic simulator with 6 object categories and 149 objects. Domain randomization over camera poses, lighting, object sizes, textures, and occlusions make the trained network robust to these variations. Experiment results show that our approach generalizes to novel object categories in simulation and can be applied to real-world images without fine-tuning.

## 3.7 Acknowledgments

This work is supported by the NSF Graduate Research Fellowship Program Grant No. DGE 1745016, NSF Grants No. IIS-1956163 and CMMI-1925130, the Office of Naval Research Grant No. N00014-18-1-2775, the NVIDIA NVAIL Program, and the CMU Summer Undergraduate Research Fellowship.

# III

# Structural Sim-to-Real Transfer

# 4 CREST: Causal Feature Selection for Policies

Which state features are important for learning a control policy? Our approach, CREST, addresses this question through causal feature selection. CREST selects the relevant state variables for a given control policy, which apply over the policy's preconditions. The assumptions for CREST are that an internal model (i.e., an approximate task simulation) exists, the context space representation of the internal model facilitates causal interventions, and the (parameterized) control policy and its preconditions are known. Through structure and transfer learning, CREST enables learning of policies that are compact, avoiding unnecessary state features. By construction, policies built using CREST are robust to distribution shifts in irrelevant variables, whereas baseline methods may yield policies with spurious correlations that are brittle. Such distribution shifts could arise from transfer between the internal model and reality, due to variations in dynamics or context distributions not encountered during pretraining with the internal model.

**Application example.** As a concrete example of the types of problems that could be addressed using CREST, consider the case of a home manipulation robot that is tasked with cleaning up fitness weights in a household living room. This task requires the robot to grasp a weight, pick it up, and place it back on a rack. The robot executes this action using a control policy that is a parameterized controller that takes as input various context variables corresponding to factors related to the scene. Fitness weights may often have a coating with a particular, uniform color based on its mass. So, some of the context variables the robot could consider includes the weight color and the geometric dimensions (e.g., length) of the weight. For this example, we assume that the weight's mass is observable or otherwise known to the robot, so mass would also be a variable to consider. With an approach that is agnostic to causal structure, the robot may learn a correlation that, in order to provide the correct controller input, color has an influence on the controller parameters. However, this relationship does not hold in general. If the robot were to attempt this task using weights from a different manufacturer (and thus different color for each weight), the robot would fail to complete the task by incorrectly estimating the controller parameters based on color. However, CREST can provide a solution for determining

which variables should be used as input to the control policy. Assuming an internal model and context space representation are available of the task that admits causal interventions, the robot can use CREST to conduct interventions upon context variables within the internal model to determine which variables affect the policy. This approach would correctly determine that color, while useful for perception, would be irrelevant for control, whereas the robot should instead incorporate mass as an input to the policy.

This example demonstrates the difference between conditioning and intervening upon a variable. For the set of weights from one manufacturer, conditioning upon color may work, but this will not generalize to weights from other manufacturers. Instead, intervening upon color would demonstrate that changing a weight's color, with all other factors held equal, would not affect how the robot should pick and place the weight. On the other hand, an intervention upon mass would correctly expose this feature as the causal variable needed for control, instead of color, which is a rule that generalizes across weights from different manufacturers.

*Sections of the remainder of this chapter first appeared in [31]. Additional information is provided in App. B. This work was presented at the 2021 IEEE International Conference on Robotics and Automation (ICRA 2021). We thank our collaborators on this work: Jialiang (Alan) Zhao, Amrita S. Sawhney, Siddharth Girdhar, and Prof. Oliver Kroemer.*

## 4.1 Summary

*We present CREST, an approach for causal reasoning in simulation to learn the relevant state space for a robot manipulation policy. Our approach conducts interventions using internal models, which are simulations with approximate dynamics and simplified assumptions. These interventions elicit the structure between the state and action spaces, enabling construction of neural network policies with only relevant states as input. These policies are pretrained using the internal model with domain randomization over the relevant states. The policy network weights are then transferred to the target domain (e.g., the real world) for fine tuning. We perform extensive policy transfer experiments in simulation for two representative manipulation tasks: block stacking and crate opening. Our policies are shown to be more robust to domain shifts, more sample efficient to learn, and scale to more complex settings with larger state spaces. We also show improved zero-shot sim-to-real transfer of our policies for the block stacking task.*

## 4.2 Introduction

Real-world environments, such as homes and restaurants, often contain a large number of objects that a robot can manipulate. However, usually only a small set of objects and state variables are actually relevant for performing a given manipulation task. The capability of reasoning about what aspects of the state space are relevant for the task would lead to more efficient learning and greater skill versatility.

Current approaches to learning versatile manipulation skills often utilize sim-to-real transfer learning [126–129], wherein the skill is learned in a simulation and then deployed and fine-tuned (if feasible) on the real robot. Sim-to-real learning is often combined with domain randomization (DR) [43], which involves training the skill on a wide range of task instances in simulation such that the resulting skill is more robust and generalizes across task variations. However, for scenes with distractor objects, the policy still takes the irrelevant state features as inputs. Larger domain shifts in the irrelevant features can therefore still be detrimental to the performance of the skill policy. Rather than relying only on DR, the robot can use model-based reasoning to identify the "structure" of a policy — the interplay between relevant state inputs and control outputs.

In this paper, we propose using causal reasoning to improve sim-to-real transfer through conducting interventions in simulation to determine which state variables are relevant for the successful execution of a task using a given controller. We refer to the resulting algorithm as **CREST**: **C**ausal **R**easoning for **E**fficient **S**tructure **T**ransfer. The relevant state variables from CREST are used to construct policies that encode the causal structure of the manipulation task. These policies are initially trained in simulation using domain randomization over only the states that have explicitly been determined as relevant. Moreover, policies that only use relevant state variables require significantly fewer parameters and, by construction, are robust to distribution shifts in irrelevant state spaces. In this manner, our approach produces lightweight policies that are designed for efficient online adaptation to unforeseen distribution shifts that may occur when bridging the sim-to-real gap. This contrasts with existing sim-to-real approaches that train large policies over enormous state spaces, where the costs for achieving robust zero-shot transfer may be intractable.

Our proposed approach was successfully evaluated on block stacking and crate opening tasks. Although our method is intended for sim-to-real transfer, we primarily conduct our experiments by transferring to task simulations in NVIDIA Isaac Gym [130], a high-fidelity physics simulator, as a proxy for real systems. This is necessary to experimentally evaluate distributions shifts that would be intractable to evaluate (but nonetheless feasible) in practical manipulation scenarios. Additionally, we validate our approach for zero-shot, sim-to-real performance for the block stacking task.

The contributions of our work are as follows.

- We propose CREST, an algorithm that uses causal reasoning in simulation to

identify the relevant input state variables for generalizing manipulation policies.

- We propose two neural network architectures that are constructed using the causal information from CREST.
- We conduct rigorous transfer learning experiments to demonstrate these policies generalize across task scenarios, scale in relevant state complexity, and are robust to distribution shifts.
- We propose CREST as one approach to a broader methodology of structure-based transfer learning from simulation as a new paradigm for sim-to-real robot learning, i.e., *structural* sim-to-real.

## 4.3 Related Works

Our work relates to research areas in robotics and machine learning for structure-based learning, causality, attention, and sim-to-real transfer of higher-level policies.

**Structure-based learning, causality, and attention.** Causal reasoning for structure, transfer, and reinforcement learning is an emerging area of research [131, 132], having been demonstrated for transferring multi-armed bandits policies [133], examining distribution shifts via imitation learning [134], modeling physical interactions from videos [135], and clustering causal factors [136]. Causality reasons about the data generation process with respect to an underlying model [16], which our CREST policies encode. The motivation of our approach to transfer learned causal structure is similar to that described by [137]. In our work, we represent this structure through the policy inputs, and we demonstrate the approach achieves sim-to-real transfer.

Our view of policy structure can be seen as an explicitly encoded form of state space attention, achieved via construction of policies using only relevant inputs. We are primarily concerned with object feature states, which enable significantly smaller neural networks to be constructed under our assumptions. As a comparison, [138] learns implicit attention to task-relevant objects to generalize manipulation skills. This approach requires a few example trajectories to be provided and uses a vision-based state representation, making explicit reasoning about states more challenging.

Similar to [139], our approach can also generalize policies to unforeseen dynamic distribution shifts; ours does so primarily through more efficient fine-tuning.

Our work is also similar in spirit to the work of Nouri and Littman [140] in terms of achieving dimensionality reduction for reinforcement learning. Whereas our work seeks to reduce the dimensionality for the state's possible influence on the task policy under different contexts, they instead demonstrate dimensionality reduction in the action space. Kolter and Ng [141] as well as Parr et al. [142] approximate the value function by learning the relevant basis functions.

**Sim-to-real and higher-level policies.** Our work is a form of simulation-to-reality transfer of controllers [43, 126, 143, 144]. Unlike in the typical sim-to-real paradigm, our policies are not required to transfer zero-shot. Rather, the dimensionality reduction afforded by transferring the relevant state space with CREST enables more efficient online adaptation. Unlike [145], we do not transfer any target samples back to the internal model. Similar to the problem settings of [146–148], our agent predicts the best parameters for a controller.

## 4.4 Problem Formulation

We formulate our problem as a multi-task reinforcement learning problem, wherein a policy $\pi$ is learned to complete a series of tasks $\mathcal{T}_i$. Each task is modeled as a Markov decision process (MDP). The state space $\mathcal{S}$ and action space $\mathcal{A}$ are the same across tasks. However, each task $\mathcal{T}_i$ defines a separate initial state $s_0(c_i)$, transition function $p(s'|s, a, c_i)$, and reward function $r(a, s, c_i)$ that are parameterized by the task's context variables $c_i \in \mathcal{C}$. We assume that the robot has an internal model $p_{\text{int}}(s'|s, a, c_i)$ that approximates the transition function of the target task domain $p(s'|s, a, c_i)$.The context variables capture object parameter variations, such as shapes, appearances, and initial states. We assume that the context variables are always set such that the task is feasible.

To solve the tasks, the robot learns a policy $\pi(a|s, c)$ that is decomposed into two parts [149]: an upper policy $\pi(\theta|c)$ and a lower control policy $\pi_\theta(a|s)$, where $\theta \in \mathbb{R}^d$ are the controller parameters. Transferring the parameters from the internal model to the target domain may require fine-tuning. At the start of each task, the upper policy, which is responsible for generalizing between different task contexts, selects a set of parameters for the control policy to use throughout the task execution. We use multilayer perceptron (MLP) neural networks for the upper policy. In principle, the control policies can take on a variety of parameterized forms, such as motor primitives, planners, waypoint trajectories, or linear feedback controllers, depending on the task. We assume a control policy with known preconditions is available.

Of the context variables $c$ that describe the object variations in the scene, only a subset may be relevant for the policy. We refer to this subset of relevant variables as $\tau \subseteq c$, such that the robot can learn a policy $\pi(\theta|\tau)$ to complete the tasks. Our goal is to determine $\tau$ through causal reasoning with the internal model, yielding policies with fewer input variables as compared to naively using all of $c$.

The set of controller parameters can be divided into individual parameters $[\theta]_j \forall j \in 1, ..., d$, where $[\theta]_j$ indicates the $j$th element of vector $\theta$. Each of these parameters may rely on a different set of relevant variables. We can thus further divide the problem into determining a set of context variables $\tau_j$ for each policy parameter $[\theta]_j$, such that the robot can learn a partitioned task policy $\pi([\theta]_j|\tau_j)\forall j \in 1, ..., d$.

## 4.5 Causal Structure Learning

CREST uses an internal model of the task to learn the relevant context $\tau$ and parameter-specific mappings $\tau_j$ that define the structure of the upper policy (c.f, Sec. 4.6.1).

### 4.5.1 Internal Model for Causal Reasoning

Our approach assumes an *internal model*, an approximate simulation of the task, is available. Analogous to mental models and approximate physics models [150, 151], the internal model facilitates reasoning about the effects of different context variations $\Delta c$ on completing the task with policy parameters $\theta$. Varying the relevant context parameters $\tau$ will affect the execution and outcome of the task in the internal model while irrelevant ones will not.

Given its approximate nature, the solution obtained in the internal model is not necessarily expected to transfer zero-shot to reality. Instead, the internal model provides both an estimate of the policy structure and a task-specific initialization via network pretraining. Intuitively, it is easier to reason about *which* variables are important for a model rather than exactly characterizing the exact model itself. For example, the internal model can capture that the weight of an object affects the required pushing force, but the exact details of frictional interaction may be approximated for the purposes of pretraining the policy. We assume that the internal model approximates the task sufficiently well and includes all context variables $c$ that vary in the target domain. Given the existing challenges in causal representation learning [25], we additionally assume the representation of $c$ is amenable for determining the underlying model for $\theta$ via causal interventions [18].

### 4.5.2 Causal Reasoning to Determine Relevant Contexts

At its core, CREST uses simulation-based causal reasoning to determine the relevant context variables for the policy input. This process is divided into two phases: 1) determining the overall set of relevant variables $\tau$, and 2) determining the relevant variables $\tau_j$ for specific policy parameters $[\theta]_j$.

**Causal interventions to determine relevant variable set.** In this phase, the relevance of a state variable is determined by posing the following question: "If a context variable were different, would the same policy execution still complete the task successfully?" To answer this, we first uniformly sample a context $c_i \sim p(c)$ and solve the resulting task in simulation to acquire the corresponding policy parameters $\theta_i$. In practice, the policy parameters are optimized using Relative Entropy Policy Search [152]. Importantly, the policy is solved for only this specific context $c_i$ (not the general policy).

Given the solved task, we conduct interventions $\Delta c$ to determine if the policy parameters $\theta_i$ remain valid for the new context $c_i+\Delta c$. Interventions $\Delta c$ are conducted to only alter one context variable at a time, i.e. $||\Delta c||_0 = 1$. If the policy subsequently failed, the intervened variable is considered relevant and thus appended to $\tau$. If the policy succeeded, despite the intervention, then the variable is considered irrelevant and is not required for the general policy. The resulting relevant variable set $\tau$ is sufficient for constructing Reduced MLP policies (c.f., Sec. 4.6.1).

**Causal interventions to determine individual policy mappings.** Reducing the set of state variables from the full set $c$ to the relevant set $\tau$ may greatly reduce the size of the policy input. However, for some problems, each of the policy parameters may only depend on a subset of $\tau$. Therefore, in the next phase, the individual mappings from the relevant state variables to the controller parameters are determined by posing the question: "Does altering this relevant context variable require this policy parameter to be changed?"

We begin from the previous phase, where $c_i$ with solution $\theta_i$ is available with interventions $\Delta c$. In this phase, interventions are only applied to relevant context variables $\tau$.

For each new context $c_i + \Delta c$, the task is solved to obtain the resulting policy parameters $\theta_i + \Delta\theta$, starting the optimization from the original parameters $\theta_i$. The optimization will often alter all of the policy parameters, i.e., all elements of $\Delta\theta$ are non-zero, and the magnitude of their changes are not reliable estimators of their importance. Instead, a solution that minimizes the number of non-zero changes, i.e., $\min ||\Delta\theta||_0$, is obtained using search. This search involves setting subsets of elements in $\Delta\theta$ to zero and evaluating if the resulting policy still solves the task with context $c_i + \Delta c$. In our experiments, we used a breadth-first search to find a solution with a minimal set of parameter changes. Once a subset of parameter changes has been found, the context variable intervened on in $\Delta c$ is added to the sets $\tau_j$ of relevant inputs for the policy parameters with non-zero elements in the final $\Delta\theta$. The output of this phase (the parameter-specific variables $\tau_j$) can then be used to learn Partitioned MLP policies (c.f., Sec. 4.6.1).

### 4.5.3 CREST Evaluation

Although we primarily use CREST for manipulation policies, we quantify the accuracy of CREST in a limited, application-agnostic manner. Table 4.1 shows the results of CREST on an environment that replicates the causal structure of hierarchical manipulation policies. This environment is designed so that, given a set of ground truth mappings between context variables and policy parameters, a controller with randomized structure is generated for the agent to "manipulate" the environment to a goal location determined by the causal structure of the problem. These mappings were either linear or (weakly) nonlinear.

Under the assumptions of our controller and the context $c$, CREST excels at

Table 4.1: CREST evaluation for a toy environment on an aggregate ("Agg.") and mapping-specific ("Map.") basis. Accuracy ("Acc.") is whether all relevant states were detected. False positives ("F.P.") are states that were incorrect detected as relevant. 100 trials are used.

| Class | Dim. | Noise | Agg. Acc. | Agg. F.P. | Map. Acc. | Map. F.P. |
|---|---|---|---|---|---|---|
| Linear | 8 | None | 1.00 | 0.00 | 0.98 | 0.19 |
| Nonlinear | 8 | None | 1.00 | 0.00 | 0.97 | 0.23 |
| Linear | 20 | None | 1.00 | 0.00 | 0.99 | 0.53 |
| Nonlinear | 20 | None | 1.00 | 0.00 | 0.97 | 0.24 |
| Linear | 8 | Limited | 1.00 | 0.23 | 0.99 | 0.32 |
| Nonlinear | 8 | Limited | 1.00 | 0.13 | 0.95 | 0.22 |
| Linear | 20 | Limited | 1.00 | 0.22 | 0.98 | 0.50 |
| Nonlinear | 20 | Limited | 1.00 | 0.12 | 0.96 | 0.23 |

determining whether a variable is relevant. This is expected by construction of the underlying mathematics of this environment and represents an expected upper bound. We choose action space dimension of 8 and 20 (larger than our transfer learning experiments, c.f., Sec. 4.7) and select the state space accordingly to permit calculation of ground truth for testing. We also introduce action noise to test the robustness to uncertainty from interventions, leading to more variables detected. Relatively higher false positive rates arise in higher dimensions, but the overall reduction to the relevant set of variables is nonetheless significant.

## 4.6 Policy Learning and Transfer

Given the relevant context from CREST, the structure and transfer learning pipeline of our work begins through 1) constructing the appropriate policy; 2) pre-training using the internal model; and 3) fine-tuning in the target setting.

### 4.6.1 Policy Architectures

Given the full context $c$, the reduced context $\tau$, and the parameter-specific contexts $\tau_j$, the three MLP-based network architectures shown in Fig. 4.1 are constructed and trained using actor-critic approaches [153].

The baseline $\pi(\theta|c)$ uses a standard MLP network. The approach $\pi(\theta|\tau)$ uses a Reduced MLP (RMLP): an MLP network where the inputs are reduced to only the relevant context $\tau$. The approach $\pi([\theta]_j|\tau_j)$ has independent sets of fully connected

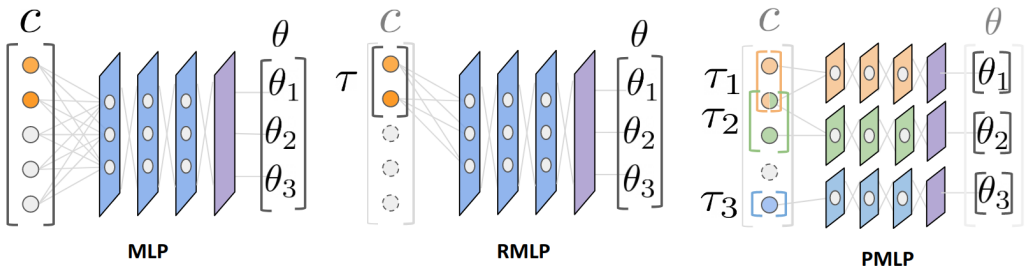


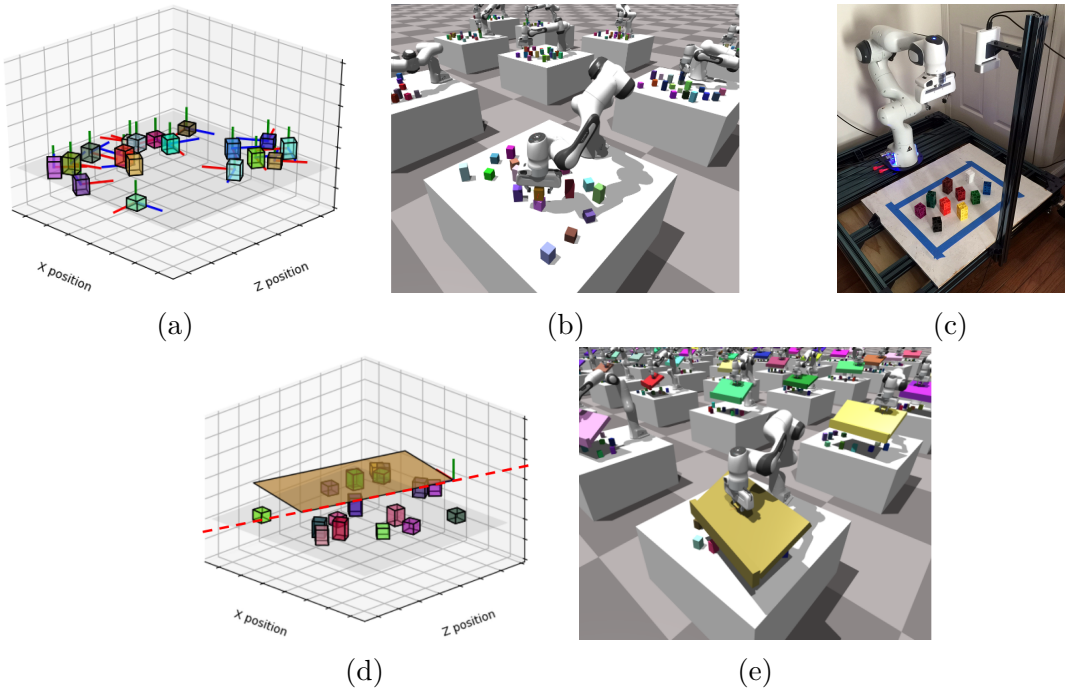

Figure 4.1: A visualization of the different policy types. CREST is used to construct both the Reduced MLP (RMLP) and Partitioned MLP (PMLP). The baseline MLP is also shown for comparison. The relevant states are also used for the critic portion of the networks (only the actor portion is shown). The notation used is $[w_1,...,w_d]$, specifying the hidden units and depth of both the actor and critic.

layers for each policy parameter $\theta_j$, but with potentially overlapping inputs depending on $\tau_j$. This Partitioned MLP (PMLP) network represents the structural causal model [16] for each $\theta_j$ with $\tau_j$ as parent variables.

To provide a fair comparison, we choose the weights for the PMLP according to heuristics and multiply the number of hidden units by the size of the action space to size the RMLP and MLP. We originally sized the PMLP network according to [154] to provide theoretical guarantees regarding function approximation for one-element outputs (i.e., each $\theta_j$), but the resulting network size for the baseline MLP was intractable to train. All neural network weights are randomly initialized per orthogonal initialization [155] using $\sqrt{2}$ and 0.01 for the scale terms of the hidden layers and output layers, respectively. All networks use tanh activations.

### 4.6.2 Network Training and Transfer

For both the internal model and target domain, we train our policies using Proximal Policy Optimization (PPO) [156] with Stable Baselines [157]. First, each network is pretrained with the internal model until the task family is solved. Then, we transfer the network weights to the target domain and evaluate the policy to determine whether the policy transfers zero-shot. Otherwise, fine-tuning is performed. Although freezing network layers has been explored for fine-tuning control policies [158], we permit the entire network to adapt because of the approximate nature of pretraining with the internal model. The learned policy is considered to have solved the task family if it successfully achieves a predefined reward threshold on 50 validation tasks; this evaluation occurs after each policy update.




(a) (b) (c)

(d) (e)

Figure 4.2: Transfer experiments for block stacking and crate opening manipulation tasks. Policies are pretrained in the internal model ((a),(d)) and then transferred to the target domain ((b),(c),(e)). Target domains consist of replications of real systems using a Franka Panda robot, along with a real system for block stacking. Both tasks have distractor objects. For block stacking, only two blocks are necessary to generalize the policy. For crate opening, blocks represent distractor objects (e.g., if the crate were for a chest containing toys).

## 4.7 Experimental Results

We evaluate how CREST can construct policies with greater (target) sample-efficiency and robustness for the robot manipulation tasks of block stacking and crate opening (Fig. 4.2). Our experiments for each task follow the structure and transfer learning pipeline motivating our approach: 1) use CREST to determine the causal structure of the task; 2) construct and pretrain policies with this structure; and 3) transfer and fine-tune these policies in the target domain. The target domains include manipulation tasks in NVIDIA Isaac Gym, enabling rigorous investigation of representative distribution shifts that may occur when deploying sim-to-real policies. For the block stacking task, we additionally leverage a real robot system to assess sim-to-real transfer.

Target simulation and training was conducted using a NVIDIA DGX-1. Pretraining was done using a NVIDIA GeForce RTX 2080. Samples from the block stacking

and crate opening internal models were 400 and 65 times faster to obtain than target simulation samples, respectively. This is consistent with the concept of the internal model as a cheap, approximate simulator, whereas samples from the target domain are costly and therefore desirable to minimize.

Ten independent trials (from internal model pretraining to target fine-tuning) are conducted for each simulation experiment to provide statistically meaningful results given the variance inherent in model-free learning. Statistics are provided in terms of mean and $\pm 1$ standard deviation of policy updates requires to solve the task family. Samples are provided per 1000 ("k-Samples") using a batch size of 512.

The supplementary materials describe further experiment details, such as the setup for the real block stacking target.

### 4.7.1 Block Stacking

The network architectures for the block stacking policies are specified in Table 4.2. Although our policies are nonlinear, note this particular task is linear between $\tau$ and $\theta$.

**Task representation.** In the block stacking task, the context vector

$$c = [\, {c_{B_0}}^{\mathrm{T}}, \ldots, {c_{B_{N_B-1}}}^{\mathrm{T}} \,]^{\mathrm{T}} \in \mathbb{R}^{7N_B}$$

consists of the concatenation of $N_B$ individual block contexts. The context vector for block $b$ is $c_{B_b} = [\, x_b^w,\, z_b^w,\, \psi_b,\, h_b,\, {\boldsymbol{C}_b}^{\mathrm{T}} \,]^{\mathrm{T}} \in \mathbb{R}^7$ In the above equations, $x_b^w$ and $z_b^w$ are the world $x$- and $z$-positions of the blocks. Each block orientation is defined by its rotation angle $\psi_b$ about the block's vertical axis ($y$). The $y$-dimension, or height, of each block is $h_i$. Lastly, the block color $\boldsymbol{C}_b = [\, R_b,\, G_b,\, B_b \,]^{\mathrm{T}}$ is specified via red-green-blue tuple. Note that $y_b^w$ is not part of the context, as the initial scene always consists of blocks on the workspace plane.

The control policy $\pi(a|s, \theta_b)$ for block stacking is a sequential straight-line skill parameterized by $\theta_b = [\, \theta_{\Delta x},\, \theta_{\Delta y},\, \theta_{\Delta z} \,]^{\mathrm{T}} \in \mathbb{R}^3$. This skill specifies waypoints that the robot traverses via impedance control by lifting the source block vertically, moving horizontally, and descending to the desired location. The skill preconditions are that the block is grasped and there are no obstructions to moving the object. The reward function is determined from the source block's position and the goal position upon the target block.

Using the internal model, CREST correctly obtained the relevant context variables:

$$\tau = [x_0^w, x_1^w, h_1, z_0^w, z_1^w]^{\mathrm{T}}$$
$$\tau_{\Delta x} = [x_0^w, x_1^w]^{\mathrm{T}}, \tau_{\Delta y} = [h_1], \tau_{\Delta z} = [z_0^w, z_1^w]^{\mathrm{T}}$$

**Nominal transfer for increasing context size.** We conduct transfer experiments for $N_B = \{2, 6, 10, 14, 18\}$, with each $N_B$ conducted independently. Our approach scales with the relevant part of the context space (Fig. 4.3), bounding the

Table 4.2: Networks used for the block stacking task.

| Network | Parameters | Input Dim. (Total) | Architecture |
|---|---|---|---|
| MLP ($N_B = 2$) | 3298 | 14 (14) | [24, 24, 24] |
| MLP ($N_B = 6$) | 4642 | 42 (42) | [24, 24, 24] |
| MLP ($N_B = 10$) | 5986 | 70 (70) | [24, 24, 24] |
| MLP ($N_B = 14$) | 7330 | 98 (98) | [24, 24, 24] |
| MLP ($N_B = 18$) | 8674 | 126 (126) | [24, 24, 24] |
| RMLP (ours) | 2866 | 5 ($7N_B$) | [24, 24, 24] |
| PMLP (ours) | 754 | 5 ($7N_B$) | [8, 8, 8] x 3 |

sample requirements for the target (as well as the cheaper, internal model). The increasing number of irrelevant dimensions from more blocks are eliminated by CREST prior to conducting domain randomization during pretraining.

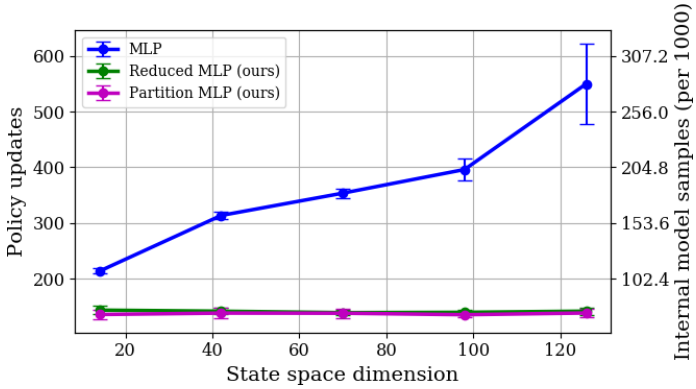
For the case of $N_B = 10$, we trained directly in the target domain without transfer and observed similar results as for pretraining, suggesting the internal model accords well with the target domain. This explains why our approaches exhibit good zero-shot behavior over increasing context dimensions, unlike the baseline whose initial performance degrades as the number of irrelevant contexts increase.

**Distribution shift in irrelevant contexts.** We now evaluate the robustness of the learned block stacking policy to distributions shifts in irrelevant context variables. We conduct two transfer experiments, wherein the policies are pretrained using only half of the color space. In the first case, the target has the same context distribution as the internal model. In the second case, the target has the opposite color space (without overlap). The experimental results (Table 4.3) elucidate how a seemingly inconsequential variable can degrade policy execution through a distribution shift that the robot is not trained to expect. Our approaches generate policies that are robust to these irrelevant domain shifts by construction; as CREST explicitly identifies this dimension as unimportant and excludes it from target learning.

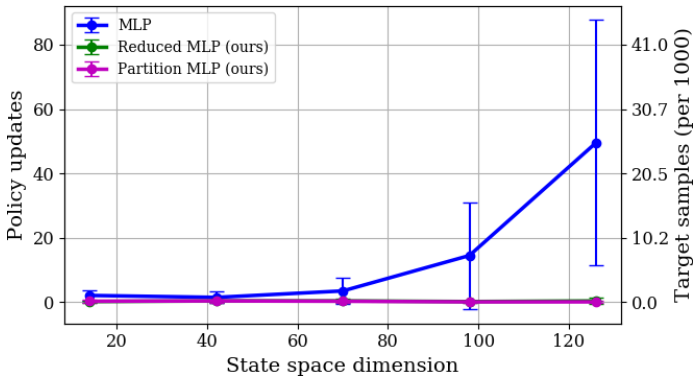
**Sim-to-real policy evaluation.** Lastly, to validate our approach for sim-to-real transfer, we evaluate the zero-shot policy performance on a real robot system that implements the block stacking task with $N_B = 10$ (Fig. 4.2c). As shown in Table 4.4, our policies successfully demonstrate greater zero-shot, sim-to-real transfer as compared to the baseline.

### 4.7.2 Crate Opening

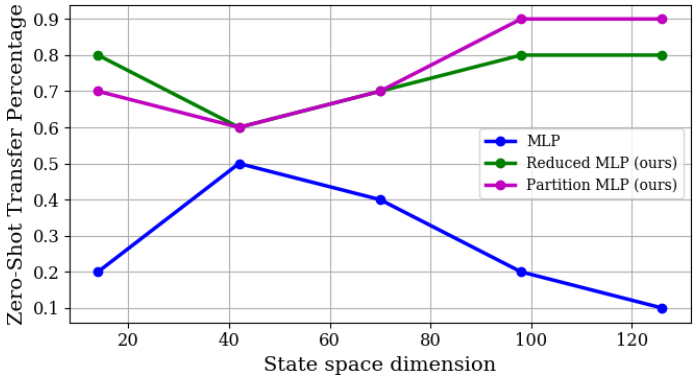
The crate opening experiment is nonlinear between $\tau$ and $\theta$, and the internal model, which is kinematic, presents a greater sim-to-real gap than block stacking.




(a)




(b)




(c)

Figure 4.3: Sample complexity of training a solved block stacking policy based on context dimension for (a) internal model and (b) target setting. c) Zero-shot transfer percentage, wherein the transferred policy needs no further target training to solve the task.

Table 4.3: Transfer results for a distribution shift in 30 context variables (color) that are irrelevant for the block stacking policy.

| Network | MLP | RMLP (ours) | PMLP (ours) |
|---|---|---|---|
| IM Updates (k-Samples) | 237.5 (121.6) ± 11.6 (5.9) | 137.6 (70.5) ± 7.2 (3.7) | 139.2 (71.3) ± 8.2 (4.2) |
| Target Updates (k-Samples), no shift | 1.6 (0.8) ± 1.5 (0.8) | 0.5 (0.3) ± 0.9 (0.5) | 0.0 (0.0) ± 0.0 (0.0) |
| Zero-Shot Transfer no shift | 4 | 7 | 10 |
| Target Updates (k-Samples), shift | 17.3 (8.9) ± 4.8 (2.5) | 1.0 (0.5) ± 1.5 (0.8) | 0.1 (0.1) ± 0.3 (0.2) |
| Zero-Shot Transfer, shift | 0 | 6 | 9 |

Table 4.4: Sim-to-real policy evaluation results for block stacking with $N_B = 10$. The reward threshold for zero-shot transfer is -0.025 (about half of the block width). We also note how often the block was successfully stacked. "GT" is a ground truth policy to illustrate the degree of uncertainty present within the robot perception and control system. Each policy was evaluated 10 times.

| Policy | Reward | Zero-Shot Transfer | Block Stacked |
|---|---|---|---|
| MLP | -0.033 ± 0.012 | 3 | 1 |
| RMLP (ours) | -0.018 ± 0.007 | 9 | 4 |
| PMLP (ours) | -0.014 ± 0.004 | 10 | 6 |
| GT | -0.009 ± 0.003 | 10 | 10 |

Table 4.5: Networks used for the crate opening task.

| Network | Parameters | Input Dim. (Total) | Architecture |
|---|---|---|---|
| MLP | 13496 | 80 (80) | [40, 40, 40] |
| RMLP (ours) | 7576 | 6 (80) | [40, 40, 40] |
| PMLP (ours) | 1152 | 6 (80) | [8, 8, 8] x 5 |
| PMLP-R (ours) | 8032 | 6 (80) | [24, 24, 24] x 5 |

Therefore, we also consider a second partitioned network, PMLP-R, with the same number of weights as the RMLP, to elicit possible influence of network expressivity in this domain due to the structural assumptions of the PMLP. The crate experiment primarily focuses on dynamics and context shifts, rather than varying numbers of objects, so unlike in blocks, the networks (Table 4.5) are the same for all experiments. Beyond our two experiments in nominal transfer and dynamics shift, we also conducted a color shift experiment with similar results as with the blocks experiment.

**Task representation.** For the crate opening task, the context vector is $c = [\, {c_C}^{\mathrm{T}}, {c_{B_0}}^{\mathrm{T}}, \ldots, {c_{B_9}}^{\mathrm{T}} \,]^{\mathrm{T}} \in \mathbb{R}^{80}$. The block context is as defined previously. The crate context is $c_C = [\, {\boldsymbol{p}_C^w}^{\mathrm{T}}, \Phi, x_g^C, z_g^C, \Theta_o, {\boldsymbol{C}_C}^{\mathrm{T}} \,]^{\mathrm{T}} \in \mathbb{R}^{10}$, where $\boldsymbol{p}_C^w = [\, x_C^w, y_C^w, z_C^w \,]^{\mathrm{T}}$ is the position of the crate coordinate frame with respect to the world frame with vertical angle $\Phi$. The crate is always initially closed (horizontal), but the desired goal angle is specified by $\Theta_o$. The robot interacts with the crate via a grasp point specified in the frame of the crate by $x_g^C$ and $z_g^C$ which are orthogonal and parallel to the crate rotational axis, respectively. The color of the crate is $\boldsymbol{C}_C$.

The control policy $\pi(a|s, \theta_a)$ is a robot skill that executes circular arcs emerging from the grasp point with the following parameterization: $\theta_a = [\, {\theta_{\boldsymbol{p}_a^w}}^{\mathrm{T}}, \theta_{\Delta\gamma}, \theta_{\Delta\phi} \,]^{\mathrm{T}} \in \mathbb{R}^5$, where $\theta_{\boldsymbol{p}_a^w} = [\, \theta_{x_a^w}, \theta_{y_a^w}, \theta_{z_a^w} \,]^{\mathrm{T}}$ is the sphere position used to calculate the radius from the grasp point. Then, the arc is traced out $\theta_{\Delta\gamma}$ in azimuth and $\theta_{\Delta\phi}$ in inclination in polar coordinates from the grasp point. The skill preconditions are that the crate is grasped and unobstructed. To learn this policy, the reward function is calculated from the crate angle error and total kinematic error.

In this formulation, CREST determined the following relevant context variables, which are expected based on rigid body articulation kinematics:

$$\tau = [\, x_C^w, y_C^w, z_C^w, \Phi, z_g^C, \Theta_g]^{\mathrm{T}}$$
$$\tau_{x_a^w} = [x_C^w, \Phi, z_g^C]^{\mathrm{T}}, \tau_{y_a^w} = [y_C^w], \tau_{z_a^w} = [z_C^w, \Phi, z_g^C]^{\mathrm{T}},$$
$$\tau_{\Delta\gamma} = [\Phi, \Theta_g]^{\mathrm{T}}, \tau_{\Delta\phi} = [\Phi, \Theta_g]^{\mathrm{T}}$$

**Nominal transfer.** The transfer learning results for the crate opening policy is shown in Table 4.6. Pretraining the model reduced the number of target updates for the non-partitioned networks. However, this was not the case for the partitioned networks, regardless of size. This is likely a result of the discrepancy between the

Table 4.6: Pretraining and transfer results for crate opening policies compared to training directly in target (without transfer).

| Network | IM Updates (k-Samples) | Target Updates (k-Samples), transfer | Target Updates (k-Samples), direct |
|---|---|---|---|
| MLP | 45.0 ± 3.16 (23.04 ± 1.62) | 12.20 ± 1.72 (6.25 ± 0.88) | 38.70 ± 10.99 (19.8 ± 5.63) |
| RMLP (ours) | 32.40 ± 3.67 (16.59 ± 1.88) | 7.0 ± 1.18 (3.58 ± 0.61) | 16.40 ± 4.05 (8.40 ± 2.08) |
| PMLP (ours) | 48.20 ± 13.33 (24.68 ± 6.83) | 14.0 ± 3.74 (7.17 ± 1.92) | 14.20 ± 6.32 (7.27 ± 3.24) |
| PMLP-R (ours) | 51.30 ± 10.82 (26.27 ± 5.54) | 14.3 ± 4.34 (7.32 ± 2.22) | 15.80 ± 3.97 (8.09 ± 2.03) |

Table 4.7: Fine-tuning for crate opening policies with increasing crate stiffness and correspondingly greater transition model difference between the internal model and target task.

| Network | Target Updates (k-Samples), light | Target Updates (k-Samples), nominal | Target Updates (k-Samples), stiff |
|---|---|---|---|
| MLP | 3.90 ± 0.54 (2.00 ± 0.28) | 12.20 ± 1.72 (6.25 ± 0.88) | 27.30 ± 5.51 (13.98 ± 2.82) |
| RMLP (ours) | 3.20 ± 0.60 (1.64 ± 0.31) | 7.00 ± 1.18 (3.58 ± 0.61) | 16.40 ± 5.90 (8.40 ± 3.02) |
| PMLP (ours) | 9.10 ± 1.58 (4.66 ± 0.81) | 14.00 ± 3.74 (7.17 ± 1.92) | 24.00 ± 15.06 (12.29 ± 7.71) |
| PMLP-R (ours) | 9.20 ± 1.54 (4.71 ± 0.79) | 14.30 ± 4.34 (7.32 ± 2.22) | 19.10 ± 4.91 (9.80 ± 2.51) |

internal model and the target domain, which also explains the difference between the policy updates required for pretraining versus training directly in the target. However, the reduction of relevant variables reduces the number of updates required to train directly in the target for both the RMLP and PMLP.

**Dynamics distribution shift.** Unlike the block stacking problem, the modeling gap between the internal model and target setting is sufficiently large that the trained policies incur a significant performance degradation upon first evaluating in the target domain. We investigated this further by transferring the policies to two target settings with different crate stiffness values. To focus on this dynamics shift, no other shifts (e.g., in context space) were induced.

The results in Table 4.7 suggest that increasing the stiffness is sufficient as a proxy for increasing the modeling difference between the internal model and the target. The

optimal parameters for the kinematic case (internal model) are not necessarily the same as the target domain with realistic dynamics of manipulation using impedance control. Therefore, greater modeling differences implies that greater search in policy parameter space is required to converge to parameters that generalize in the target domain.

In all cases, we see that the RMLP network performs best. As a likely consequence of a less expressive network with a larger dynamics gap, the smaller PMLP network demonstrated a significant variance increase in the higher stiffness case than the larger PMLP-R. Overall, our policies are more robust to distribution shifts in model dynamics. However, we note that partitioning imposes structure that may not be optimal for this problem, as the MLP outperformed the partitioned networks in the light and nominal stiffness cases.

## 4.8 Conclusion

The causal reasoning afforded by CREST allows the robot to structure robot manipulation policies with fewer parameters that are more sample efficient and robust to domain shifts than a naive approach that includes all known contexts. Indeed, using causality to reason about the simulation of a task identifies *what* variables are important to generalize a policy, while domain randomized pretraining provides a strong, task-specific prior in terms of *how* they matter. We believe that CREST is one step towards a new paradigm for structural sim-to-real transfer of robot manipulation policies that are sufficiently lightweight to be adapted in-the-field to overcome unforeseen domain shifts.

For future work, we will investigate using precondition learning to relax the assumption that the policy execution is feasible. We will also explore how the robot can learn the internal model used as the causal reasoning engine.

## 4.9 Acknowledgments

We gratefully acknowledge support for this work from the U.S. Office of Naval Research (Grant N00014-18-1-2775), U.S. Army Research Laboratory (Grant W911NF-18-2-0218 as part of the A2I2 Program), and the NVIDIA NVAIL Program.

# 5 SCALE: Causal Learning of Skills

In our preceding work, we demonstrated how the causal features for a low-level control policy could be determined using CREST (Ch. 4) [31]. CREST assumes that the preconditions of this low-level control policy are given. In this chapter, we present SCALE, our approach for causal learning of skills. This work both extends and builds upon CREST by learning multiple distinct skills, which are behaviors defined by control policies and additional characteristics. These skills are also learned with their preconditions, which relaxes the assumptions needed for CREST. Importantly, we make the connection between causal learning for manipulation and data generating processes explicit in this work. The Data Generating Region for each skill, in fact, captures where the underlying data generating process exists in state space, where the process is the skill being used to successfully complete a task.

*Sections of the remainder of this chapter first appeared in [32]. Additional information is provided in App. C. This work was presented at the 7th Annual Conference on Robot Learning (CoRL 2023). We thank our collaborators on this work: Shivam Vats (co-first author), Siddharth Girdhar, and Prof. Oliver Kroemer.*

## 5.1 Summary

*We propose SCALE, an approach for discovering and learning a diverse set of interpretable robot skills from a limited dataset. Rather than learning a single skill which may fail to capture all the modes in the data, we first identify the different modes via causal reasoning and learn a separate skill for each of them. Our main insight is to associate each mode with a unique set of causally relevant context variables that are discovered by performing causal interventions in simulation. This enables data partitioning based on the causal processes that generated the data, and then compressed skills that ignore the irrelevant variables can be trained. We model each robot skill as a Regional Compressed Option, which extends the options framework by associating*

*a causal process and its relevant variables with the option. Modeled as the skill Data Generating Region, each causal process is local in nature and hence valid over only a subset of the context space. We demonstrate our approach for two representative manipulation tasks: block stacking and peg-in-hole insertion under uncertainty. Our experiments show that our approach yields diverse skills that are compact, robust to domain shifts, and suitable for sim-to-real transfer.*

## 5.2 Introduction

We want robots to help and work alongside humans in their homes, kitchens, and restaurants. However, outside of structured environments, robots currently struggle at reliably performing even some of the basic manipulation tasks that humans can do with ease. Why are humans so much better despite the vast diversity of objects and their complex interactions that they potentially need to reason about? First, humans usually know multiple ways to solve a task to be robust to failures and variations in the environment. For example, if a tight jar doesn't open with our bare hands, we may use a piece of cloth to improve our grip. Second, humans excel at selectively attending [159] to only a small part of the environment that is relevant to the task. Selective attention significantly reduces the computational complexity of reasoning and allows us to handle complicated situations.

Prior works in manipulation skill learning have leveraged these two observations separately. Most methods [160–164] learn skills by associating each skill with a sub-goal, where, the sub-goals are hand-designed or learned from demonstrations. Once the sub-goals have been assigned, feature selection [138, 141] and abstraction selection [165, 166] can be used to reduce the complexity of skill learning. However, such approaches are quite sensitive to the sub-goals and struggle to distinguish between different strategies to achieve the same goal. Our main insight is to *associate a skill with not just a sub-goal, but also with the variables that are causally relevant to it.* For example, opening a jar with our bare hands is a skill distinct from opening it with the help of a piece of cloth. Only *hand* and *jar* are relevant to the former, while the latter also relies on the properties of *cloth*. Hence, these two strategies should be represented as two distinct skills even though they achieve the same goal.

Based on this principle, a manipulation task involving $n$ variables can have up to $2^n$ skills, based on variable subsets being causally relevant. This is a very large space to search for skills and not all subsets may correspond to a useful skill. Hence, we propose SCALE (**S**kills from **CA**usal **LE**arning), an efficient approach for robot skill learning through causal feature selection in simulation.[1] Instead of naïvely generating data, in our approach, the robot *interacts* with the simulator by conducting causal interventions. This elicits the causal features for completing a task under different settings, yielding a diverse and compact library of skills. Our approach learns skills

[1]Website: `https://sites.google.com/view/scale-causal-learn-robot-skill`

that are described by physically meaningful properties without spurious variables that would be related to irrelevant objects.

Our contributions of this work are two-fold. First, we introduce SCALE, an algorithm for learning a robot skill library from causal interventions in simulation. Second, we conduct a variety of experiments that demonstrate that SCALE outperforms baseline approaches for two manipulation domains of block stacking and sensorless peg insertion. As a part of these experiments, we also demonstrate sim-to-real transfer of the skills learned by SCALE for block stacking.

## 5.3 Related Work

**Robot skill learning.** Building robots that can solve a wide variety of complex tasks is one of the fundamental problems in robotics. A popular approach is to learn skills parameterized by the task parameters as these can generalize over related tasks. Prior works [167, 168] show such parameterized skills lie on a low dimensional piecewise-smooth manifold in the context space and identify this structure using ISOMAP [169]. For higher-dimensional problems, it becomes infeasible to learn directly in the full context space. One approach is to learn a library of simple parameterized skills which can be composed to solve more complex tasks [170–173]. Recent works [174, 175] propose a differentiable attention mechanism to learn context-specific attention, but these have been evaluated only in relatively small domains. Popular methods for unsupervised skill discovery include graph-based methods [176, 177] that seek to build a graph of skills to cover the task space and information-theoretic methods [164, 178, 179] that seek to maximize the diversity of skills.

**Causality in robotics and reinforcement learning.** Causality is the science of cause and effect [16, 17, 19]. Although the advantages of causal inference and discovery within the biomedical sciences, economics, and genomics have been well-established [20, 28], the integration of causality within machine learning is nascent [25, 37]. In robotics, causality-based approaches are particularly underexplored despite the potential advantages of greater reasoning and learning capabilities [180], particularly through structure and transfer learning [137]. Most similar to our work is that of CREST [31], an algorithm for identifying features for a robot policy through causal interventions. Our algorithm SCALE leverages the work of Lee et al. on CREST for determining the causally relevant variables for each robot skill. Causality has also empowered learning the structure of physical systems from videos [135] and explanations for robot failures [181]. Within reinforcement learning more broadly, causality plays a central role for improving performance through greater structure [182], learning latent factors in dynamics via causal curiosity [183], learning invariant policies [184], and learning a dynamics model that can yield state abstractions [38].

**Intuitive physics.** Please see App. C.2 for a discussion of how SCALE relates

to intuitive physics.

## 5.4 Preliminaries

The robot learns a set of skills $\boldsymbol{\mathcal{K}} = \{\mathcal{K}_1, \ldots, \mathcal{K}_K\}$, where each skill solves a distribution of manipulation tasks. Each task is modeled as a *manipulation MDP* [185], $M \coloneqq (\mathcal{S}, \mathcal{A}, \mathcal{R}, T, \gamma, \tau)$, where $s \in \mathcal{S}$ is the state space, $a \in \mathcal{A}$ is the action space, $\mathcal{R}$ is the reward function, $T$ is the transition function, $\gamma$ is the discount factor, and $\tau$ is additional task information. Tasks are solved if the final reward $R_f > R_S$, where $R_S$ is a solved threshold.

**Options.** Each skill $\mathcal{K}$ is a parameterized option [166, 186]. An option $\mathcal{O} \coloneqq (\pi, \mathcal{I}, \beta)$ is defined using three components: (1) the option control policy $\pi(a|s)$; (2) an initiation set $\mathcal{I} = \{R_f > R_S|s\}$ that specifies where the final reward $R_f$ solves the task when taking option $\mathcal{O}$ in state $s$ using option policy $\pi$; and (3) termination condition $\beta(s)$ that specifies when the option concludes. For the termination condition of this work, our skills execute open-loop with fixed duration.

**Context.** We define the context $c \in \mathcal{C}$ as a set of variables $\mathbf{C}$ that fully specify the manipulation task. The context space $\mathcal{C} \coloneqq \mathcal{S} \times \tau$ generalizes the state space to include geometric and other time-invariant task properties defined by $\tau$. Each skill requires only a subset of the full context, determined via causal feature selection (c.f., Sec. 5.6.1).

**Contextual policies.** We use a hierarchical approach [149] to decompose the option control policy into an *upper-level* policy $\pi_u(\theta|c)$ and a *lower-level* policy $\pi_l(a|s, \theta)$. Given a context $c \in \mathcal{C}$, $\pi_u : c \mapsto \theta$ specifies the parameters $\theta$ for the lower-level policy $\pi_l$. For example, $\pi_l$ could be a Cartesian-space impedance controller, where $\theta$ specifies the sequence of waypoints to be followed by the robot end-effector. For our work, we assume the lower-level controller $\pi_l$ is given, and we learn the upper-level policy $\pi_{uk}$. The lower-level controller is shared across the different skills.

**Compressed Context and Feature Selection.** SCALE learns *compressed* skills that only use causally relevant context variables, as many will be unimportant. For our work, we disregard dimensions of the context space that are not chosen by causal feature selection (c.f., Sec. 5.6.1), leading to a compressed context space $\hat{c}$ that is obtained by selecting dimensions of the full context space that correspond to the relevant variables of interest.

**Causal Reasoning in Simulation.** SCALE leverages a simulator with the key capability of interacting with scenes through context interventions, which enables the causal learning in SCALE. For this reason, we formalize the simulator as a *causal reasoning engine* $\mathcal{W} \coloneqq (\mathfrak{C}_S, T)$, where $\mathfrak{C}_S$ is the scene structural causal model (SCM) and $T$ is the transition model. App. C.3 provides greater discussion of this formalization. This formalism addresses SCALE's assumption that the simulator is capable of answering questions to scene interventions, i.e., constructing new scenes with a

change to one variable to assess if there is a change (c.f., Sec. 5.6.1). These variables are required to be intervenable within the simulator, but not all variables need to be intervenable. For instance, gravity is a simulation variable, but for this work, it is not considered as a candidate for causal reasoning; therefore, it does not need to be intervenable.

## 5.5 Skill Formulation

### 5.5.1 Regional Compressed Option

In our work, we formalize each robot skill $\mathcal{K}$ as a *Regional Compressed Option* (RCO), where $\mathcal{K} := (\pi_k, \text{Pre}, \beta, \mathscr{D})$ and $\pi_k$ is the option control *policy*, Pre is the *precondition*, $\beta$ is the *termination condition*, and $\mathscr{D}$ is the *data generating region* (DGR). In this model, the policy $\pi_k(a|\hat{c}_{\mathbf{A}_k})$ uses compressed context $\hat{c}_{\mathbf{A}_k}$, which is obtained by selecting dimensions of the context space according to the relevant variable set $\mathbf{A}_k \subseteq \mathbf{C}$ (c.f., Sec. 5.6.1). The learned, upper-level policy is $\pi_{uk}(\theta|\hat{c}_{\mathbf{A}_k})$. The precondition $\text{Pre}(c) = P(R_f > R_S|c)$ is a probabilistic initiation set [187].

### 5.5.2 Data Generating Region

Our goal is to learn an upper-level policy $\pi_u : c \to \theta$, i.e., a mapping that generates the correct parameters $\theta$ for solving the task from any initial context $c$. We refer to this unknown mapping as a *data generating process* or *causal process*. Instead of trying to learn this data generating process directly, which may be difficult when many variables are involved, our main insight is to model it as a mixture of multiple causal processes. Each such process is likely to have a smaller set of relevant variables and thus would be easier to learn. For example, consider the task of opening jars, where the jar could be *tight* or *not tight*. We can model the data generating process for this task as a combination of two simpler causal processes: $\mathfrak{C}_1$ which uses only your hand, and $\mathfrak{C}_2$ which uses a piece of cloth along with your hand. However, these causal processes don't hold for all jar opening tasks. $\mathfrak{C}_1$ holds when the jar is *not tight*, while $\mathfrak{C}_2$ holds when the jar is *tight*. Thus, every causal process is valid only in a subset of the context space. We refer to this subspace $\mathscr{D} \subseteq \mathcal{C}$ as the data generating region (DGR) of the causal process. Here, $\mathscr{D}_1 := \{\textit{not tight}\}$ and $\mathscr{D}_2 := \{\textit{tight}\}$. The robot learns a separate skill for every such causal process. Furthermore, each skill should be trained by only using data from inside its DGR; data lying outside the DGR are generated by a different causal process and is hence out-of-distribution. The DGR uses compressed context $\hat{c}_{\mathbf{D}_k}$ obtained from relevant variable set $\mathbf{D}_k \subseteq \mathbf{C}$ (c.f., Sec. 5.6.1).

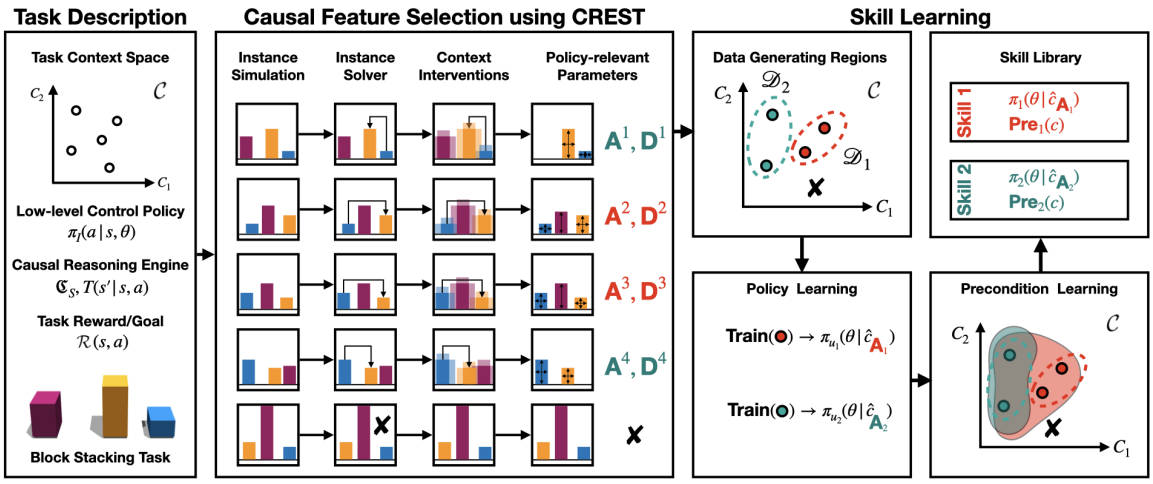



Figure 5.1: The figure shows an overview of the proposed framework applied to a block stacking task. The robot is given a context space, control policy, task simulator, and task reward. The robot samples a set of contexts to create task instances, which it subsequently solves for that instance. The robot then applies interventions on the contexts to identify skill-relevant parameters. Contexts with the same set of policy-relevant parameters come from the same causal model and are hence combined to form data generation regions. Here, we have two causal models: $\mathfrak{C}_1$ with relevant variables from the yellow, blue, and red blocks and DGR $\mathscr{D}_1$; and $\mathfrak{C}_2$ with relevant variables from the yellow and blue blocks and DGR $\mathscr{D}_2$. Each region is then used to learn a separate skill policy with the corresponding set of policy-relevant parameters. For each skill, we finally learn a set of preconditions within the context space to determine where the skill can ultimately be applied. The pairs of policies and preconditions are then combined to create a skill library for completing the given task.

## 5.6 Skill Discovery through Causal Reasoning in Simulation

The SCALE algorithm (Fig. 5.1) comprises two steps: 1) skill dataset generation and 2) skill training. These steps are described in Sec. 5.6.1 and 5.6.2, respectively. Algorithm descriptions are in App. C.5.

### 5.6.1 Batch Data Generation

First, the robot interacts with the simulator $\mathcal{W}$ to collect skill training data. This is done by collecting a batch dataset $\mathcal{D}_B$. The robot samples $n$ random scenes represented by $c_i$ and attempts to determine the lower-level controller parameters $\theta_i$ that solve the specific task. In practice, we use Relative Entropy Policy Search

(REPS) [152], but any suitable planner, trajectory optimizer, or reinforcement learning algorithm would suffice. Unsolved tasks are disregarded and not collected in $\mathcal{D}_B$.

**Causal feature selection.** For successfully solved scenes, the relevant variables for the policy $\mathbf{A}_i$ and DGR $\mathbf{D}_i$ are selected using the CREST algorithm [31]. CREST conducts feature selection through causal interventions. Intuitively, a variable is causal if, for all other variables held equal, interventions upon this variable induce a change in the final obtained reward $R_f$. A spurious variable has no effect on the reward and thus can safely be ignored. To summarize CREST, the process begins by solving a scene, which we refer to as the non-intervened scene. For each context variable, a new value is randomly sampled from a distribution (in the CREST work, this distribution is the context variable's possible values). A scene is constructed with that intervened value, with all other context variables holding the same, non-intervened value. Then, the robot executes the solution to the non-intervened scene in this intervened scene to obtain an intervened reward. This process repeats a given number of times, and a statistical test is assessed to determine how often the intervened rewards differ from the non-intervened reward. If the intervened rewards are frequently no different than the non-intervened reward, the context variable is considered spurious (and causally relevant otherwise).

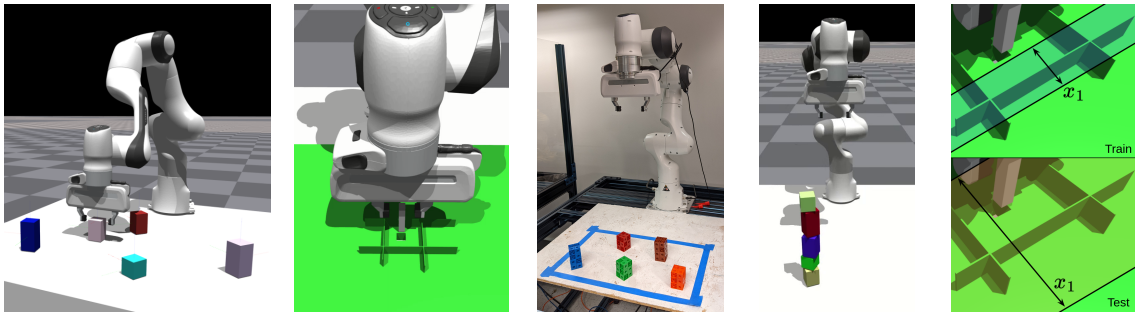
In this work, CREST performs interventions $\mathbf{I}$ over a local (e.g., 10%) fraction of context space $\mathcal{C}$ to yield $\mathbf{A}_i$. Similarly, $\mathbf{D}_i$ is obtained through interventions over the entire space $\mathcal{C}$. Finally, the batch dataset is appended by the dataset point $(c_i, \theta_i, \mathbf{A}_i, \mathbf{D}_i)$. Note that CREST is not a strict requirement of SCALE. In principle, SCALE requires only a determination of which variables are causally relevant, which CREST provides. Other approaches, such as using causal discovery, are also possible. An important consideration of choice of approach is whether the context space is disentangled. In our work, we assume a disentangled context space, and so the variable-by-variable intervention process of CREST (which also assumes disentangled variables) will suffice. If the context space is entangled, then causal disentanglement approaches could first be used.

**Splitting batch data into skill data.** After dataset collection, batch dataset $\mathcal{D}_B$ is split into different skill datasets according to the relevent variable sets. In this work, we assign highly occurring batch data that contain the same relevant variables $\mathbf{A}$ into the same skill dataset $\mathcal{D}_k$, while also taking the union over all associated $\mathbf{D}$. This assumption may not always hold, but is sufficient for the tasks we examine in this work. More sophisticated ways of splitting the batch dataset is left for future work.

### 5.6.2 Skill Training

The second phase of SCALE trains each skill $\mathcal{K}_k$ using $\mathcal{D}_k$. Each skill has relevant variable sets $\mathbf{A}_k$ and $\mathbf{D}_k$ with task solution datapoints $(c, \theta)$.

**DGR.** The DGR $\mathscr{D}$ is first trained on $\hat{c}_{\mathbf{D}_k}$ using $\mathbf{D}_k$. For this work, we use a one-




(a) (b) (c) (d) (e)

Figure 5.2: SCALE discovers skills for the Franka Emika Panda robot using causal learning in simulation for two manipulation tasks: (a) block stacking and (b) peg-in-hole insertion. In addition to skill learning experiments, we also show how SCALE can yield skills (c) for sim-to-real transfer (App. C.9); (d) for generalization in downstream tasks, such as stacking a block tower (App. C.10); and (e) for robustness to task domain shifts (App. C.12).

class SVM to model the DGR, but in principle, any one-class classification algorithm would suffice.

**Policy.** The policy is trained next. The skill dataset contexts are filtered through the DGR $\mathscr{D}$ to obtain inliers $c^+$ for policy training data. This ensures policy training data are consistent with the underlying causal process. Then, the policy $\pi_{uk}(\theta|\hat{c}_{\mathbf{A}_k})$ is trained using $\hat{c}^+_{\mathbf{A}_k}$ (using $\mathbf{A}_k$) and the corresponding parameters $\theta^+$. For our work, policies are learned using regression, but reinforcement learning could also be used [31]. With $\pi_{uk}$ learned, the final skill policy $\pi_k(a|\hat{c}_{\mathbf{A}_k})$ is determined.

**Preconditions.** The preconditions Pre are learned last through policy evaluation. Using the simulator $\mathcal{W}$, contexts $c$ are re-sampled and evaluated with policy $\pi_k$ to obtain rewards $R_f$. This evaluation data $(c, R_f)$ is used to train a precondition classifier to obtain Pre. For our work, we use a nonlinear SVM classifier that has probability estimates.

## 5.7 Experimental Results

We conduct skill learning experiments with SCALE for block stacking and peg-in-hole insertion tasks with the Franka Emika Panda robot (Fig. 5.2). Both tasks are emblematic of high-precision control that is desirable in many industrial applications [188]. We conduct our experiments in NVIDIA IsaacGym [130, 189], a high-fidelity physics simulator that also serves as our causal reasoning engine $\mathcal{W}$. We use a custom library that implements the scene SCM $\mathfrak{C}_S$ to facilitate scene creation and interventions. The forward simulation of physics provides the transition model $T$.

**Baselines.** We compare SCALE to baseline approaches with monolithic policies (without any skills) for either the full-dimensional context space ("monopolicy") or a reduced context space obtained by using the most commonly occurring CREST result ("crest-monopolicy"). The CREST monopolicy represents naïvely using CREST, ignoring that CREST provides locally different results within the underlying data (a property that SCALE leverages).

### 5.7.1 Block Stacking

**Task representation.** In the block stacking task, the robot starts with a source block ($B_1$) grasped, and it learns to place it on top of a target block ($B_2$). To do this, the robot uses a controller $\pi_l$ that defines the trajectory for the robot end-effector to traverse via impedance control. This trajectory is parameterized by $\theta_b = [\,\theta_{\Delta x},\, \theta_{\Delta y},\, \theta_{\Delta z_u}\, \theta_{\Delta z_d}\,]^{\mathrm{T}} \in \mathbb{R}^4$, which specify waypoints the robot follows sequentially. Specifically, these parameters characterize a trajectory where the robot lifts the source block vertically, moves horizontally, descends vertically, and releases the block.

For this task, the context variables $\mathbf{C}_B$ are $\{\mathbf{C}_{B_1}, \ldots, \mathbf{C}_{B_{N_B}}, h_\pi\}$, which is the union of context variables for each of $N_B = 5$ blocks plus the table height $h_\pi$ upon which the blocks are placed. The context variables for each block $b$ are $\{x_b^w, y_b^w, \psi_b, h_b, R_b, G_b, B_b\}$, yielding a 36-dimensional context space for this problem. Here, $x_b^w$ and $y_b^w$ are the world $x$- and $y$-positions of the block, and the block's orientation is represented by a rotation angle $\psi_b$ around the block's vertical axis ($z$). The $z$-dimension (height) of the block is $h_i$. Additional experimental details are available in App. C.8.

**Skill learning results: variable selection.** From a batch dataset of 585 samples, SCALE found the skill library $\boldsymbol{\mathcal{K}}_{blocks} = \{\mathcal{K}_1, \mathcal{K}_2, \mathcal{K}_3\}$ that is shown in Tab. 5.1 that were learned using 340 samples of the dataset. These 340 samples were selected for being the most commonly occurring within the dataset, based on a heuristic threshold. Even though there are five blocks and 36 possible variables, the skills generally consisted of a much smaller subset of variables, relating to the geometry of the source and target blocks. Note that $\mathcal{K}_2$'s relevant variables for the policy, $\mathbf{A}_{\mathcal{K}_2}$, are consistent with earlier work by Lee et al. [31] for this domain. This is generally considered to be the "ground truth" variable result for unobstructed block motion in this case. Skill $\mathcal{K}_1$ could be seen as a version of $\mathcal{K}_2$ when $h_2$ is not needed. Rarely, the source block's rotation $\psi_1$ become important (e.g., the source block's final pose was not fully stable when stacked on the target block), and thus a skill emerges with this variable ($\mathcal{K}_3$). Variables for neither block color nor table height are observed as expected.

**Skill learning results: task evaluation.** We evaluate the skill library $\boldsymbol{\mathcal{K}}_{blocks}$ over the entire task distribution and show the results in Tab. 5.2. That is, for each context sample, the robot evaluates each skill's precondition and selects the skill

Table 5.1: Skills $\boldsymbol{\mathcal{K}}_{blocks}$ that were discovered for the block stacking task. **A** and **D** are the variables used for the skill's policy and DGR, respectively. Data is the quantity of data used for each skill (from a batch dataset of 585 samples, 340 samples were used to train skills). Tsk. Sv. %, shown for both scale-lin and scale-nonlin, is the rate of task solves over the entire context space using only that skill.

| Skill | | Data | Tsk. Sv. %, Lin | Tsk. Sv. %, N.L. |
|---|---|---|---|---|
| $\mathcal{K}_1$ | **A**: $\{x_1^w, y_1^w, x_2^w, y_2^w\}$<br>**D**: $\{x_1^w, y_1^w, x_2^w, y_2^w, h_2\}$ | 53 (9.06%) | 65.36% (200) | 18.36% (56) |
| $\mathcal{K}_2$ | **A**: $\{x_1^w, y_1^w, x_2^w, y_2^w, h_2\}$<br>**D**: $\{x_1^w, y_1^w, h_1, x_2^w, y_2^w, h_2\}$ | 272 (46.50%) | 78.76% (241) | 55.88% (171) |
| $\mathcal{K}_3$ | **A**: $\{x_1^w, y_1^w, \psi_1, x_2^w, y_2^w, h_2\}$<br>**D**: $\{x_1^w, y_1^w, \psi_1, x_2^w, y_2^w, h_2\}$ | 15 (2.56%) | 34.31% (105) | 1.31% (4) |

with the highest probability of success. The suffix "-all" denotes that the entire batch dataset is used for the approach. For both function classes, SCALE yields an approach that outperforms full-dimensional policies and is generally comparable to CREST-reduced policies. However, the CREST-reduced policies only learn one approach to solving the task, whereas SCALE learns three. The overall best performing approach was scale-lin (90.49%) with similar performance to the CREST baseline. Performance across all nonlinear approaches was generally lower. App. C.8 details the SCALE skill selection and further ablations.

**Sim-to-real experiment.** We transfer the skills learned by SCALE and our baselines to a real Franka Panda robot without any fine-tuning. As discussed in App. C.9, SCALE outperforms the baselines.

### 5.7.2 Sensorless Peg-in-Hole Insertion

Our second domain is peg-in-hole insertion under sensing uncertainty. It requires the robot to insert a cuboidal peg of cross-section 1 cm $\times$ 1 cm into a cuboidal hole of cross-section 1.3 cm $\times$ 1.3 cm. The robot gets a noisy initial position of the hole with the noise sampled from a Gaussian distribution $\mathcal{N}(0, 0.3^2\ \text{cm}^2)$. No further sensory observations are available. Due to this uncertainty, a naïve strategy of directly trying to push the peg down at the observed location of the hole achieves a success rate of only 34%. To address this, the robot should take *uncertainty reducing* [190] actions by initiating contact with the environment (e.g., a fixture next to the hole). Our goal in this experiment is to learn such skills autonomously.

**Task representation.** Each assembly task has 4 axis-aligned cuboidal fixtures (i.e., walls) of fixed dimensions around the hole. The 8-dimensional context variables $\mathbf{C}_P$ are $\{x_1, y_1, \ldots, x_4, y_4\}$, containing the $(x, y)$ coordinates of these fixtures with respect to the hole. The positions are different in every task, but it is always possible for the robot to localize against any of the walls to complete the task. We use a

Table 5.2: Task evaluation results for using the skill library $\boldsymbol{\mathcal{K}}_{blocks}$ for the block stacking task. Ctrl. is the approach control (skills or one monolithic policy). Fn. Cl. is the approach's function class. Linear approaches use Bayesian ridge regression, whereas nonlinear methods consist of a multilayer perceptron with a 16x16x16 architecture using ReLU activations. Task Solve % is the rate of task solves over the entire context space using the approach. Methods within ±2% (the stochasticity of the simulator) of the best approach are bold. $|\mathbf{A}|$ is the quantity of input variables used for the approach's policy. Data is the amount of training data used for the approach. A ground truth policy is also shown, using all context variables and additional domain knowledge.

| Approach | Ctrl. | Fn. Cl. | Task Solve % | $\lvert\mathbf{A}\rvert$ | Data |
|---|---|---|---|---|---|
| scale-lin (ours) | 3 skills | Linear | **90.49%** (276) | 4/5/6 | 340 |
| monopolicy-lin-all | 1 policy | Linear | 85.95% (263) | 36 | 585 |
| crest-monopolicy-lin-all | 1 policy | Linear | **89.87%** (275) | 5 | 585 |
| scale-nonlin (ours) | 3 skills | Nonlinear | **63.40%** (194) | 4/5/6 | 340 |
| monopolicy-nonlin-all | 1 policy | Nonlinear | 10.13% (31) | 36 | 585 |
| crest-monopolicy-nonlin-all | 1 policy | Nonlinear | **60.78%** (186) | 5 | 585 |
| ground-truth-policy | 1 policy | Nonlinear | 95.75% (293) | * | – |

6-parameter policy space: three $(\Delta x, \Delta y, \Delta z)$ actions executed in sequence in the robot's end-effector frame. In every policy, $\Delta z$'s are designed to move the peg down while $\Delta x$ and $\Delta y$ are parameters that are learnt using RL. Additional experimental details are available in App. C.11.

**Skill learning results: variable selection.** Table 5.3 enumerates the skills $\boldsymbol{\mathcal{K}}_{peg} = \{\mathcal{K}_1, \ldots, \mathcal{K}_5\}$ discovered by SCALE. Skills $\mathcal{K}_{2-5}$ localize against one of the 4 walls. For each such skill, only the wall being used for localization is relevant to the skill and the other walls can be ignored. Consequently, the set of relevant variables for these skills contains only the distance to the wall that it localizes against. For the linear case, all of these skills except for $\mathcal{K}_3$ have high success rate, whereas success rates are slightly lower for the nonlinear approach. Interestingly, SCALE also discovers a skill $\mathcal{K}_1$ that has an empty set of relevant variables. For more discussion of this skill, see App. C.11.

**Skill learning results: task evaluation.** Table 5.4 presents the task evaluation of the skill library $\boldsymbol{\mathcal{K}}_{peg}$ for 256 randomly sampled tasks. For both linear and nonlinear cases, SCALE outperforms both baselines. The low success of monopolicy-nonlin-all is likely due to insufficient data owing to a larger network. The most common CREST result was variable $x_4$ (21.90%), so this was used for the CREST baselines. However, it only localizes against one wall. The improvement of SCALE over the CREST baselines implies SCALE skills benefit from the DGRs through greater quality training data, whereas the CREST approaches use the entire dataset despite most samples having a differing CREST result than $x_4$. For details of SCALE skill selection

Table 5.3: Skills $\boldsymbol{\mathcal{K}}_{peg}$ that were discovered for the peg-in-hole insertion task. Columns are the same as in Tab. 5.1, except Data represents which 168 samples were used to train skills (from a batch dataset of 210 samples).

| Skill | **A** | **D** | Data | Task Solve %, Lin | Task Solve %, Nonlin. |
|---|---|---|---|---|---|
| $\mathcal{K}_1$ | $\{\}$ | $\{x_1, y_2, y_3, x_4\}$ | 56 (26.67%) | 64.84% (166) | 61.72% (158) |
| $\mathcal{K}_2$ | $\{x_4\}$ | $\{x_4\}$ | 25 (11.90%) | 97.66% (250) | 84.38% (216) |
| $\mathcal{K}_3$ | $\{x_1\}$ | $\{x_1\}$ | 27 (12.86%) | 44.53% (114) | 84.77% (217) |
| $\mathcal{K}_4$ | $\{y_3\}$ | $\{y_3\}$ | 28 (13.33%) | 94.14% (241) | 82.81% (212) |
| $\mathcal{K}_5$ | $\{y_2\}$ | $\{y_2\}$ | 32 (15.24%) | 98.44% (252) | 79.69% (204) |

Table 5.4: Task evaluation results for using the skill library $\boldsymbol{\mathcal{K}}_{peg}$ for peg insertion (columns in Tab. 5.2).

| Approach | Ctrl. | Fn. Cl. | Task Solve % | $\lvert\mathbf{A}\rvert$ | Data |
|---|---|---|---|---|---|
| scale-lin (ours) | 5 skills | Linear | **96.48%** (247) | 0/1/1/1/1 | 168 |
| monopolicy-lin-all | 1 policy | Linear | 62.50% (160) | 8 | 210 |
| crest-monopolicy-lin-all | 1 policy | Linear | 62.89% (161) | 1 | 210 |
| scale-nonlin (ours) | 5 skills | Nonlinear | **88.67%** (227) | 0/1/1/1/1 | 168 |
| monopolicy-nonlin-all | 1 policy | Nonlinear | 12.89% (33) | 8 | 210 |
| crest-monopolicy-nonlin-all | 1 policy | Nonlinear | 55.47% (142) | 1 | 210 |

and further ablations, see App. C.11.

**Domain shift experiment.** To evaluate the out-of-distribution generalization capabilities of SCALE, we evaluate the skills on a test distribution that is significantly harder than the training distribution. All approaches see a degradation in performance, but ours is more robust. See App. C.12 for details.

## 5.8 Conclusion

We present SCALE, an approach for discovery of compact, diverse robot manipulation skills from causal interventions in simulation. These skills arise from the skill DGR: a region that captures the underlying data generating process. We demonstrate the advantages of skill libraries discovered with SCALE for two simulation domains as well as on a real robot system.

**Limitations and future work.** SCALE assumes the robot has access to a causal reasoning engine. We provide this via simulation and scene structural causal models, but these models could be learned via causal discovery. SCALE primarily learns from batch dataset collection; active learning of skills would reveal useful behaviors that are statistically uncommon in the batch setting. Lastly, SCALE assumes that the context variables are defined, intervenable, and disentangled. For tasks and domains where these assumptions do not currently hold, future work in adjacent fields may ultimately

provide a path forward. Specifically, methods from causal representation learning [25] — learning high-level intervenable variables from low-level observations — may hold promise for learning a causal, disentangled state representation that SCALE can use in cases where a suitable representation cannot be provided by hand.

## 5.9 Acknowledgments

We gratefully acknowledge support from the National Science Foundation (Grant No. CMMI-1925130), U.S. Office of Naval Research (Grant No. N00014-18-1-2775), U.S. Army Research Laboratory (Grant No. W911NF-18-2-0218 as part of the A2I2 Program), and the NVIDIA NVAIL Program. We also gratefully thank our reviewers, whose helpful comments strengthened this work.

# IV

# Causality and Dynamical Systems

# 6 Learning By Doing: Controlling a Dynamical System using Causality, Control, and Reinforcement Learning

The implications of reasoning and learning with respect to data generating processes for control extend more broadly than robot manipulation. For example, the control of a chemical process may be more effective if the underlying interaction structure of the chemical reactions can be learned and leveraged. Indeed, the principles of causality can be insightful for control of not only robots and chemical reactions, but *dynamical systems* in general. Controlling dynamical systems is an objective for not only the field of causality (interventions), but also for the fields of control theory (controllers) and reinforcement learning (policies). This work, the Learning By Doing competition, investigates this intersection more deeply.

*Sections of the remainder of this chapter first appeared in [33]. Additional information is provided in App. D. This work was first presented at the Thirty-Fifth Conference on Neural Information Processing Systems (NeurIPS 2021) Competition Track. It also appeared in the Proceedings of Machine Learning Research (PMLR), Volume 176: NeurIPS 2021 Competitions and Demonstrations Track. We thank our collaborators on this work: Sebastian Weichwald (University of Copenhagen), Søren Wengel Mogensen (Lund University), Dominik Baumann (Uppsala University), Prof. Oliver Kroemer, Prof. Isabelle Guyon (Université Paris-Saclay, ChaLearn), Prof. Sebastian Trimpe (RWTH Aachen University), Prof. Jonas Peters (University of Copenhagen), and Prof. Niklas Pfister (University of Copenhagen). We also thank all of the participants of this competition.*

## 6.1 Summary

*Questions in causality, control, and reinforcement learning go beyond the classical machine learning task of prediction under i.i.d. observations. Instead, these fields consider the problem of learning how to actively perturb a system to achieve a certain effect on a response variable. Arguably, they have complementary views on the problem: In control, one usually aims to first identify the system by excitation strategies to then apply model-based design techniques to control the system. In (non-model-based) reinforcement learning, one directly optimizes a reward. In causality, one focus is on identifiability of causal structure. We believe that combining the different views might create synergies and this competition is meant as a first step toward such synergies. The participants had access to observational and (offline) interventional data generated by dynamical systems. Track CHEM considers an open-loop problem in which a single impulse at the beginning of the dynamics can be set, while Track ROBO considers a closed-loop problem in which control variables can be set at each time step. The goal in both tracks is to infer controls that drive the system to a desired state. Code is open-sourced to reproduce the winning solutions of the competition and to facilitate trying out new methods on the competition tasks:* `https://github.com/LearningByDoingCompetition/learningbydoing-comp`

## 6.2 Introduction

Modeling actively performed changes in an observed system is an important goal that has appeared in various versions and settings in statistics, engineering, and computer science. Each community has developed their own terminology and methods to tackle the specific applications relevant to their disciplines, leading to the emergence of causality, control theory, and reinforcement learning (RL). Each of these fields brings a different perspective to modeling system changes and our goal of the *Learning by Doing* NeurIPS 2021 competition was to bring together researchers from each of these fields to work on the same set of tasks.

We decided to focus on dynamical systems as these appear in all three fields. To offer a sufficiently diverse set of problems, we provided two competition tracks: Track CHEM and Track ROBO. Track CHEM considers the open-loop problem of choosing a single impulse that can be set at the beginning of a chemical reaction with the goal of reaching a specific concentration of a target reactant. Track ROBO considers the closed-loop problem of continuously providing inputs to a robot that guides the tip of the robot to move along a target trajectory. In both cases, participants were given a recorded data set from the systems and needed to use this data to learn how to optimally interact with the system in new settings.

In Section 6.3, we briefly introduce the three fields, provide the relevant terminology, and discuss how each field models exogenous changes to a system. In Section 6.4,

we introduce Track CHEM and in Section 6.5 Track ROBO. In both aforementioned sections, we also point out the challenges the tasks pose and how they relate to each field. We conclude in Section 6.6 with some of the lessons learnt throughout the competition.

The competition website learningbydoingcompetition.github.io provides tutorials, results, and presentations of some of the competing teams. We provide open-source code at github.com/LearningByDoingCompetition/learningbydoing-comp to reproduce the winning solutions and to allow the application of new methods to the competition tasks.

## 6.3 Causality, Control, and Reinforcement Learning

To foster cross-pollination, we begin by introducing causality, control, and RL and describe how each framework models changes to a system.

**Causality** In classical statistics, a multivariate stochastic system is thought of as describing a single observational distribution. In contrast, a causal system[1] describes a set of distributions that models system behavior not only under passive observation, but also under interventions [16, 19, 20]. To make this more precise, assume we observe a response variable $Y$ and a set of predictors $X = (X_1, \ldots, X_p)$ and wish to model how $Y$ is affected by interventions on the predictors $X$. We now assume that there exists a subset $\mathrm{PA} \subseteq \{1, \ldots, p\}$ of the predictors, called the parents of $Y$, that determine the value of the response $Y$ via the following fixed functional form

$$Y = f(X_{\mathrm{PA}}, \epsilon), \tag{6.1}$$

where $\epsilon$ is a noise variable. This equation is understood to be functional in the sense that the expected value of $Y$ given that $X_{\mathrm{PA}}$ was set to a fixed value $x_{\mathrm{PA}}$ (denoted by $\mathbb{E}[Y \mid \mathrm{do}(X = x_{\mathrm{PA}})]$) is given by $\mathbb{E}[f(x_{\mathrm{PA}}, \epsilon)]$. Such structural equations are the building blocks of structural causal models (SCMs) [16, 17], which is an important class of causal models. Usually, causal models assume some version of stability of mechanisms under interventions [191, 192]. In (6.1) this corresponds to assuming that $f$ remains fixed under any intervention on $X$. Much research in causality investigates under which conditions the function $f$ and the set of parent variables $X_{\mathrm{PA}}$ can be identified from data and how to do so data-efficiently. Causal models also exist for dynamical systems, for example, a multivariate process $Z(t) = (Z_1(t), \ldots, Z_p(t))$ can be modeled by differential equations of the form $dZ_j(t) = F(Z(t))_j$ for each

[1] Here, the notion of a causal system differs from what is usually called a causal system in the systems and control literature where it refers to systems in which outputs only depend on past and current inputs.

component. Interventions then correspond to modifying parts of these equations, for example, by fixing one of the coordinate processes at a certain value over a given time interval (see Peters et al. [193] for an overview). Causal models induce a *graph* over the coordinate processes, which is useful for visualizing a multivariate causal system. In such a graph, each node represents a coordinate process, $Z_i$. We include a *directed edge*, $Z_i \to Z_j$, $i \neq j$, if the right-hand side of $dZ_j(t) = F(Z(t))_j$ is not constant in $Z_i$ and in this case we say that $Z_i$ is a *(causal) parent* of $Z_j$ and that $Z_j$ is a *child* of $Z_i$.

**Control** In automatic control, one assumes that a target process $y : [0, T] \to \mathbb{R}^p$ is generated by a dynamical system. A common setup is a continuous-time state-space model, given by

$$\begin{aligned} \dot{x}(t) &= F(x(t), u(t))f \\ y(t) &= H(x(t), u(t)), \end{aligned} \tag{6.2}$$

where $x(t)$ is the time-dependent *state* of the systems (which could include $y(t)$ itself), $\dot{x}(t)$ the derivative of $x$ with respect to time, and $u(t)$ is the *control input*. *Control design* is the task of constructing a *controller*, that is, a map from measurements $y(t)$ to control inputs $u(t)$. The dynamical system (6.2), sometimes called a *plant*, is often nonlinear in the state $x(t)$, and linear in the control input $u(t)$. When both the inputs $x(t)$ and $u(t)$ and the target process are vector-valued, one also uses the term *multi-input multi-output* (MIMO) system.

A typical approach of control engineering for obtaining a controller may consist of the following steps: (1) Use problem insight to select a useful (often parametric) model class (such as linear/nonlinear, auto-regressive, continuous/discrete-time); (2) Fit the parameters of the model (that have not been set yet); (3) Use the model in a control design method to obtain a controller (for example, by minimizing a given loss function); (4) Test the controller on the system or in simulation; (5) Possibly repeat the cycle if results are not satisfactory.

Control design traditionally builds model classes from first principles, for example, laws in chemistry or physics, even though not all parameters may be known. The task of fitting the model from input-output data is known as *system identification* (see, for example, Ljung [194]). Identifying a system and obtaining a controller is a well-understood problem for linear dynamical systems and we refer the reader to textbooks such as the one by Åström and Murray [195] for a more detailed introduction into system identification and control design.

For nonlinear systems, however, many design methods exist for specific problem settings, but they lack the generality of approaches for linear systems. Furthermore, in some practical applications, including the setup of this competition, a derivation from first order principles (including all parameters) may be impossible and even the model class may be unknown. In slight deviation of items (1–3) above, one can attempt to control the system without an explicit model of the underlying dynamics, for example, using a PID controller [195]. Alternatively, *model-predictive control* [196,

197] and similar optimization-based controller schemes may not compute a controller explicitly but model the effect of control inputs directly and exploit this model in an online optimization procedure. While model-predictive control typically relies on a given system model with fixed parameters, the field of *adaptive control* [198] considers settings where model or controller parameters need to be tuned online. More recently, researchers in control have been exploring ways to incorporate data-based and machine-learning approaches into control design, partly making the above pipeline more flexible. This emerging area between control and machine learning is known as *learning-based control* or *data-based control*.

**Reinforcement learning** In RL, one commonly starts by defining a *reward* that specifies how desirable it is to transition from one state to another under a given action. The system is modeled sequentially by explicitly accounting for interactions with the system in each time step. The goal is to learn how to interact with the system in a way that maximizes the expected value of the reward. Mathematically, this can be achieved using Markov Decision Processes (MDPs). An MDP consists of a tuple $(S, A, R, T, \gamma)$ [199]. $S$ is the set of states of the system and $A$ is a set of actions that the agent can execute. The reward $R(s, a, s')$ expresses the immediate reward for executing action $a \in A$ in state $s \in S$ and then transitioning to the next state $s' \in S$. $T(s'|s, a)$ is the transition distribution which gives the distribution over next states $s'$ given the current state $s$ and action $a$. $\gamma \in [0, 1]$ is a discount factor that expresses the agent's preference for immediate rewards over long-term rewards. Usually, $S$, $A$ and $\gamma$ are known (user-defined), $T$ and $R$ are unknown. To select an action, the agent applies a policy $\pi(a|s)$ that defines the distribution over the next action, $a$, to execute given the current state $s$. Policies can be stochastic or deterministic. The $t$-th sampled transition thus results in a tuple $(s_t, a_t, s'_t, r_t)$, where $s_t$ is the current state, $a_t$ is the sampled action, $s'_t$ is the next state after the transition, and $r_t \in \mathbb{R}$ is the resulting scalar reward. The goal of learning is to acquire an optimal policy, often denoted as $\pi^*$, that maximizes the expected return $\mathbb{E}_{s' \sim T, a \sim \pi}\left[\sum_t^T \gamma^t r_t\right]$ where T is the duration of the task [200].

## 6.4 Track CHEM: Optimally controlling a chemical reaction

Track CHEM tackles the problem of optimally choosing impulses or shocks in a dynamical system to control a specific part of the system. This task is motivated by applications related to chemical reactions in which one is interested in generating a desired concentration of a specific chemical compound by controlling the initial concentration of some other chemical compounds. When considering such systems there are constraints on how and at what cost experimentation can be performed. To reflect this, we only provided participants with offline training data instead of allowing

them to actively interact with the reactions. The task was to extract knowledge from observed experiments, and use it to control the system in previously unseen settings. Methods that tackle this problem may also apply to systems in which experimentation is infeasible and instead only exogenous shocks to the system can be observed and leveraged for learning.

**Background on chemical reaction networks** In a chemical reaction, one set of chemical compounds is transformed into another. We usually say that reactants are turned into products. Reactants and products are both called species. In Track CHEM, the goal is to find optimal controls (or policies or interventions) on the concentrations of reactants to ensure a desired concentration of one of the species. The dynamical behaviour of species concentrations in chemical reactions is modeled by mass-action kinetics [201], which results in an ordinary differential equation (ODE) over the species. During training, the participants were not able to interact with the system and instead only had access to past observations from the system. For these observations, the applied control inputs were known to participants. The goal in this track is to control one specific process in the observed system when provided only with initial observations. We give more details on chemical reaction networks in Appendix D.1.

**Data generating process** Data is generated by an artificial chemical reaction network. Specifically, a 15-dimensional process $Z(t)_{t\geq 0}$ is generated as:

$$\begin{aligned} Z(0) &= z \\ \dot{Z}(t) &= F(Z(t)) + BU(t), \end{aligned} \tag{6.3}$$

where $U(t) \in [-10, 10]^8$ is the control input at time $t$, $z \in (0, \infty)^{15}$ is an initial value, $B \in \mathbb{R}^{15\times 8}$ is a matrix specifying how the controls influence the dynamics and $F : \mathbb{R}^{15} \to \mathbb{R}^{15}$ is a function from the function class

$$\mathcal{F} = \left\{ F : \mathbb{R}^{15} \to \mathbb{R}^{15} \mid F_\ell(Z) = \sum_{j=1}^{15} \theta_j^\ell Z_j + \sum_{k,j=1}^{15} \theta_{j,k}^\ell Z_j Z_k, \quad \ell = 1, \ldots, 15 \right\}. \tag{6.4}$$

The parameter $\theta$ satisfies additional constraints since we only consider ODE systems that are generated by converting chemical reactions using the law of mass-action kinetics. Furthermore, the rates of the underlying chemical reactions are non-negative which also adds constraints on the coefficients $\theta$.

Among the 15 species, $Y = Z_{15}$ is the species for which the concentration should be controlled. There are eight controls, $U_1(t), \ldots, U_8(t)$, that affect the concentrations of a subset of the species. A (to participants unknown) graphical representation of the model that generated the data for the competition is given in Figure 6.1. The model has a simple structure consisting of four blocks of variables: $\{Z_1, Z_2, Z_9, Z_{13}\}$,

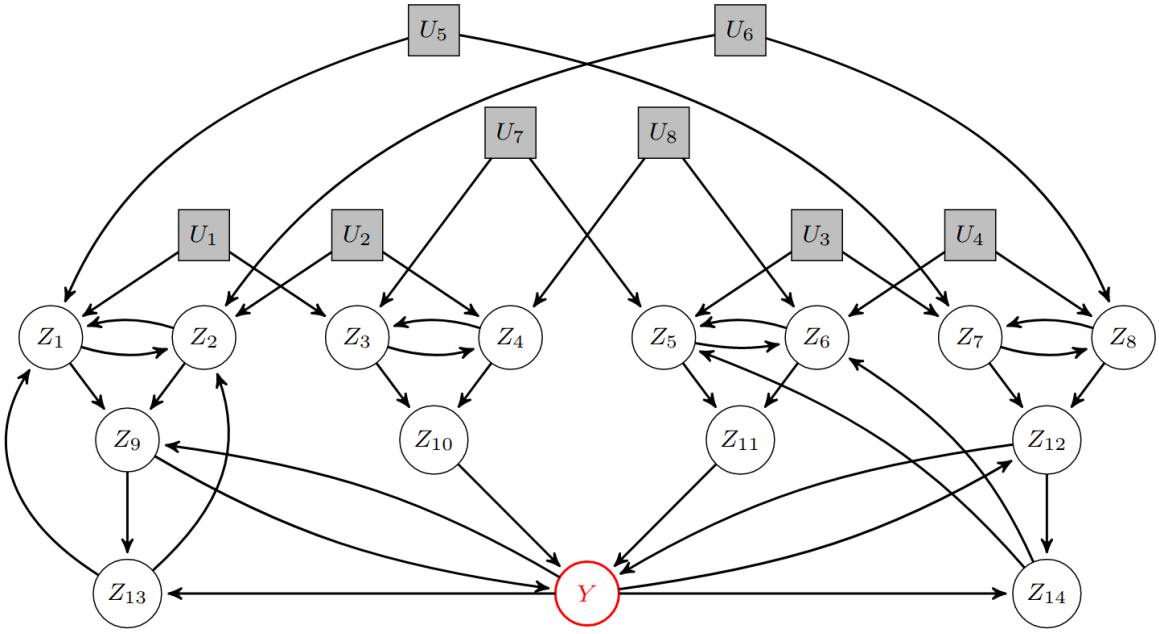



Figure 6.1: Graphical representation of the chemical reaction in Track CHEM.

$\{Z_3, Z_4, Z_{10}\}$, $\{Z_5, Z_6, Z_{11}\}$, and $\{Z_7, Z_8, Z_{12}, Z_{14}\}$. Each block corresponds to an interaction mechanism that can either increase or decrease the concentration of $Y$. There are two types of controls. (1) Control variables $U_1, U_2, U_3$, and $U_4$ affect the system strongly and they affect an increasing and a decreasing block simultaneously. (2) Variables $U_5, U_6, U_7$, and $U_8$ have a weaker effect on the system but they target only an increasing block ($U_7$ and $U_8$) or only a decreasing block ($U_5$ and $U_6$). Hence, the controls in (2) offer an easy strategy to control $Y$ but are expensive, while using the controls in (1) is cheaper but might be more difficult. The participants only observed data with pre-specified control settings where we had incorporated confounding structure in the observed controls; the problem therefore also featured a causal challenge.

**Task** Participants knew the function class but did not know the parameters. They did not know the graphical structure of the system, either. Furthermore, participants did not observe the process $Z$ directly. Instead, they only observed $X$, a noisy version of the process, sampled on a time grid $(t_0, \ldots, t_L)$; that is, the observed data is sampled, at $t \in \{t_0, \ldots, t_L\}$, from the process

$$X(t) = (Z_1(t), \ldots, Z_{15}(t)) + N(t), \tag{6.5}$$

where $N$ is a mean-zero noise process such that $\{N(t) : t = t_0, \ldots, t_L\}$ are independent. The goal of the task is to choose a value $u \in \mathbb{R}^8$ such that the controls

$$U(t) = \begin{cases} u & \text{if } t \in [t_0, t_3) \\ 0 & \text{if } t \geq t_3, \end{cases} \tag{6.6}$$

lead to $Y$ being close to a (pre-specified) desired value $y_*$ at the end of the reaction. As the value $u$ for the control is set only once and after having observed $X(0)$, Track CHEM is an open-loop problem.

Participants had access to data from 12 different ODE systems which are specified by the functions $F^1, \ldots, F^{12} \in \mathcal{F}$ and they knew the index of the system from which data originated. The function class $\mathcal{F}$ was known but $F^1, \ldots, F^{12}$ were unknown to the participants. Participants knew that the 12 systems had the same structure, that is, the same parameters $\theta_j^\ell$ and $\theta_{j,k}^\ell$ were zero in all 12 systems, and that every $\theta_j^\ell$ and $\theta_{j,k}^\ell$ had the same sign in all systems. The parameters of the noise as well as the matrix $B$ were the same in all 12 systems and these facts were also known to participants.

The training data available to participants was generated by running the data generating process 20 times for each $F^i$ with different pairs of initial conditions $z$ and controls $u$. The distributions used to select $z$ and $u$ in the training data were unknown to participants and differed among systems.

**Evaluation** For each of the systems participants were provided with 50 additional sets of initial vectors, $X(0)$, as well as an indicator specifying the corresponding system ($i = 1, \ldots, 12$). For each of these combinations participants were asked to select a control input to minimize the loss function. The loss function measures the proximity of $Y^{i,k}$ to the desired value $y_*^{i,k}$ toward the end of the observation interval while also adding a penalty term depending on the size of the control input used. The exact loss function can be found in Appendix D.1.

**Three perspectives** *Causality* The system is causal in the sense that it specifies not only an observational distribution ($U \equiv 0$ in Equation (6.3)) but also a set of interventional distributions ($BU \not\equiv 0$ in Equation (6.3)). The intrinsic dynamics are the same regardless of which (if any) intervention is applied as they are described by the function $F$. That is, the mechanism described by $F$ is stable under interventions. The task can be thought of as a causal learning task where participants should predict the effect of interventions and choose an optimal intervention.

*Control theory* The task seeks a functional map from measurements $y(t)$ to control inputs $u(t)$ which is the classical task of *control design*. The control inputs are to be of the form (6.6), which can be understood as *impulse control*. Formalizing the control objective as an optimization problem is commonly known as *optimal control* (see, for example, Bertsekas [202] or Anderson and Moore [203]). The objective function is a

weighted sum that, as typical in control applications, balances *control performance* and *control effort*.

*Reinforcement learning* In this task, the vector $u$ is selected at the start of each trial and then executed for a number of steps. This problem formulation without state transitions is closely connected to bandit or contextual bandit problems [199] where the agent receives a reward based solely on the selected action and context. Classical online RL approaches learn the policy by iteratively interacting with the environment and improving the policy. However, for the proposed task, the agent will need to use *offline* reinforcement learning as no interaction with the system is possible (see, for example, Lange et al. [204] or Levine et al. [205]).

## 6.5 Track ROBO: Controlling a robotic arm in a dynamical environment

Track ROBO is motivated by the long-term goal of learning skills that can be performed by a diverse set of robots to complete real-world tasks. For instance, one may want to teach robots new skills such as stirring a pot or cutting vegetables for cooking. The new skills can be learned more efficiently by leveraging prior experience in related tasks such as whisking eggs. Robots may also have different kinematic structures, requiring individualized control policies to accurately execute the end-effector trajectory required by a new skill. Even robots of the same type can differ because of minor variations in the production process. If we can leverage a robot's prior movement data to derive an individualized controller for a new skill, we may avoid the need for additional training and enable rapid roll-out of new skills.

We mimic this challenge in Track ROBO: participants are provided with movement data and asked to provide a controller that sequentially interacts with a robotic arm such that its end-effector reaches a target position provided for the next time step. The two difficulties are: (1) Participants can only set abstract control variables instead of, for example, setting the torques of individual joints directly. This restriction imitates a setting in which the robot dynamics are complicated to write down explicitly. (2) The training data is comprised of different types of trajectories than those in the test data, imitating a setting in which the robot must adjust to a new task given previous data of an old task.

We consider three robot arms: (1) A two-joint rotational robot arm, (2) a three-joint rotational robot arm, and (3) a two-joint prismatic robot arm. The rotational joints produce a rotary motion around the joint, and the prismatic joints produce a linear motion between links (see Figure 6.2). Each joint can be controlled by applying a voltage signal to a DC motor located in the joint. (In this challenge, the voltage is not set directly, see below.) In rotational joints, this creates a torque, while in prismatic joints this creates a linear acceleration. The resulting movement of the joints of the robot arm (and its tip in particular) are governed by the physical

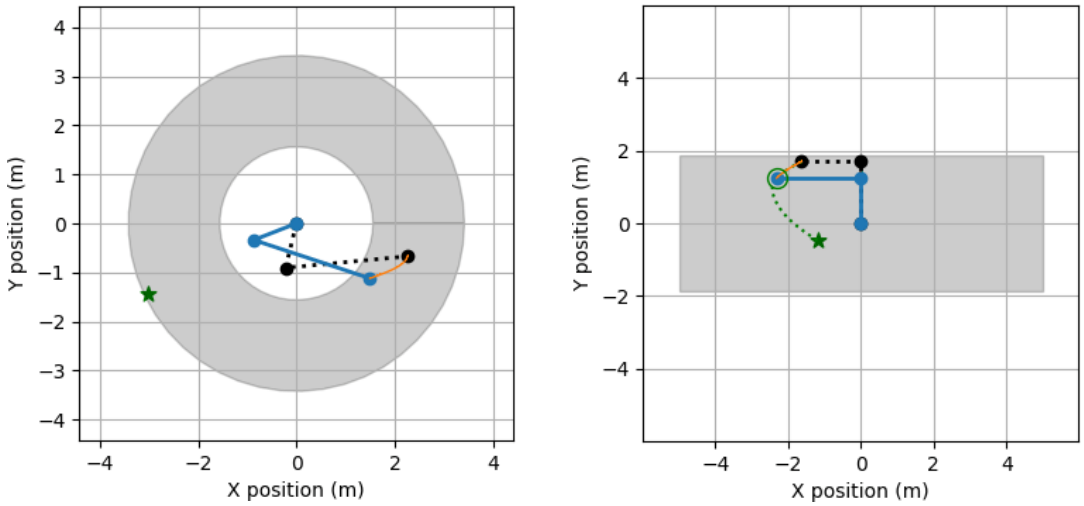



Figure 6.2: (Left) Rotational 2-link robot, the trail of previous positions of the robot tip (orange line), and the target position (green star). (Right) Prismatic 2-link robot and a target trajectory (green dotted line) to a target position (green star). The gray area corresponds to the reachable workspace of the robots. The black dotted lines indicate the initial position of the robots.

laws of motion. Given a specific robot one can derive exact differential equations that describe the robot's movement, known as the *dynamics model* (or *dynamics* for short) of the robot with parameters that depend on various specifications of the robot such as link mass, rotational moment of inertia, link length, location of center of mass, and friction coefficients, see Appendix D.2.

**Data generating processes** Participants were able to sequentially interact with 24 different robotic arm systems, with dynamics given by

$$\begin{aligned}(Z(0), W(0)) &= (z, w)\\ (\dot{Z}(t), \dot{W}(t)) &= F^s(Z(t), W(t), C(t)) = F^s(Z(t), W(t), A^s \cdot U(t)),\end{aligned} \tag{6.7}$$

for $s \in \{1, ..., 24\}$ where $Z(t) = ((X(t), Y(t)) \in \mathbb{R}^2$ is the position of the tip of the robot, the positions of other joints are $W(t) = (X_1(t), Y_1(t), ..., X_d(t), Y_d(t)) \in \mathbb{R}^{2d}$, $z \in \mathbb{R}^2$ and $w \in \mathbb{R}^{2d}$ are the initial values, and $C(t) \in \mathbb{R}^q$ are the underlying robot controls, that is, voltage signals applied to DC motors located in each joint ($d$ and $q$ depend on the underlying robot). The (to participants unknown) functions $F^s : \mathbb{R}^{2(d+1)+q} \to \mathbb{R}^{2(d+1)}$ are given by the second-order dynamic system of the underlying robots: $F^1$, ..., $F^8$ and $F^9$, ..., $F^{16}$ correspond to 2-link ($d = 1$) and 3-link ($d = 2$) open chain planar manipulators with revolute joints (cf. D.2.1), respectively, and $F^{17}$, ..., $F^{24}$ correspond to 2-link ($d = 1$) prismatic manipulators (cf. D.2.2); the dynamics differ further between robots due to different robot specifications denoted by $\theta^s$ (link

masses and lengths, moments of inertia, friction coefficients, and locations of link center of masses). The (to participants unknown) *interface function* $G^s : \mathbb{R}^p \to \mathbb{R}^q, x \mapsto A^s x$ for $A^s \in \mathbb{R}^{q \times p}$ relates the participants' abstract control inputs $U(t) \in \mathbb{R}^p$ to the underlying robots' control inputs $C(t)$ via $C(t) = G^s(U(t))$; $A^s$ is either the identity, a square ($p = q$), or a rectangular ($p > q$) real matrix with imbalanced row-norms and full rank. Participants can control the systems only on a linearly spaced discrete time grid $(t_0, t_1, \ldots, t_{200})$ with $t_0 = 0$ and $t_{200} = 2$, that is, for each time step $\ell \in \{0, \ldots, 199\}$ it holds that $C(t) = G^s(U(t)) \equiv \text{const}$ for all $t \in [t_\ell, t_{\ell+1})$. Some of the 24 systems share robot dynamics ($F^s$), specifications ($\theta^s$), and/or the control interfaces ($A^s$) in a systematic way, which is reflected in the naming convention but was not explicitly revealed to the participants (cf. Table D.1 for an overview).

**Task** The competition task is to control the robots' end-effector position $Z(t)$ to follow a given target process $t \mapsto z_*(t)$. More specifically, participants needed to implement a controller, that is, a function $\texttt{controller}^s : \mathbb{R}^{2(d+1)+2} \to \mathbb{R}^p$ for each robot ($s \in \{1, ..., 24\}$). At each time step $\ell \in \{0, \ldots, 199\}$, the controller is queried for the next control input $U(t_\ell) \in \mathbb{R}^p$ given the current positions $Z(t_\ell), W(t_\ell) \in \mathbb{R}^{d+1}$, their derivatives $\dot{Z}(t_\ell), \dot{W}(t_\ell) \in \mathbb{R}^{d+1}$, and a target end-effector position $z_*(t_{\ell+1}) \in \mathbb{R}^2$ for the next time step. The task does not involve planning as the controller only gets access to the next time step's end-effector target position, however, the implemented controller can gather information during the control process[2]. If the controller does not return within given compute time and resource constraints, we set $C(t_\ell) = G^s(U(t_\ell)) = \mathbf{0}$. This way, the different robots are propagated forward for different target trajectories $(z_*(t_0), ..., z_*(t_{200}))$ following their respective dynamics under the participant-provided controller; the participants' task is to align the resulting end-effector trajectory $(Z(t_0), ..., Z(t_{200}))$ with the target trajectory.

For deriving and implementing their controllers, participants were provided with (offline) training data for each system $(F^s, \theta^s, A^s)$ in the form of 50 realized end-effector trajectories and corresponding control input sequences. Training trajectories are obtained using an LQR-controller [203] (based on inverse dynamics and the (pseudo-)inverse of $G^s$ to map robot controls to participant controls) to transition from some random starting positions to some random target positions in the robot's workspace. The one-step ahead end-effector target positions used to generate the training trajectories were not provided to participants. For each of these repetitions, participants were provided with the observed processes $W$ and $Z$, their derivatives $\dot{W}$ and $\dot{Z}$, a time indicator $t$, the applied controls $U$ and an indicator $i$ specifying the system.

[2]By implementing $\texttt{controller}^s$ as a function with state to log previous queries, the controller can at each time step also be viewed as a function of positions $Z(t_0), W(t_0), ..., Z(t_\ell), W(t_\ell)$, derivatives $\dot{Z}(t_0), \dot{W}(t_0), ..., \dot{Z}(t_\ell), \dot{W}(t_\ell)$, and targets $z_*(t_1), ..., z_*(t_{\ell+1})$, while the interface between robot and participant controller was specified as a function of only the current position, derivatives, and target.

**Evaluation** For each of the 24 systems $(F^1, \theta^1, A^1), \ldots, (F^{24}, \theta^{24}, A^{24})$ the participants' controller implementation is used to follow 10 different target processes. More specifically, for each system $i \in \{1, \ldots, 24\}$ and repetition $k \in \{1, \ldots, 10\}$, there is a target process $z_*^{i,k} : [0, 2] \to \mathbb{R}^2$ and the robot is propagated forward using the participant-provided controller. The loss function measures how far the realized end-effector trajectory is from the target process and penalizes the size of the participants' control inputs (cf. Appendix D.2 for details). Mimicking a real-time robot control scenario, the participants' computational resources spent on evaluating the code that implemented their controller were restricted. If the time constraints were not met, the submission was invalid.

**Three perspectives** *Causality* We can formulate the task of controlling a robot as a causal task. First, we need to estimate a model that allows us to evaluate the effect of various interventions, where interventions now correspond to setting the inputs $U(t)$. Second, we optimize a sequential intervention scheme.

*Control* The task consists of both system identification and controller design and it relies on all the steps in a typical control engineering application as outlined in Section 6.3.

*Reinforcement learning* Track ROBO has a standard MDP formulation wherein the action is defined by the command sent to the abstract controller and the state space is given by the joint positions and velocities. The time-varying reward is given by the robot's accuracy in following the desired trajectory with a penalty on the size of the control inputs.

## 6.6 Results and lessons learned

The results of the competition can be found on our website[3] and in the appendices. The website also contains videos, in which some of the competing teams describe their solutions in more detail. Code is available, too[4].

For us, one of the key questions was whether, for the model classes considered in this challenge, it is better to aim to build a model and then infer the optimal control, or to directly estimate the effect of the applied control. The competition results suggest the former in that in both tracks the winning solution was indeed inferring a data-generating model first (see in D.1.1 and D.2.3 in the appendices); it

[3]See learningbydoingcompetition.github.io

[4]See github.com/LearningByDoingCompetition/learningbydoing-comp for open-source code to reproduce the winning solutions of the competition and to try out new methods on the competition tasks; see [206] for code that implements the winning solutions to Track CHEM and Track ROBO; see github.com/Quarticai/learning_by_doing_solution for code that implements the second winning solutions to Track CHEM and Track ROBO; see [207] for code that implements the third winning solution to Track CHEM; see [208] for code that implements the third winning solution to Track ROBO.

outperformed approaches of the latter type by a significant margin. We speculate that imposing a model structure (even if both the structure and the parameters still need to be inferred from data) acts as strong regularization helping to ensure successful control which is robust to environmental changes. Clearly, further research is needed to better understand in which settings this is expected to be the case. In the future, it would be interesting to consider situations where the model inference becomes even harder, for example, because more parts of the system are unobserved.

## 6.7 Acknowledgments

We thank the NeurIPS 2021 competition track organizers, Barbara Caputo, Marco Ciccone, and Douwe Kiela. We also thank all the participants who took part in the competition. SW and JP were supported by the Carlsberg Foundation. SWM was supported by a DFF-International Postdoctoral Grant (0164-00023B) from Independent Research Fund Denmark. TL and OK were supported by the Office of Naval Research under Grant No. N00014-18-1-2775. IG was supported by ANR Chair of Artificial Intelligence HUMANIA ANR-19-CHIA-0022. JP was supported by a research grant (18968) from VILLUM FONDEN. NP was supported by a research grant (0069071) from Novo Nordisk Fonden. The competition has been supported by the Department of Mathematical Sciences at the University of Copenhagen.

# 7 Hybrid Causal World Models: Integrating Latent and Semantic Information

As evidenced by our previous work in the Learning by Doing competition (Ch. 6), there exists a rich overlap between causality and dynamical systems. This remains true for *world models*, models that capture environment dynamics. Typically, world models learn a latent state space through unsupervised representation learning. While advantageous in certain regards (e.g., a hand-specified state space is not required), a learned representation may be difficult to understand or interpret. Moreover, subject-matter expertise of a particular robotic system may be known in *semantic*, not latent, terms. This work, LMeshNet, explores how semantic information can be integrated into a world model, yielding a *hybrid world model* that synchronizes latent and semantic information. Specifically, semantic information is integrated by the addition a separate semantic world model, which operates synchronously with a given and pretrained latent world model. This hybridization procedure yields a hybrid world model that comprises one latent world model and one semantic world model. Synchronization of both halves of the hybrid world model is provided at inference time through blending latent-to-semantic predictions into the semantic space. In cases where the (latent) world model is a *causal world model*, the latent space may have favorable disentanglement properties, and causal discovery can then be used to provide interpretability for the latent space based on the semantic space. The resulting *hybrid causal world model* provides greater utility and interpretability as compared to standard world models.

**Application example.** To provide a concrete example, one such application could be a depth-based robot tracking system for a dynamical object, where a (latent) causal world model captures dynamics through a depth observation. Assuming there is a particular point of interest or fiducial on the object, a semantic world model can be used to track the distance to this fiducial through a unidimensional range observation. As both the depth and range observations have overlapping information, the causal discovery procedure used by this work can recover the overlapping interrelationships

between the latent space and the semantic space. This procedure provides two major benefits. First, interpretability arises by understanding which the dimensions of the latent space relate to the semantic space. Second, only the dimensions of the latent space obtained from causal discovery can be used for latent-to-semantic predictions, rather than using all dimensions, leading to a smaller network and an approach that scales with the dimensionality of the latent space.

**Same-environment synchronicity and benefits of learning causal structure.** Our work leverages Variational Causal Dynamics (VCD) [209] as the causal world model used for hybridization. VCD has been shown to more accurately learn the underlying dynamical structure through causal representation learning, and furthermore, this approach was used to adapt to new environments. Our work focuses primarily on synchronizing semantic information with the same environment distribution that was used to train the latent space, although our work can be extended to consider cases of adapting this synchronization to new environments. Moreover, we show that the advantage of learning the causal structure in the latent side of the hybrid world model also benefits the semantic world model in terms of more accurate long-term semantic predictions, as compared to hybridizing a standard world model that is agnostic to causal structure.

**Identifiability.** One important consideration for world models is that of identifiability: whether the underlying latent causal processes can be identified and recovered. For LMeshNet, we demonstrate that causal discovery can obtain the relationship between the latent state and given semantic information for a hybrid causal world model. Our experiments show that the latent state captured the ground truth causal processes, and the semantic information was one of the ground truth causal processes. Therefore, this relationship could be found. For this work, we assume that overlap exists between the latent state and semantic information. In general, this assumption will hold if 1) the ground truth causal processes are identifiable and recovered with the latent state, and 2) the semantic information is related to the ground truth causal processes. Specifically, we require that a mapping exists between the latent causal processes and the semantic information used for hybridization.

The requirement for identifiability of the latent state is important, as LMeshNet inherits the identifiability and modeling assumptions of the latent world model. This study uses Variational Causal Dynamics, which models each latent variable as an independent causal mechanism [209]. However, Liu et al. recently showed this assumption may be overly restrictive and instead presented a general factorization for causally-related latent processes [210]. Specifically, in IFactor, Liu et al. model four distinct categories of latent variables (reward-relevant and controllable, reward-relevant but uncontrollable, reward-irrelevant but controllable, and reward-irrelevant and uncontrollable) and prove block-wise identifiability of these variables [210]. In principle, other causal world models and factorizations, such as IFactor, should yield a causal representation that would permit the relationship between the latent and semantic information to be found, assuming that the assumptions stated above are

satisfied. These assumptions may also restrict the types of tasks for which this approach can be used. Tasks with more complex relationships or structures may require interventions.

Lastly, a related assumption is that of observability. The experiments in LMeshNet concern a multi-particle dynamics problem, where the underlying causal processes could be inferred and identified. If the particles became occluded or otherwise not completely observed, then it may not be possible for the latent state to be identified or the relationship with the semantic space to be found.

For this work, our study in learning the relationship between latent and semantic information is empirical, and we believe there is rich future work in understanding the theoretical aspects of discovering this relationship and its implications for identifiability. For more interest in state space identifiability, we direct the reader to Yao et al. [211, 212], Huang et al. [213], Lippe et al. [214, 215], and Liu et al. [210].

*This work was completed during an internship at Lockheed Martin Space Advanced Technology Center. We thank our Lockheed Martin collaborators on this work: Joseph Gleason, Shruti Mahadevan, Daniel Kolosa, and Eric Dixon. We also thank Anson Lei for providing an official implementation of VCD [209].*

## 7.1 Summary

*Although world models provide a compelling model-based approach to learning environment dynamics, the latent state representation that is learned is often uninterpretable. Therefore, the utility of these world models is limited when comparing against semantic state information about the environment that is known by domain experts. Recent work has extended world models to yield a causal latent representation, leading to a causal world model. In this work, we propose the hybrid causal world model, which integrates known semantic information with a pretrained causal world model. Importantly, our approach uses causal discovery to learn the relationship between the world model latent space and the semantic space, rendering the learned state representation interpretable. Our experiments show that the hybrid causal world model can successfully synchronize both latent and semantic representations, improving world model utility and functionality.*

## 7.2 Introduction

A hallmark of human intelligence is the capability of creating *mental models* that capture our perception of the world for use in reasoning [216]. The promise of endowing machine learning agents and robots with similar cognitive capabilities has lead to the development of learning *world models*: models of environment dynamics that can be learned from observations, akin to how humans develop such models of the

world from our senses. Through representation learning, world models typically learn the dynamics in terms of a *latent* state representation, as opposed to a (commonly hand-provided) state representation that often is understandable in *semantic* terms. Learning a latent state space can be particularly advantageous in domains where it is not clear how a semantically meaningful state space can be constructed.

However, the advantages of a latent state space yield disadvantages towards their utility. Such learned latent spaces are not inherently interpretable, as they arise from learning an opaque world model. It is not obvious how the learned latent space relates to a state space that is known to end users of systems, such as robotics practitioners who monitor the success of these systems in real operational scenarios. Subject-matter expertise may be known in terms of semantic information, making analysis of latent information cumbersome or even impossible. For this reason, there may be semantic knowledge about the system that is desirable to track in a certain representation known to end users, even if it is not possible to construct an entire state space that would be semantically meaningful. Yet, this semantic information may also encapsulate knowledge that is also known by the latent space. How can latent and semantic information work harmoniously together?

To this end, we introduce LMeshNet,[1] a "best of both worlds" approach for one world model that encapsulates *both* a learned latent model and a semantic model. We assume the semantic model captures information known already about the system that is desirable for end users. When uniting the semantic world model with a structured world model that more strongly captures the underlying causal structure of the world (i.e., a *causal world model*), LMeshNet creates a *hybrid causal world model* that leverages the favorable disentanglement properties of the latent space. The contributions of this work are as follows:

- We propose a methodology to "hybridize" an existing world model, integrating the world model's latent space with a given semantic space.
- We show that with a causal world model, causal discovery can elicit a latent subspace that provides overlapping information with the semantic space, leading to interpretability of the latent space and a reduction in model parameters needed for a hybrid causal world model.
- We perform experiments that demonstrate that integrating the semantic space provides greater utility, and long-term predictions of the semantic space can be stabilized from the latent space when no observations are available.

[1] LMeshNet stands for "**L**atent **Me**shed with **S**emantic **H**ybrid **Net**work." The name refers to how the latent and semantic spaces are *meshed* together, working synchronously. LM is also a reference to Lockheed Martin.

## 7.3 Related Works

**World models and dynamics models.** The goal of a world model is to learn a latent dynamics model of an environment, typically from high-dimensional observations (e.g., images) and often for use in model-based reinforcement learning [217–219]. Representative examples includes the model introduced by Ha and Schmidhuber [151] and several models by Hafner et al. [220–223]. PlaNet [220] uses a latent dynamics model for planning, and this work was subsequently followed by Dreamer [221] for learning long-horizon behaviors in the latent space. DreamerV2 [222] builds from Dreamer, using a categorical (rather than Gaussian) latent distribution and focusing on Atari tasks. Recently, DreamerV3 [223] expands on DreamerV2 in various areas, such as using symlog predictions and unimix categorical distributions. Notably, DreamerV3 enables a Minecraft agent to collect diamonds from scratch. Other works that investigate learning a dynamics model include E2C [224], RCE [225], and SOLAR [226].

**Structured world models.** Coinciding with broader interest in causal machine learning [37] and causal representation learning [25], recent works have explored imbuing world models with greater causal structure [209, 227, 228]. Zhu et al. [227] investigate training a structured world model using offline reinforcement learning. Their contributions include 1) a theoretical result showing causal world models are more performant than standard world models, and 2) the FOCUS algorithm for learning the underlying causal structure from offline data. Lei et al. [209] introduce Variational Causal Dynamics (VCD), which uses differentiable causal discovery to learn a world model with a structured transition model that captures the causal structure of the environmental dynamics in both observational and intervened settings. Poudel et al. [228] proposes a methodology for training a world model using a contrastive loss and an interventionally-invariant auxiliary task to learn invariant causal features. Our work explores the use of "hybridizing" a causal world model with semantic information.

**Causal interpretability.** Through causal discovery, LMeshNet provides a degree of interpretability to a causal world model's latent space by identifying dimensions that overlap with the provided semantic information. In this view, our work can be seen as a method for causal interpretability [229], a subset of interpretable machine learning [230–233] that specifically seeks to provide understanding through a causal lens.

## 7.4 Preliminaries

### 7.4.1 World Models

A *world model* learns a dynamics model of an environment in terms of a latent state space $z$ that emerges from unsupervised representation learning. Typically, this representation is learned from high-dimensional observations $o$, such as images. The distribution of the latent space $z$ is a design choice, with examples such as Gaussian [151, 209, 221] or categorical [222, 223]. Figure 7.1 illustrates the standard architecture of a world model. Major components include 1) the *encoder*, which models the probability $P(z^t|o^t)$ and is referred to as the *posterior*; the *decoder* (or *observation model*) which models $P(o^t|z^t)$ and reconstructs the observation $\tilde{o}$ from the latent state $z$; and a *transition model* $T$ that models the forward progression of dynamics $P(z^{t+1}|z^t, a^t)$, where $a$ is the environment action. As shown in Fig. 7.2a, the transition model $T$ is typically modeled as a recurrent neural network (RNN) that is fully connected with the latent space $z$. Because the transition model $T$ in entirely in the latent space (i.e., without incorporating observations), long-term predictions can be achieved by forward simulating the environment dynamics model with given actions $a$. This capability has been referred to as the "latent imagination" [221] of world models. The encoder and decoder are typically modeled as feedforward neural networks.

For this work in comparing to standard world models, we use the recurrent state space model (RSSM) [220] as the implementation of the transition model $T$. The RSSM comprises two parts: 1) a deterministic part $h^{t+1} = f_d(h^t, z^t, a^t)$, where $f_d$ is a gated recurrent unit (GRU) [234] and $h^t$ is the associated hidden state; and 2) a stochastic part, where $z^{t+1} \sim P(z^{t+1}|h^{t+1})$.

### 7.4.2 Causal World Models: Variational Causal Dynamics

Despite the success of world models, it is unclear how high-dimensional observations relate to the low-level latent space. To this end, Variational Causal Dynamics (VCD) [209] introduces a structured world model with a causally factorized transition model (Fig. 7.2b), yielding a *causal world model* that is learned from differentiable causal discovery. This causal approach lends itself toward modeling environment dynamics in terms of independent mechanisms which can be modeled only with causally relevant latent variables. Moreover, by the Sparse Mechanism Shift hypothesis [25, 235], domain shifts in the environment data generating process arise due to sparse and local changes, and therefore other mechanisms remain invariant across these domains. Therefore, VCD also captures how these independent mechanisms change from an undisturbed environment to those with interventions.

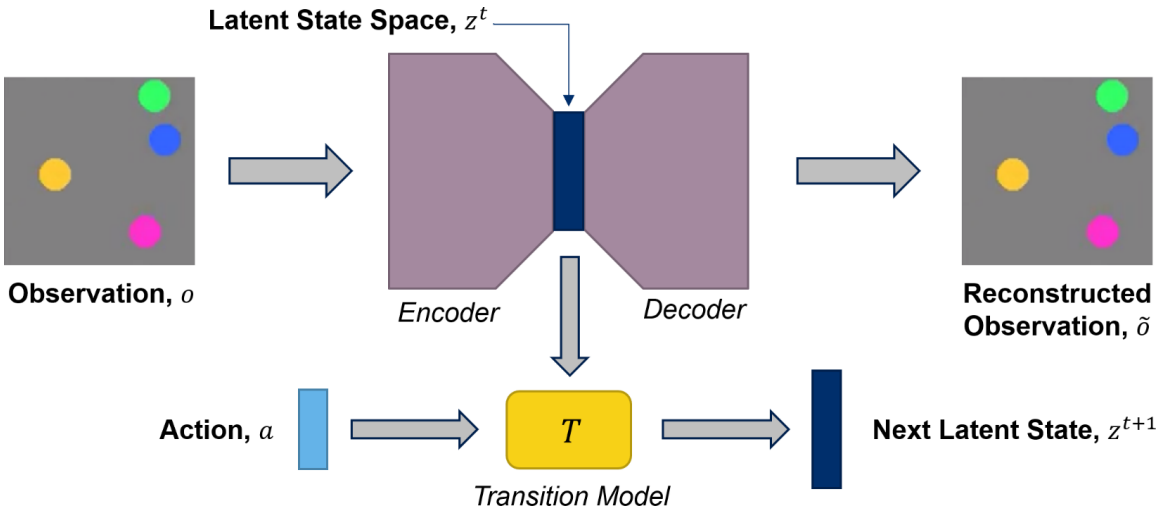


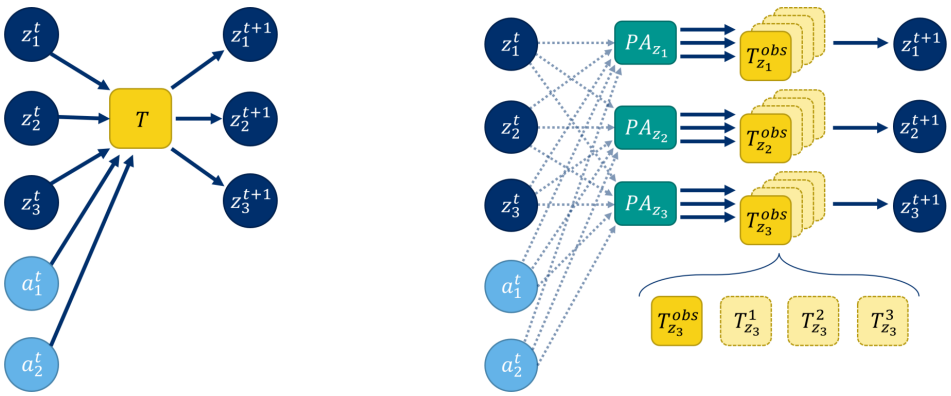

Figure 7.1: A *world model* typically consists of the following components: an *encoder* that takes as input an *observation* $o$ and learns a latent state space embedding $z^t$; a *decoder* that, from a latent state $z^t$, yields a *reconstructed observation* $\tilde{o}$; and a *transition model* $T$ that learns the forward dynamics of the system (in the latent space) from the latent state $z^t$ and action $a$ in order to predict the next latent state $z^{t+1}$.

To accomplish this, along with jointly learning the latent representation and transition model (as in standard world models), VCD also jointly learns a causal graph $\mathcal{G}$ of the environment dynamics and intervention targets $\mathcal{I}_k$ across $K$ intervened environments for use in the causally factorized transition model. This is achieved by extending Differentiable Causal Discovery from Interventions (DCDI) [236] for use with latent variables. These improvements in imbuing greater causal structure into the world model lead towards learning a causal latent representation that has favorable disentanglement properties as compared to standard world models.

For our work, we use VCD as our representative causal world model. Notably, VCD is the first structured world model to implement a causal transition model with high-dimensional inputs.

## 7.5 Hybridization of World Models

LMeshNet integrates a semantic space $s$ in concert with an existing world model, leading to a *hybrid world model* with two halves for the latent and semantic spaces (Fig. 7.3). This semantic space $s$ enables world model interpretability along dimensions that may be critical for system practitioners to monitor in a specific representation. We assume the semantic space $s$ is inferred from a semantic observation, $o_s$, which is distinct from the observation $o$.




(a) (b)

Figure 7.2: World models can become *causal world models* depending on their transition models and latent space representation. (a) A standard transition model is typically implemented as a recurrent neural network that takes a fully connected input of $z^t$ and $a^t$ and yields a fully connected output of $z^{t+1}$. (b) In VCD [209], a separate transition model exists for each dimension of the latent state $z$, and the inputs of each transition model are the parents according to a learned causal graph. In other words, the transition model for $z_1$ ($T_{z_1}$) would predict $z_1^{t+1}$ from the parents of $z_1$ ($PA_{z_1}$). Moreover, VCD learns environment-specific models (when a particular environment has an intervention) along with an observational model when there are no interventions. Shown is the case where there 4 environments total, one without interventions and three with environment-specific interventions.

We assume that this semantic space $s$ is known and provided to LMeshNet as a part of the hybridization process. Moreover, we assume that there exists some overlap of information between the observation $o$ and semantic observation $o_s$, such that latent information could in principle be used to augment semantic information. For example, a depth-based robotic system for tracking an object could utilize a hybrid world model with depth observations as $o$ and a unidimensional range measurement (i.e., to a particular fiducial on the object) as $o_s$.

### 7.5.1 Semantic World Model

As shown in Fig. 7.3, the hybrid world model is created by integrating a latent world model with a semantic world model. The components of the semantic world model are generally similar to those in a standard world model, except that the models are intended to learn a particular semantic space $s$ of interest using a semantic observation $o_s$. The semantic world model also contains components to synchronize the latent and semantic spaces (c.f., Sec. 7.5.2), which may primarily be of concern when conducting long-horizon predictions.

In principle, it is possible to have a completely orthogonal semantic space $s$ and

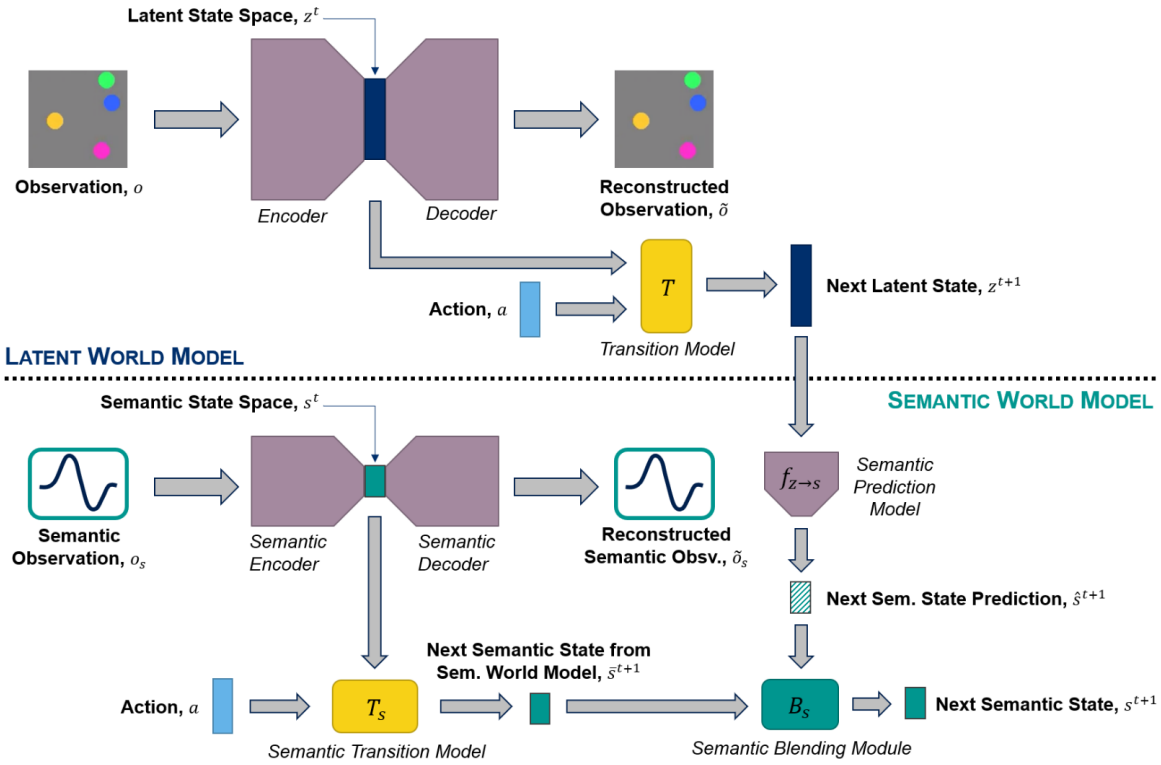



Figure 7.3: The *hybrid world model* comprises a *latent* world model and a *semantic* world model, with machinery to synchronize these two halves. The latent world model is as described in Fig. 7.1. The semantic world model uses the standard world model architecture to model specific, semantically meaningful dimensions within the *semantic state space* $s^t$. The *semantic encoder*, *semantic decoder*, and *semantic transition model* operate similarly to a (latent) world model. Synchronicity of the two halves for predicting the next semantic state $s^{t+1}$ is provided by aligning the semantic and latent halves through the *semantic prediction model* $f_{z\to s}$ and the *semantic blending module* $B_s$.

observation $o_s$ that shares no information with the latent world model. For these cases, synchronicity does not become a concern, and the two halves of the world model will operate independently. However, such cases are not examined in this work.

### 7.5.2 Synchronizing the Latent and Semantic Spaces

As the latent world model and semantic world model operate separately, disagreement may arise between the latent and semantic spaces. When the world model is continually updating the state spaces at each timestep with new observations $o$ and $o_s$, this disagreement may be bounded and within acceptable tolerances. However, in long-horizon forward predictions without new observations, the disagreement may grow unacceptably large, leading to desynchronization of the hybrid world model. To ensure the latent and semantic spaces remain synchronized, the semantic space can be updated by predictions from the latent space.

Synchronicity of the hybrid world model is provided at inference time by two components: 1) a *semantic prediction model* $f_{z\to s}$ and 2) a *semantic blending module* $B_s$.

**Semantic prediction model.** The semantic prediction model $f_{z\to s}$ takes as input the next latent state $z^{t+1}$ and outputs a prediction for the next semantic state $\hat{s}^{t+1}$. This model is implemented as a multilayer perceptron that is either fully connected with $z^{t+1}$ or, if the latent space has favorable disentanglement properties, the dimensions can either be prespecified or learned by causal discovery (c.f., Sec. 7.5.3).

**Semantic blending module.** The semantic blending module $B_s$ blends $\hat{s}^{t+1}$ with $\bar{s}^{t+1}$, the next semantic state from the semantic world model, to yield $s^{t+1}$. This module implements the weighted sum of two independent Gaussian distributions:

$$\mu_{s^{t+1}} = (1-\beta)\mu_{\bar{s}^{t+1}} + \beta\mu_{\hat{s}^{t+1}} \tag{7.1}$$

$$\sigma^2_{s^{t+1}} = (1-\beta)^2\sigma^2_{\bar{s}^{t+1}} + \beta^2\sigma^2_{\hat{s}^{t+1}} \tag{7.2}$$

where $\mu_{s^{t+1}}$ and $\sigma^2_{s^{t+1}}$ refer to the mean and variance, respectively, of the next semantic state and $\beta$ is a hyperparameter that controls the degree to which the latent prediction is blended into the semantic space. When $\beta = 0$, no blending occurs and the hybrid world model operates asynchronously. When $\beta = 1$. the semantic world model becomes disabled, and the semantic space is exclusively estimated from the latent space.

### 7.5.3 Causal Discovery of Shared Latent/Semantic Dimensions

Although it is possible to use all dimensions of the latent space $z$ to infer the semantic space $s$ using a fully-connected $f_{z\to s}$, in practice, this naïve assumption may

lead to unnecessary complexity and network size, particularly if the latent dimensionality is large. Such unnecessary complexity may be unacceptable for hardware with SWaP (Size, Weight, and Power) constraints, such as robotic space systems and satellites. Therefore, for certain applications, it is critical to determine only the necessary dimensions of the latent space $z$ that are needed for predicting the semantic space $s$.

To do this, we use causal discovery to determine the interrelationships between the latent space $z$ and semantic space $s$. We first generate a dataset consisting of trajectories of the latent variables $z$ and semantic variables $s$. Then, we use the TS-ICD [237] algorithm to learn a temporal graph $\mathcal{G}_{z \to s}$ with hyperparameter $\alpha$ (controlling graph sparsity) and window size of 2. Thus, the learned graph $\mathcal{G}_{z \to s}$ encodes the relationships between the preceding timestep ($z^{t-1}$ and $s^{t-1}$) and the present timestep ($z^t$ and $s^t$). To extract the dimensions, we focus on the contemporaneous subgraph and take the dimensions of $s^t$ that are connected to $z^t$. In principle, the edges between $s^t$ and $z^t$ will often be bidirectional, corresponding to having a common cause (the physical mechanism captured by the observations $o$ and $o_s$), rather than being directed. Nonetheless, a bidirectional edge suggests statistically which dimensions of latent space $z$ overlap with $s$.

We note that using causal discovery to determine the latent/semantic overlapping dimensions requires a compatible latent representation and depends on the degree of disentanglement within the latent space. We find that latent representations obtained with structured world models from causal learning have better disentanglement properties and thus perform better than representations trained from correlation-based learning (c.f., Sec. 7.6.3).

### 7.5.4 Training the Hybrid World Model

For training the hybrid world model, our goal is to unify the semantic world model with a pre-existing latent world model. Such cases would arise when fine-tuning of the latent world model is prohibitive due to limited computational resources or would lose favorable generalization properties during the fine-tuning process. Thus, for training the hybrid world model in this work, the latent world model is frozen and not adapted; gradient updates therefore only affect the weights of the semantic world model.

To train the hybrid world model, three training objectives are used:

1. Minimizing the reconstruction error between the semantic observation $o_s$ and the reconstructed semantic observation $\tilde{o}_s$ (training the semantic encoder and semantic decoder);

2. Minimizing the KL divergence between the prior semantic state and the posterior semantic state (i.e., before and after updating the semantic state $s$ with an observation $o_s$);

3. Minimizing the KL divergence between the next semantic state from the semantic and latent sides, i.e., between $\bar{s}^{t+1}$ and $\hat{s}^{t+1}$.

Note that the semantic blending module $B_s$ is not used during training, as blending only occurs at inference time.

## 7.6 Experiments

To assess the capability of LMeshNet, we conduct experiments in learning a hybrid causal world model for a multi-body particle dynamics domain.

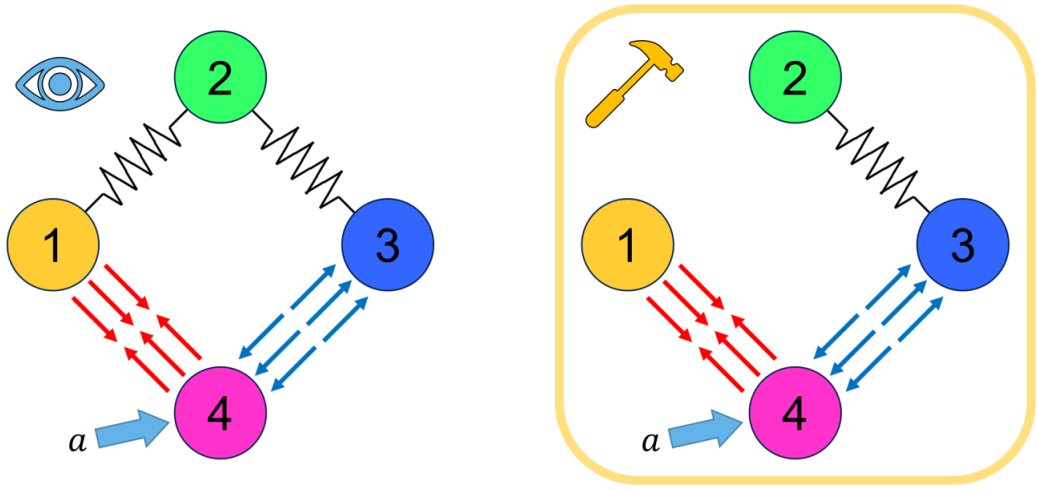
### 7.6.1 Multi-Body Dynamics Domain

As introduced by Lei et al. [209] and illustrated in Fig. 7.4, this domain consists of four particles that interact with one another based on various interparticle forces, such as spring force, attraction, and repulsion. The action $a \in \mathbb{R}^2$ applies a force to Particle 4 in the $x$- and $y$-direction. The domain consists of an observational setting and 18 interventional settings, where various interventions are applied to the observational setting. Such interventions include removing springs, changing particle masses, increasing the spring force constant, and constraining particle movement horizontally or vertically. The underlying causal variables in this problem can be represented as a ground truth state $c \in \mathbb{R}^8$, corresponding to the $x$- and $y$-position of each of the four particles. The system observations either are *images*, where an RGB image of the scene is captured, or *mixed-state*, where the 8 causal variables are passed through a mixing matrix $M \in \mathbb{R}^{12\times 8}$ to yield a 12-vector observation (i.e., $o = Mc$).

### 7.6.2 Experimental Setup

The hybrid causal world model is trained to unify a semantic space to a pretrained latent causal world model that has been trained with VCD ("VCDHybrid"). We compare the performance of the hybrid causal world model against a hybrid world model that adjoins a semantic space to a structure-agnostic (i.e., no specific causal training objectives) pretrained latent world model that uses a fully-connected RSSM transition module ("RSSMHybrid"). Additionally, for each hybrid world model, we explore the the effects of using a fully-connected latent-to-semantic prediction ("-FC") or using dimensions obtained from causal discovery ("-M") as described in Sec. 7.5.3. We assess these four models for two cases: when the world models operate independently (no latent-to-semantic blending) or when latent-to-semantic blending occurs ("-B").

For these experiments, the mixed-state observation space is used; examining image observations is left for future work. The semantic state of interest $s \in \mathbb{R}$ is Particle 1's $x$-position (i.e., $c_1$), which also serves as the semantic observation $o_s$. Neither the




(a) (b)

Figure 7.4: The multi-particle dynamics domain. The domain consists of four particles that interact according to various forces. The domain consists of either an observational setting with no interventions (a), or 18 interventional settings where various interventions are applied to the observational setting. The environment action $a$ applies a force to Particle 4. (a) In the observational setting (no interventions), Particles 1 and 2 are connected by a spring, as are Particles 2 and 3. Particles 1 and 4 are pulled towards each other through an attraction force, whereas Particles 3 and 4 are repelled from each other through a repulsion force. (b) In this example interventional setting, the spring connecting Particles 1 and 2 has been removed. Besides this intervention, all other environment mechanics are as defined in the observational setting. (Illustration is adapted from [209].)

mixed-state observation $o$ nor semantic observation $o_s$ have sensing noise. The six environments used for these experiments consisted of one observational setting with five interventional settings, consisting of 1) removing the spring between Particles 1 and 2 (Fig. 7.4b), 2) constraining Particle 2 horizontally, 3) constraining Particle 1 vertically, 4) increasing the mass of Particle 3, and 5) constraining Particle 4 vertically.

Experimental results are from evaluation on a test dataset of trajectories that are drawn from the same distribution as the training data. For the first 100 timesteps, the hybrid world models receive the observation $o$ and semantic observation $o_s$ at every timestep. Starting at the 100th timestep, no observations are used by the hybrid world models, assessing their capability for long-horizon predictions from only their environment model. Results are shown in terms of reconstruction error for both the latent and semantic spaces, by comparing the reconstructed observations $\tilde{o}$ and $\tilde{o}_s$ against their ground-truth observations $o$ and $o_s$. Synchronicity between the two halves of the hybrid world model is measured in terms of disagreement: the difference between the semantic state $s$ and the predicted semantic state $\hat{s}$ from the latent state

$z$ via $f_{z \to s}$. Representative time history trajectories are also presented.

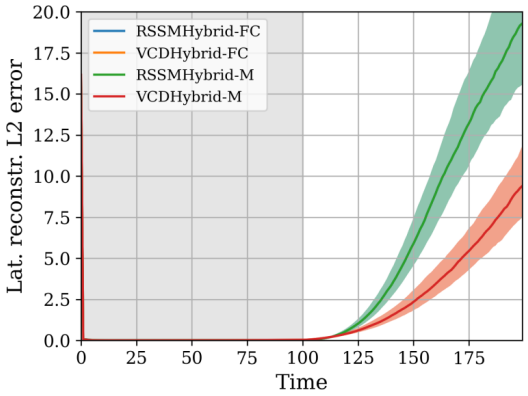
### 7.6.3 Experimental Results

**Independent (no latent-to-semantic blending).** Figure 7.5 shows the aggregate results of the hybrid world models when no latent-to-semantic blending occurs ($\beta = 0$) and the latent and semantic sides operate independently.

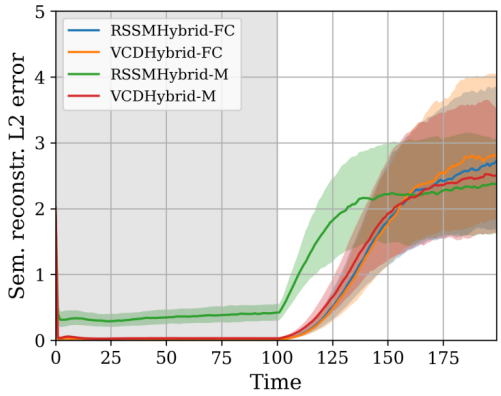
As shown in Fig. 7.5a, VCDHybrid models (VCHybrid-FC, VCDHybrid-M) provide lower reconstruction error for the latent space during long-term predictions than RSSMHybrid-FC and RSSMHybrid-M. This is specifically because of VCD's improved modeling of environmental dynamics through capturing the underlying causal factors. However, the reconstruction error grows over 100 timesteps as the latent space increasingly diverges from ground truth and long-term predictions continue to worsen as the duration since the last observation increases. Both types of models had similar performance while observations are used. Note that the choice of dimensions for the latent-to-semantic module $f_{z \to s}$ does not affect the reconstruction error for the latent space, as information does not flow into the latent side from the semantic side.

Figure 7.5b shows that three models (both VCHybrid models and RSSMHybrid-FC) have little semantic reconstruction error before the 100th timestep and similar rise in error afterward. As in the latent case, errors arising after 100 timesteps are due to the semantic state diverging from ground truth. However, unlike the latent case, Fig. 7.5b shows that the choice of $f_{z \to s}$ dimensions affects semantic reconstruction, even while observations are used. RSSMHybrid-M has a notable, consistent semantic reconstruction error before the 100th timestep; afterward, the error sharply increases before reaching a steady-state value similar to the other three models. The reason for RSSMHybrid-M's poorer semantic reconstruction error is likely because the latent space is more entangled than in the VCD case. This leads to causal discovery not capturing enough dimensions that would be needed for predicting the semantic space. Thus, when training the hybrid world model, the training objective to align $\bar{s}^{t+1}$ and $\hat{s}^{t+1}$ spoils learning by leading to a poorer and misaligned semantic representation. This does not occur in the RSSMHybrid-FC or VCDHybrid-FC case, as all dimensions are used. For the VCDHybrid-M model, the VCD latent representations have better causal disentanglement properties, so the downselection of dimensions sufficiently captures enough latent dimensions to reasonably predict the semantic state. Thus, the same reconstruction performance is achieved for VCDHybrid-M with fewer parameters than VCDHybrid-FC.

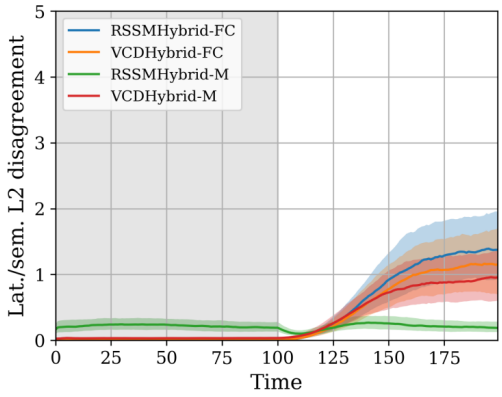
The synchronicity between the latent and semantic spaces (expressed in terms of disagreement) is shown in Fig. 7.5c. Generally, RSSMHybrid-FC, VCDHybrid-FC, and VCDHybrid-M have little disagreement prior to 100 timesteps, and afterward, the models start to diverge and desynchronize from the latent space. Statistically similar performance between the fully connected models is observed, whereas VCDHybrid-M




(a)




(b)




(c)

Figure 7.5: Hybrid world model performance when the latent and semantic halves operate independently in terms of (a) latent reconstruction L2 error, (b) semantic reconstruction L2 error, and (c) latent/semantic L2 disagreement. Results are aggregated across the test dataset and shown for each timestep. Shading indicates the interquartile range. At the 100th timestep, observations are no longer used by the hybrid world models, assessing their capability for long-term predictions.

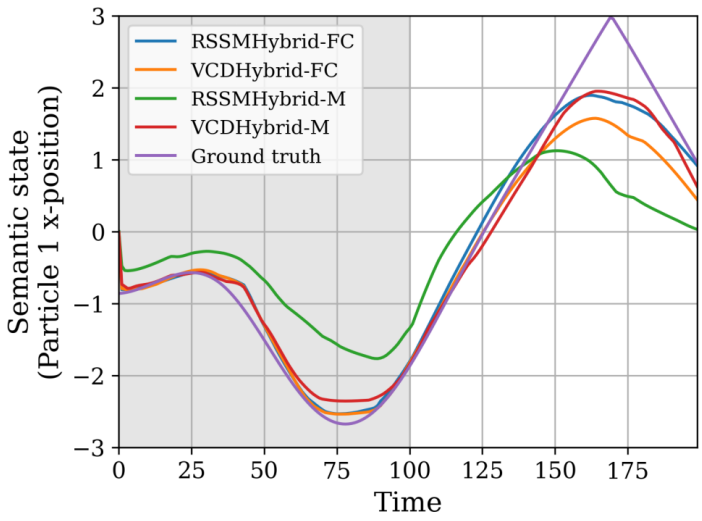
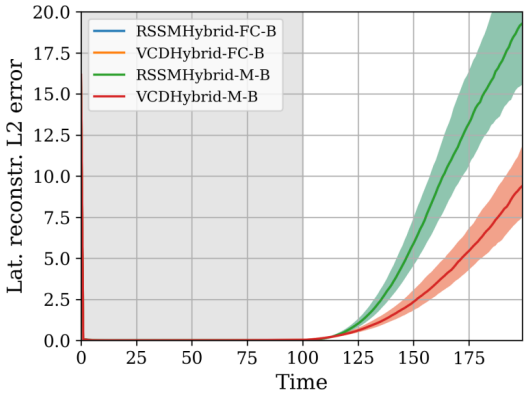



Figure 7.6: A representative time history of the semantic state $s$ for one test trajectory, with the ground truth observation also shown. The latent and semantic sides of each hybrid world model operate independently.

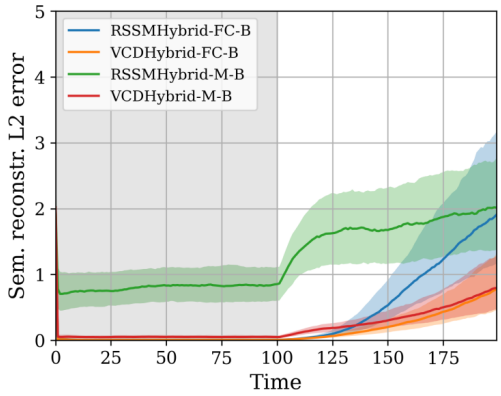
has slightly better disagreement than RSSMHybrid-FC. As in Fig. 7.5b, RSSMHybrid-M exhibits differing behavior. Prior to 100 timesteps, the consistent offset in disagreement is likely due to the same reason it has poorer reconstruction error in this regime. Although RSSMHybrid-M has the lowest disagreement after 100 timesteps, this model has the worst semantic reconstruction error. In this view, the latent and semantic spaces of RSSMHybrid-M are better synchronized with each other, but not with the ground truth.

Figure 7.6 shows the time history of the semantic state $s$ (i.e., Particle 1 $x$-position) for one representative trajectory, where the ground truth semantic observation $o_s$ is also shown. RSSMHybrid-FC and the VCD models have similar behavior in generally tracking $o_s$ prior to the 100th timestep, whereas RSSMHybrid-M has a notably worse error both before and after (similar to aggregate results in Fig. 7.5b). After 100 timesteps, RSSMHybrid-FC, VCDHybrid-FC, and VCDHybrid-M start to diverge from the ground truth around the 150th timestep, before reconverging near the end of the trajectory. As shown by the ground truth observation, this is just before Particle 1 collides into the boundary at $x = 3$ and reverses direction. This behavior is not well-captured by the semantic transition models $T_s$ of these world models, yet the predictions eventually reverse direction as desired. Although by Fig. 7.6 it appears that VCDHybrid-FC has slightly poorer error, in an aggregate sense (Fig. 7.5b), the three models have similar error characteristics and also outperform RSSMHybrid-M.

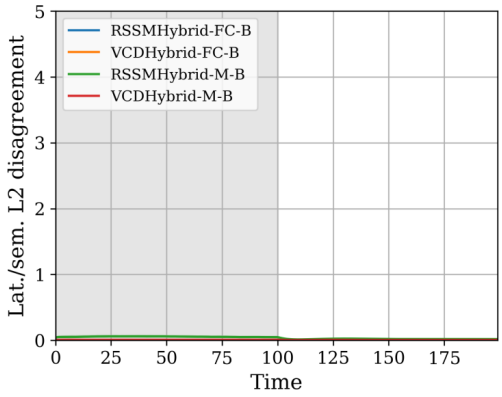
**Latent-to-semantic blending.** Figure 7.7 presents aggregate results for the hybrid world models when latent predictions are integrated into the semantic state.




(a)




(b)




(c)

Figure 7.7: Hybrid world model performance when latent-to-semantic blending occurs in terms of (a) latent reconstruction L2 error, (b) semantic reconstruction L2 error, and (c) latent/semantic L2 disagreement. Results are aggregated across the test dataset and shown for each timestep. Shading indicates the interquartile range. At the 100th timestep, observations are no longer used by the hybrid world models, assessing their capability for long-term predictions.

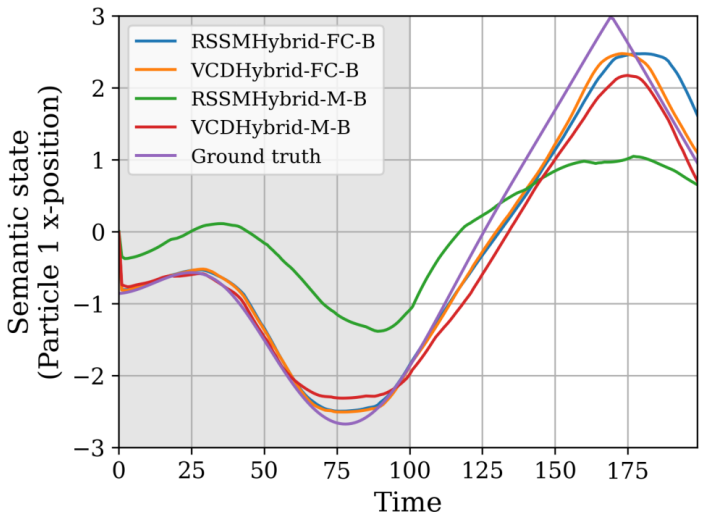



Figure 7.8: A representative time history of the semantic state $s$ for one test trajectory, with the ground truth observation also shown. Each hybrid world model uses latent-to-semantic blending.

A blending coefficient of $\beta = 0.5$ is used.

The latent reconstruction error is presented in Fig. 7.7a. Because information only flows from latent half into the semantic half, the latent space operates independently. Thus, results are identical to the case without blending (Fig. 7.5a).

Figure 7.7b shows the results for semantic reconstruction error. All models except RSSMHybrid-M-B experience a marked improvement of error during long-term predictions as compared to when the latent and semantic spaces operate independently. With blending, both VCDHybrid models now outperform RSSMHybrid-FC-B, where VCDHybrid-M-B is as performant as VCDHybrid-FC-B with fewer parameters. These blending results indicate that stabilization of the semantic space can be improved by integrating information from the latent side. This result implies that, during long-term predictions in these models, the latent transition model $T$ captures the environment better than the semantic transition model $T_s$ (or else, blending would increase error). For RSSMHybrid-M-B, blending prior to the 100th timestep yields an increase in reconstruction error, likely owing to poor learned representations for the semantic space as discussed previously. Compared to Fig. 7.5b, error is slightly improved when blending, but overall error rates lead to RSSMHybrid-M-B remaining the least performant model.

When blending, Figure 7.7c shows that synchronization of the two halves of the hybrid world model is achieved: disagreement is all but eliminated.

Figure 7.8 presents the time history of the semantic state for the same candidate trajectory as in Fig. 7.6. Compared to without blending, the long-term predictions have been improved, leading to closer predictions to the ground truth observation

from 150 - 200 timesteps.

**Summary.** The experimental results show that a hybrid causal world model that uses blending and masked $f_{z \to s}$ dimensions (VCDHybrid-M-B) is most performant, successfully tracking semantic information with fewer parameters than in the full-dimensional case. Moreover, masked $f_{z \to s}$ dimensions from causal discovery can be used for hybrid causal world models (i.e., VCDHybrid models), whereas this methodology appears incompatible with more entangled latent representations (e.g., RSSMHybrid models). Lastly, we find that blending improves semantic reconstruction error and eliminates disagreement during long-term predictions. Improvement in semantic reconstruction error occurs because long-term predictions in the latent space are more stable than in the semantic space. However, this outcome is likely specific to this domain and may not apply in general. Hypothetically, for domains where long-term predictions in the semantic space are more stable than in the latent space, latent-to-semantic blending may lead to a degradation in performance. In those domains, independent operation of both halves of the hybrid world model is the recommended design choice.

## 7.7 Conclusion

We present LMeshNet, an approach for creating a hybrid causal world model by unifying a semantic world model with a pretrained causal world model. Furthermore, for causal world models with latent representations with good disentanglement properties, causal discovery can elicit the shared dimensions between the latent and semantic spaces, leading to hybrid world models with fewer parameters and improved interpretability of the latent space. Lastly, we show that latent-to-semantic blending can fully synchronize both halves of the hybrid world model for this experimental domain.

For future work, we will investigate how richer semantic information can be imbued into the hybrid causal world model beyond tracking certain semantic states. For example, the semantic transition model could be given instead of learned, or structural causal models relating to known system behavior could also be integrated. Lastly, for domains where long-term predictions of the semantic space are more stable than the latent space, we will investigate semantic-to-latent blending, wherein information would now flow from the semantic world model to the latent world model.

## 7.8 Acknowledgments

We gratefully acknowledge Anson Lei for providing an official implementation of VCD.

# V

# Curriculum Learning

# 8 Automated Curriculum Learning: Humans and Agents

The aim of this thesis is to forge a path for causal robot learning, towards a grand vision of causal embodied intelligence. Considering that humans and animals are theorized to reason and learn through causal-based structure [238–242], we can derive insight into how to construct agents by understanding how humans approach problems. Such is the goal of Automated Curriculum Learning (ACL), a collaboration that seeks to compare and contrast human versus agent cognition for the problem of curriculum learning. Specifically, this study from ACL examines the differing approaches between 5- to 7-year old children and reinforcement learning agents in selecting levels to play in order to solve a challenging level of Procgen [243]. These children show contrasting learning capabilities as those required for curriculum learning with a reinforcement learning agent, which may experience catastrophic forgetting through domain shifts induced by the curriculum. However, we find that if the agent has an auxiliary reward that indicates competence, or progress, without necessarily solving the level (as perhaps the children have), catastrophic forgetting is generally mitigated or, if it occurs, recoverable. This insight has been critical towards the development of our curriculum learning algorithm, CURATE (Ch. 9).

We note that studying the intersection between human cognition and machine learning algorithms can offer benefits in both directions. As we show in ACL, we can derive clues from human cognition and problem solving for how to design more capable machine learning agents. However, the benefits of machine learning can also provide insights into human cognition. Whereas humans often behave rationally, we are not perfect and can often make mistakes. In this light, human intelligence provides an example of a complete, though perhaps not always optimal, intelligence (although human intelligence may take more forms than problem-solving capability, such as emotional intelligence). Conversely, there are tasks for which machine intelligence and powerful computational optimization are better suited and thus where algorithms can inform human decision-making, particularly for tasks where cognitive biases can affect our judgment. Thus, we advocate for more interest in the areas of psychology and machine learning, so that we may better understand intelligence in all its forms.

*Sections of the remainder of this chapter first appeared in [36]. Additional information is provided in App. E. This work was presented at the Intrinsically Motivated Open-ended Learning Workshop at the Thirty-seventh Conference on Neural Information Processing Systems (IMOL@NeurIPS 2023). Earlier versions of this work appeared in The 6th International Workshop on Intrinsically Motivated Open-ended Learning (IMOL 2023) [34] and the 2023 Interactive Causal Learning Conference (ICLC 2023) [35]. We thank our collaborators on this work: Annya Dahmani (first author), Eunice Yiu (first author), Nan Rosemary Ke, Prof. Oliver Kroemer, and Prof. Alison Gopnik. Annya Dahmani, Eunice Yiu, and Prof. Alison Gopnik are affiliated with the University of California, Berkeley. Rosemary Ke is affiliated with Google DeepMind. Human experiments in this chapter were conducted by Prof. Alison Gopnik's team and are separate from the contributions of this thesis.*

## 8.1 Summary

*We study how reinforcement learning algorithms and children develop a causal curriculum to achieve a challenging goal that is not solvable at first. Adopting the Procgen environments that include various challenging tasks, we found that 5- to 7-year-old children actively used their current level competence to determine their next step in the curriculum and made improvements to their performance during this process as a result. This suggests that children treat their level competence as an intrinsic reward, and are motivated to master easier levels in order to do better at the more difficult one, even without explicit reward. To evaluate RL agents, we exposed them to the same demanding Procgen environments as children and employed several curriculum learning methodologies. Our results demonstrate that RL agents that emulate children by incorporating level competence as an additional reward signal exhibit greater stability and are more likely to converge during training, compared to RL agents that are solely reliant on extrinsic reward signals for game-solving. Curriculum learning may also offer a significant reduction in the number of frames needed to solve a target environment. Taken together, our human-inspired findings suggest a potential path forward for addressing catastrophic forgetting or domain shift during curriculum learning in RL agents.*

## 8.2 Introduction

Humans are exceptionally remarkable learners, especially when they are faced with challenging tasks. We possess the capacity to craft personalized curricula that shape our experiences in ways that maximize our acquisition of new knowledge and skills [244–252]. Similarly, reinforcement learning (RL) agents also rely on curriculum-based learning to accomplish challenging tasks [253–257].

When designing a curriculum, it is essential to strike a balance between exploitation (leveraging existing skills and information for rewards) and exploration (discovering new skills and information to enhance decision-making). Studies have demonstrated that humans begin to master this balance between exploration and exploitation from early childhood [244, 258–261]. The question arises: How do humans learn to reason about their own learned capabilities and use this information to bootstrap the future knowledge they need to learn to address their current limitations? Causal learning may be crucial for enabling us to efficiently explore various levels of task difficulty and complexity within an environment [182, 262, 263]. In particular, first mastering the causal relations that are involved in a simpler task can allow agents to solve more complex tasks that involve similar relations [25, 182, 255, 262, 263]. This ability contrasts with the abilities of even the most advanced RL agents. We use causal inference and monitor our competence and learning progress to help guide our exploration, rather than randomly varying policies and observing the results. Causal models are well-designed precisely to afford a wide range of novel actions and interventions on the world [17, 19, 264, 265]. The ability to collect data from causal interventions can allow an agent to construct a new causal model, leverage that model to make decisions, and repeat this process for improvement. This may be key to improving the performance of RL agents in the future [25, 182, 262, 263]. See Appendix E.1 for related works.

Machines may likewise benefit from structuring their learning through a causal curriculum, improving the speed of convergence and boosting generalization through sequencing training data, developing hierarchical causal models and self-assessment [253–257, 266]. Curriculum learning [253–255, 266, 267] is relevant and beneficial for a broad range of applications from computer vision [266, 268, 269] and natural language processing [270, 271] to reinforcement learning [256, 257, 272]. For instance, successful learning in neural networks has resulted from a curriculum that starts small [273]. In the case of RL, while the advantages of improving learning progress [274] through a curriculum are generally recognized [257, 275], it is still unclear how to develop or select a curriculum in the automated and spontaneous way that humans do [267]. Tasks are often specified a priori from human domain knowledge, and it is not necessarily clear in what sequence the tasks should be visited within a curriculum.

Our research poses a crucial question that applies to both humans and RL agents: When we approach a goal that is too complex and challenging to achieve outright, how do we evaluate our existing skills and knowledge, and then craft a curriculum of more manageable "sub-tasks"? This curated sequence of sub-tasks serves as a means to construct a causal model, ultimately aiding us in reaching the ultimate goal. This line of work bears interests on both RL and cognitive science. On the cognitive science side, we seek to understand how children autonomously develop a curriculum to attain a goal that is difficult to accomplish at first. In particular, we hypothesize that children are intrinsically motivated to monitor their competence and proceed to acquire higher levels of skill accordingly, even without explicit rewards, and find

evidence that this is true. By putting children and RL agents on a level playing field, we also provide a benchmark and point of comparison for human curriculum learning against curriculum learning in RL agents. On the RL front, our initial goal is to evaluate RL agents through a predefined curriculum inspired by recent research [272]. We subsequently use this as a baseline against which we assess RL agents that include level competence as an auxiliary reward, inspired by our results with children. Our findings indicate that incorporating this reward significantly enhances the pace of learning and improves convergence. We discover something new about children; that is, we show that children use intrinsic rewards based on level competence. We then show that designing RL agents in a similar way leads to great progress and improvement. We outline the definitions of the terms we used in the paper here in Table 8.1:

| Term | Definition |
|---|---|
| Level Competence | An indicator of success in an episode of game play on a specific level, reaching 100% when the specific level is successfully solved in that episode |
| Global Competence | A measure of success in an episode of game play with respect to the target level, reaching 100% if the target level is successfully solved in that episode |
| Level Advancement | Difference between current level competence and initial level competence within a particular level in the curriculum; it is a measure of how much competence increases or decreases in the same level |
| Global Advancement | Difference between current global competence and initial global competence across levels in the curriculum; it is a measure of how much competence increases or decreases with respect to achieving the target level |
| Auxiliary Reward | An internal reward used by the RL agent that augments the extrinisic reward |

Table 8.1: Definitions of the terms we used in this paper

## 8.3 Automated Curriculum Learning in Children

Procgen, as introduced by [243], represents a procedurally generated environment that develops a wide range of RL games with varying levels of difficulty. To systematically analyze curriculum learning for both human players and RL agents, we selected a subset of Procgen environments and tailored them by adjusting game difficulty along a single parameter or variable. Our experiments focused on three distinct

games: Leaper, Climber, and Heist (Appendix E.2, Figure E.2). We ask how children scaffold their own learning to solve a difficult level of a game. Our primary focus is to explore whether children employ a systematic and rational approach in constructing a learning curriculum for themselves. To assess this, we observe children engaging with Procgen games [243] of varying difficulty levels and analyze their automated decisions in curriculum development.

### 8.3.1 Methods

We have gathered data from a cohort of 22 children (10 females, 12 males), with a mean age of 5.55 years ($\sigma = 0.74$ years) at public museums in the Bay Area, California, USA. Participants were told that they would be rewarded with a sticker if they won a target game level. The target level was set to be too challenging to win in a single attempt - it required some form of multi-step, curriculum-like learning. Children were then given the opportunity to autonomously select different difficulty levels of the game (Levels 1-4, with Level 3 being the target game level). Children could play either until they won the target level, or for up to a total of 10 rounds, whichever came first (see Appendix E.3.2 for details).

### 8.3.2 Automated Curriculum Learning Results

Overall, children made an average global advancement toward the goal of $\mu = 58.4\%$ with a standard error $SE = 11.8\%$ towards the goal (this is computed as the difference in competence between the initial attempt and the final attempt at the target level), which was significantly different from 0% ($t(15) = 4.96$, $p < .001$). This was even though they did not necessarily make positive level advancement between the first and last attempt at the same level ($t(17) = .29$, $p = .78$ in a one-sample t-test compared to $\mu = 0$). After failing the initial goal level, 72.7% participants started their curriculum by selecting an easier level: 9 chose Level 1, 7 chose Level 2, 5 chose Level 3, 1 (of the 7 presented with Level 4) chose Level 4 - the even more difficult level.

Next, we examined the change in levels selected by children across their automated curricula, given their competence on their current level (see Figure E.5 in Appendix E.3.3). A change in level of 0 indicates that the participant chose to remain on the same level, a change in level of -1 indicates that they chose the next easiest level, and so on. We found that children were more likely to remain on the same level or go to easier levels if their progress was low, and were more likely to move to more challenging levels if their competence was high. More specifically, level change (z-scored) was positively predicted by level competence (z-scored) in a linear mixed effects model with participant as a random effect, $\beta = .37$, $t(75) = 3.38$, $p < .01$. Thus, children, like adults in free exploration [251], used competence information to avoid overly easy tasks and advance to more difficult levels that were

closer to the target level in their curriculum. However, children neither used their global advancement (z-scored) to guide their level change in the curriculum (z-scored) ($\beta = .029$, $t(77) = -.26$, $p = .80$), nor did they use their level advancement (z-scored) to guide their level change (z-scored) ($\beta = .070$, $t(50) = .49$, $p = .63$). One possible reason is because children were not told to freely explore all levels, but were given the extrinsic goal of solving a difficult level (Level 3). Thus, selecting more difficult levels was appealing even when children did not make much level advancement. This is evident in Figure E.8 in Appendix E.3.3.

Furthermore, we also found that children demonstrated a causal understanding of their curriculum learning. We introduced 7 children to Level 4 which was even more difficult than the target Level 3. If children were selecting their curricula at random, all four levels should be equally likely to be selected, resulting in a 25% chance of selecting Level 4. However, children only selected Level 4 9.68% of the time. This suggests that children recognized that spending extra effort on Level 4 would not cause them to win a reward.

## 8.4 Hand-Designed Curriculum Learning in Reinforcement Learning Agents

### 8.4.1 Formulation

Reinforcement learning (RL) agents are trained to solve tasks modeled as a Markov Decision Process (MDP) [276, 277]. In this formulation, $M = \langle \mathcal{S}, \mathcal{A}, T, \mathcal{R}, \gamma \rangle$, where $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space, $T$ is the transition function, $R \in \mathcal{R} : \mathcal{S} \to \mathbb{R}$ is the reward function, and $\gamma$ is the discount factor. Within each MDP, the agent acts according to policy $\pi : \mathcal{S} \to \mathcal{A}$, which concerns low-level control. For this work, Proximal Policy Optimization (PPO) [156] is used to train the policy $\pi$.

During the training of the low-level control policy, the agent experiences different distributions of MDPs depending on the agent's curriculum. Specifically, the distribution of MDPs is provided by a high-level curriculum selection function $\phi : \Theta \to M$, where $\Theta$ are environment parameters that specifies the "difficulty" of the tasks. The goal of the agent is to learn a policy $\pi$ that can solve a complex target task $M_t$, analogous to needing to solve the goal level as in Sec. 8.3. Tasks are considered solved when the episode reward $R$ exceeds a predetermined threshold, $R_S$. For the purposes of monitoring agent learning, we assume there exists a function $\Lambda : \mathcal{S} \to \mathbb{R}$, called level competence, that describes how far the agent is into the task. Examples of level competence may include normalized distance traveled from a start position (relative to the total distance from the start position to the goal position) for a navigation task, or the fraction of items collected for a item collection task. Level competence generally holds values from 0 to 1, although negative competence is possible for tasks where negative progress can occur (e.g., increasing distance from the goal at the start

of a navigation task).

### 8.4.2 Methods

We assess curriculum learning in RL agents through a hand-designed curriculum on the Procgen game of Leaper. Task difficulty $\Theta$ is varied within the curriculum by parameterizing the number of water lanes, from 1 (easiest) to 5 (hardest). The goal level $M_t$ has 5 water lanes, consistent with Level 3 (Sec. 8.3). The agent starts at 1 water lane and advances to a more complex level (i.e., increased water lanes) when the training reward exceeds a threshold. The level competence $\Lambda$ for Leaper is the vertical lane that the agent has reached, normalized by the total number of lanes between the start and the finish line. We use the implementation and hyperparameters from [278] for training the policy $\pi$ using PPO. All experiments were run using an NVIDIA RTX 2080 GPU. Additional method details are available in Appendix E.4.1.

### 8.4.3 Baseline Curriculum Learning Results

We conduct six trials of training the agent using the hand-designed curriculum as specified by $\phi$. Representative results are summarized in Figs. E.9-E.10 in App. E.4.2. Overall, results are poor. None of the six trials successfully finished training. Five trials experienced training divergence, where the agent experiences catastrophic forgetting, leading to a permanent regression of reward to zero and a corresponding decrease in level competence. Figure E.9 shows a representative time history of an agent exhibiting training divergence by catastrophic forgetting. It has been shown that agents can experience catastrophic forgetting in continual learning problems with distribution shifts [279–282]. In our problem, our hand-designed curriculum induces intentional distribution shifts in order to train the agent in tasks of increasing difficulty. Behaviorally, when this divergence arises, the agent completely forgets the ability to cross lanes, thereby losing the reward signal in such a sparse rewards problem. Note that, for the representative result in Fig. E.9c, after divergence, the agent never exceeds 40% mean global competence, which is approximately where the first lane occurs in the distribution of tasks of this difficulty (4.25 water lanes). Without the reward signal, the agent does not receive feedback on which actions are good to take. Although for simpler environments it may be possible to recover, as the difficulty is smaller and random actions may find the goal, for harder environments with multiple lanes, this divergence is unrecoverable. Of the five trials that diverged, the average number of water lanes at time of divergence is 3.8625 lanes, with all five trials reaching at least 3 lanes. The remaining trial did not diverge, yet it only reached 2.6875 lanes.

Figure E.10 shows the relationship between mean episode training reward and mean episode level competence. Prior to training divergence, level competence is a proxy for reward, although the specific relationship depends on the difficulty of the

task.

### 8.4.4 Curriculum Learning with Auxiliary Rewards

Motivated by the human capability for learning (i.e., achieving competence) despite not necessarily completing a level, we conducted an additional experiment where the agent uses level competence as an auxiliary reward. As before, six trials were conducted. Specifically, the auxiliary reward function is $\mathcal{R}_i = 2\Lambda$, which is calculated at the termination of an episode. For this experiment, we assume the agent has access to level competence, leaving how it should be computed for future work. The auxiliary reward function $\mathcal{R}_i$ is defined such that the maximum auxiliary reward that could be obtained (when the level is solved and therefore $\Lambda$ is 1) is 20% of the extrinsic reward obtained when solving the level (10). Here, 20% is a heuristic that is chosen such that the effects of the auxiliary reward are smaller and less influential than the extrinsic reward, but large enough so that they still impact agent behavior. In this way, the extrinsic reward predominantly drives learning, with the auxiliary reward providing smaller, consistent feedback even when levels are not solved.

When training using auxiliary rewards, results were markedly improved. Five of the six trials were able to successfully finish training by solving the goal level via generalization. On average, generalization occurred at 3.353 million frames. Figure E.11 in App. E.4.2 shows a representative time history for training using level competence as an auxiliary reward where generalization occurred. Although the remaining trial did not solve the target environment, training reached the maximum number of frames allowed without experiencing divergence.

The experimental results with using an auxiliary reward for level competence suggest this prevents training divergence. We confirmed this by examining one particular trial, shown in Fig. E.12 in App. E.4.2, in which the agent *recovers* from catastrophic forgetting using the auxiliary reward of level competence. A difficulty level increase from 4 to 4.0625 water lanes around 4.141 million frames induces catastrophic forgetting, eventually leading to zero (extrinsic) reward and the complete loss of ability to cross lanes around 4.5 million frames. However, unlike the baseline experiments, the agent still obtains an auxiliary reward. Although the auxiliary reward decreases with level competence, it does not reach zero. Despite it being small, the learning signal is sufficient: the agent can still receive feedback for how to increase competence. This leads to a gradual restoration of agent capability, starting with an increase in level competence around 4.75 million frames. Eventually, this restoration yields a gradual increase of extrinsic reward, which is only obtained if the agent can solve the task. Although the restoration is not quick, taking about 1.5 million frames, the agent can nonetheless recover from catastrophic forgetting that would have otherwise led to training divergence.

### 8.4.5 Additional Baselines and Comparisons

In addition to Secs. 8.4.3 and 8.4.4, we conducted two additional experiments with a stochastic curriculum. Additionally, we conducted a random level baseline that is described in App. E.4.3.

**Stochastic curriculum: selection of random levels** In this experiment, 30% of the time, levels are chosen to be selected from the distribution of possible levels, from 1 water lane to 5 water lanes. In the remaining 70%, levels are determined based on the agent's current progress in the curriculum (as in Secs. 8.4.3 and 8.4.4). This experiment was conducted six times, and the agent was only trained using the extrinsic reward. In general, this strategy performs poorly. Four trials experienced training divergence. In only one trial was training still active at the end of the trial (getting to 3.0 water lanes). The average progression of the agent was 2.0313 water lanes, significantly less than in Sec. 8.4.3.

**Stochastic curriculum: selection of previously solved levels** This experiment is similar to the previous stochastic curriculum experiment, except that instead of selecting from any level, previously solved levels are chosen. This experiment is conducted six times with only the extrinsic reward being used. None of the six trials experienced catastrophic forgetting. However, the agent does not progress through the curriculum as quickly as with an auxiliary reward (Sec. 8.4.4). The average progression in the curriculum was 2.9583 water lanes. Of the six trials, none could solve the goal level, and only one trial reached 4 water lanes. Therefore, learning and leveraging level competence as an auxiliary reward appears promising as not only a way to prevent forgetting, but also to advance the curriculum.

## 8.5 General Discussion

**Level competence as an auxiliary reward signal to overcome catastrophic forgetting** While children are given the extrinsic reward of a sticker if and only if they solve the goal level of the game, they can visually observe their success in each game play and so assess their competence at each level. Seeing their competence allows for dense reward signals that are set by their performance. They further use this signal to determine the next step to take in the curriculum, leading to an average of 58.4% competence made towards the goal at the end of their automated curriculum learning. Similarly, level competence is a crucial and beneficial signal for RL agents. Our RL training demonstrates that exploiting level competence as an auxiliary reward signal reduces the odds of divergence and catastrophic forgetting.

**Future directions** Building on our finding that children employed level competence as a metric to assess advancements in curriculum-based learning, our RL experiments

incorporating level competence as an auxiliary signal yielded notably improved results. This compelling evidence underscores the pivotal role of level competence as an incentive to overcome distribution shifts induced by a curriculum. These findings suggest that extracting level competence information from high-dimensional image inputs and using it as a reward mechanism has the potential to significantly enhance the efficiency of RL agents in curriculum learning. In fact, it might potentially enable them to autonomously acquire proficiency in curriculum learning. In addition, we hope to further explore whether *automated* curriculum learning necessarily outperforms curriculum learning in a sequence based on strictly incremental difficulty or in a random sequence [283, 284] among both children and RL agents.

# 9 CURATE: Learning to Train Reinforcement Learning Policies through Curriculum Learning

In our previous works of CREST (Ch. 4) and SCALE (Ch. 5), we framed causal reasoning and learning through the lens of counterfactual scene interventions in simulation that answer queries such as "What would the reward be, had one of the context components been different and the robot still used the same control inputs?" However, finding an initial solution in order to perform causal reasoning required some initial random exploration. In both CREST and SCALE, the initial step of CREST requires a model-free policy search step, which was provided by REPS [152]. Additionally, for CREST, once the relevant variables are selected, the policy is trained using a representative model-free reinforcement learning algorithm, PPO [156]. The scene interventions used by CREST are only leveraged for policy feature selection and construction, not policy training. Can scene interventions be used more effectively to train model-free reinforcement learning policies by assessing the agent's current capabilities, while limiting expensive initial random exploration?

In CURATE, we address this question by investigating how curriculum learning can offer an *automatic curriculum* for training *any* reinforcement learning policy. We present our initial findings in this chapter, noting that our eventual goal is to imbue CURATE with greater structure-based capabilities, towards causal curriculum learning.

*This work is planned for submission as either a publication of a flagship conference on machine learning or robotics, or as a larger publication combined with ACL for a machine learning or cognitive psychology journal. We thank our collaborators on this work: Nan Rosemary Ke, Annya Dahmani and Eunice Yiu, Sarvesh Patil, Prof. Alison Gopnik, and Prof. Oliver Kroemer. Rosemary Ke is affiliated with Google DeepMind. Annya Dahmani, Eunice Yiu, and Prof. Alison Gopnik are affiliated with the University of California, Berkeley.*

## 9.1 Summary

*We present CURATE, an algorithm for automatic curriculum learning for reinforcement learning agents to solve a desired target task. Initially, due to the exploration-exploitation problem, agents may receive little useful feedback at the beginning of training, leading to inefficient learning. Through "exploration by exploitation," CURATE dynamically scales the task difficulty to match the current proficiency of the agent. Thus, the agent can more easily acquire initial behaviors to solve easier tasks before advancing to more challenging tasks. To maintain the curriculum, CURATE conducts sample-based interventions during training to determine a task distribution corresponding to the easiest tasks that the agent has not yet solved. As the agent's mastery grows, the task-directed curriculum offered by CURATE adapts correspondingly in an approximately easiest-to-hardest fashion, ultimately culminating in an agent that can solve the target task. Our initial experiments in grid-based navigation demonstrate that CURATE achieves greater curriculum learning performance than state-of-the-art algorithms that yield an implicit curriculum, as well as curriculum baselines such as a random curriculum and no curriculum. Although a hand-specified, easiest-to-hardest curriculum is slightly more performant than CURATE, we expect CURATE to achieve better performance in other domains where such a hand-specified curriculum either cannot be easily specified or is too simple.*

## 9.2 Introduction

The advent of reinforcement learning (RL) has ushered in a promising era of impressive milestones in sequential decision making by agents [277, 285–287]. The generality of RL has empowered advancements across many domains, including Atari [288, 289]; board games like chess, shogi, and Go [2]; StarCraft II [290]; and in the robotics domain, learning dexterous manipulation [291] and solving Rubik's cube [10]. Of the various desiderata for RL policies, *generalization* enables agents to succeed beyond its training data to solve test or target data. However, model-free reinforcement learning methods (i.e., those without inductive biases, models, or expert trajectories) are often sample inefficient particularly because of a fundamental tradeoff between exploration and exploitation that occurs at the beginning of training. Initially, the agent's actions are essentially random, requiring many environments interactions before the agent can start accruing reward and developing useful behaviors. For some domains with sparse rewards, the sample cost is significant, if not intractable. Thus, a benefit exists to improving the sample-efficiency of model-free RL algorithms (i.e., without providing privileged information that, while could improve efficiency, may be expensive or intractable to collect).

How can such a model-free agent bootstrap learning? Inspiration can be drawn from human intelligence, where a hallmark is the capability of reasoning with respect

to one's own learned capabilities, and using this information to bootstrap what future knowledge should be learned to address these limitations. How best to attain this future knowledge through sequencing learning objectives arises both from learner self-discovery of innovations and timely instructor interventions. This characteristic of human learning is evident in effective pedagogy, such as problem-based learning and assisted discovery learning [246–249].

As informed by work in cognitive neuroscience [273, 292], these motivations have informed the possibility of learning machines to also structure their learning in a similar manner. For machine learning, *curriculum learning* [253] seeks to answer how machine learning algorithms can learn training strategies to improve speed of convergence and boost generalization, where learning guidance comes in the form of sequencing training data and self-assessment. Curriculum learning is relevant for a broad range of applications, such as computer vision, natural language processing for text and speed, and reinforcement learning [269]. These fields all benefit from the two broad aims of curriculum learning [293]: to guide training (i.e., achieve greater learning sample efficiency) and to denoise training (i.e., improve learning robustness and generalization through focus on high-confidence training data regimes).

For reinforcement learning, the advantages of improving *learning progress* [274] through a curriculum are generally recognized [257, 294]. However, an automated way of selecting the curriculum (e.g., reinforcement learning tasks), remains an open-problem; tasks are often specified *a priori* from human domain knowledge. Furthermore, it is not necessarily clear in what sequence the tasks should be visited for the curriculum. Insight from evolutionary algorithms for open-ended learning [295, 296] suggest innovations can arise in a nonlinear, spontaneous fashion. However, for single-task RL, it has been argued that solving from an easy-to-hard fashion endows greater sample efficiency [272]. Additional concerns for sample-constrained domains, such as robotics, emerge if the costs of the curriculum itself outweigh the benefits given costly real-world robot samples [257].

Our ambition is to learn an *automatic, task-directed curriculum* for *any* model-free RL algorithm that can conduct evaluations across the distribution of tasks. Our insight is that the agent should learn to select its own tasks in order to solve a more complex target task, providing a way for the agent to attempt intermediate problems that are "just right" — not too challenging, but not too simple — so that behavioral innovations can more easily emerge that can bootstrap learning. To this end, we introduce CURATE (**Cur**riculum **A**gent for **T**argeted **E**xploration, Fig. 9.1), an algorithm for conducting sample-based evaluations for curriculum learning of a reinforcement learning policy that solves a target task. Our approach overcomes the fundamental exploration/exploitation tradeoff by conducting *exploration by exploitation*, as coined by [297], through sequential solving of easier tasks that bootstraps learning across particular *axes of learning generalization*. Our contributions are as follows:

- We introduce CURATE, a curriculum learning algorithm that learns which

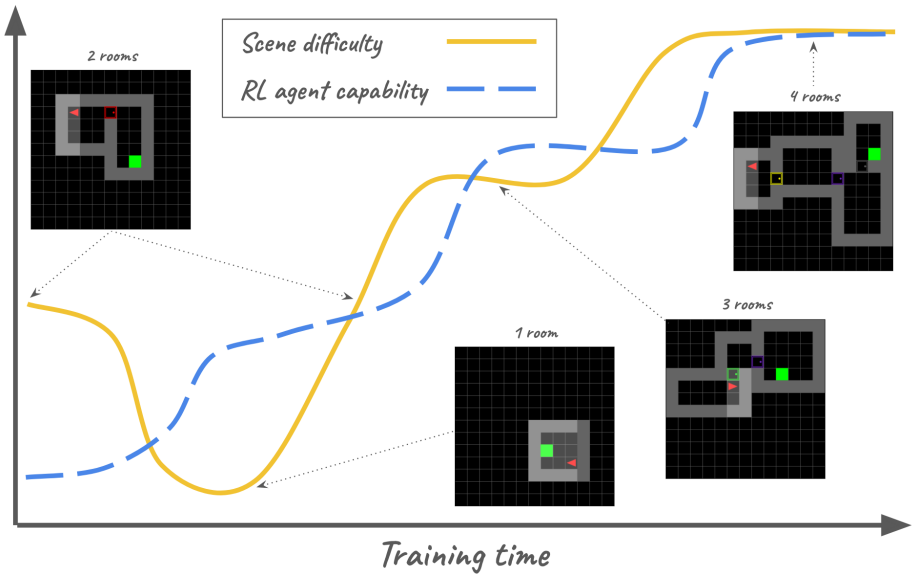



Figure 9.1: The CURATE algorithm learns to train a RL agent using an automatic curriculum provided by a curriculum agent. The curriculum agent manipulates the RL agent's training data by altering the difficulty of the scene. The RL agent's current capability is a measure of its performance in relatively more difficult scenes. In this visualization, the scenes offered by CURATE are initially too difficult, leading to a simplification of scenes. Once the RL agent begins solving these simple scenes, the curriculum agent dynamically adjusts the training data accordingly to harder scenes. Finally, the agent solves the target scene distribution at the end, indicating that training may conclude. Scenes are from the MiniGrid MultiRoom-N4-Random domain (c.f., Fig. 9.2).

tasks should be solved to improve learning progress for ultimately solving a target task.

- We formulate curriculum learning as conducting causal interventions within the agent's learning process, introducing a connection between these two fields.
- We conduct experiments that demonstrate CURATE achieves state-of-the-art performance using curriculum learning for RL policies. Currently, our experiments focus on the MiniGrid domain [298], a partially observable navigation task with discrete actions. We are currently planning to also conduct experiments for two additional domains: 1) Procgen [243], a suite of image-based games with discrete actions; and 2) IndustReal [299], a robot insertion task that requires highly precise continuous control.

## 9.3 Preliminaries

### 9.3.1 Learning with UPOMDPs

The agent learns within an Underspecified Partially Observable Markov Decision Processs (UPOMDP) framework as introduced by [300]. Essentially, the UPOMDP defines a *distribution* of Partially Observable Markov Decision Process (POMDP) tasks [301, 302] as determined by the selection of environment parameters. The UPOMDP is defined as follows:

$$\mathcal{M} = \langle \mathcal{A}, O, \Theta, \mathcal{S}^{\mathcal{M}}, T^{\mathcal{M}}, \mathcal{I}^{\mathcal{M}}, \mathcal{R}^{\mathcal{M}}, \gamma \rangle \tag{9.1}$$

where $a \in \mathcal{A} \subseteq \mathbb{R}^{|\mathcal{A}|}$ is a set of actions, $o \in O \subseteq \mathbb{R}^{|O|}$ is a set of observations, $\theta \in \Theta \subseteq \mathbb{R}^{|\Theta|}$ is a set of environment parameters, and $\gamma$ is a discount factor for future rewards. The remainder of the UPOMDP tuple is defined with respect to the chosen environment parameters $\theta$ and are thus superscripted by $\mathcal{M}$. Therefore, for the POMDP $\mathcal{M}$ specified by $\theta$, $s \in \mathcal{S} \times \Theta \subseteq \mathbb{R}^{|\mathcal{S}^{\mathcal{M}}|}$ is a set of states that are not observable to the agent, $T^{\mathcal{M}} : \mathcal{S} \times \mathcal{A} \times \Theta \to \mathcal{S}^{\mathcal{M}}$ defines the transition function, $\mathcal{I}^{\mathcal{M}} : \mathcal{S} \times \Theta \to O$ is the observation (i.e., introspection) function, and $R \in \mathcal{R}^{\mathcal{M}} : \mathcal{S} \times \Theta \to \mathbb{R}$ is the reward function. A task is considered solved if its reward exceeds a solved threshold $R_S$.

In principle, the generality of the UPOMDP framework allows a temporally-varying trajectory of environment parameters to be specified, but in practice, we are primarily concerned with environment parameters that specify the construction of the initial scene via the state space $\mathcal{S}^{\mathcal{M}}$. Other aspects of the UPOMDP, such as the transition, introspection, and reward functions, do not change in the domains we investigate in this work. In this view, our use of UPOMDPs is conceptually similar to Contextual MDPs [303–305].

### 9.3.2 Curriculum Learning within UPOMDPs

Our problem addresses automated curriculum learning for solving a particular target task that is initially challenging or impossible for the agent to complete. Curriculum learning is conducted across the *axes of learning generalization* of the UPOMDP, which is the space of environment parameters $\Theta$. The target task is defined by environment parameters $\theta_t \in \Theta$, and the POMDP specifying this target task is therefore $\mathcal{M}_{\theta_t}$. Our approach is an algorithm to decide how to select $\theta$ to efficiently train the agent by solving easier tasks in order to solve the target task $\mathcal{M}_{\theta_t}$. In this view, our problem uses the UPOMDP framework like other works in the Unsupervised Environment Design (UED) and Dual Curriculum Design (DCD) literature. However, our target task setting differs from state-of-the-art UED methods, insofar that our approach is less concerned with unsupervised discovery of environments within the environment space of the UPOMDP. In our work, traversing the space

of environments is a means to an end for solving the target environment within the UPOMDP.

## 9.4 Methodology

The goal of CURATE is to conduct curriculum learning to automatically train a control policy $\pi$ to complete a difficult target task $\mathcal{M}_t$. To do this, CURATE conducts sample-based evaluations to determine a curriculum policy $\pi_c$ that shapes the distribution of tasks used for the agent's sequencing of training data. In this way, curriculum learning is an active process in steering the agent's mastery of tasks, and changes to the curriculum can be seen as meta-level causal interventions within the agent training process. The curriculum policy $\pi_c$ is represented by a Gaussian distribution over environment parameters $\theta$ with mean $\mu_\theta$ and covariance $\Sigma_\theta$. This distribution is managed throughout training, adapting to the agent's capabilities. In cases where environment parameters take discrete values (e.g., parameters that refer to the number of rooms in a maze), the curriculum policy $\pi_c$ is obtained by discretizing the continuous distribution ($\mu_\theta$, $\Sigma_\theta$). The overview of the training procedure is presented in Alg. 9.1 (Sec. 9.4.1), which calls the UpdateCurriculum method (Alg. 9.2, Sec. 9.4.2) as a part of training. A description of algorithm hyperparameters is provided in Sec. 9.4.3.

### 9.4.1 Training RL Policies with Curriculum Learning

Algorithm 9.1 describes the training procedure. The control policy $\pi$ is initialized randomly via InitializeRandomPolicy. Initially, the curriculum policy $\pi_c$ provided by InitializeRandomCurriculumPolicy corresponds to a Gaussian distribution that approximates a uniform distribution over the environment parameter space. Then, the curriculum policy is updated prior to training with the (initially random) control policy $\pi$ via UpdateCurriculum. For each iteration in the training loop, tasks are sampled from the curriculum policy $\pi_c$ by first sampling environment parameters $\theta_i$, which are in turn transformed into tasks via TaskGenerator. TaskGenerator manages the creation of scenes of particular difficulty given by $\theta_i$. Then, RolloutAgentOnParallelTasks generates a dataset of trajectories $\mathcal{D}$ with mean training reward $R_\mathcal{D}$. The agent policy is next updated via UpdateAgent using this dataset, which handles the calculation of losses and gradient updates of policy parameters. Although in principle any on-policy reinforcement learning algorithm could be used, we use Proximal Policy Optimization (PPO) [156]. Following the policy update, the curriculum policy is updated if the training reward $R_\mathcal{D}$ exceeds the task solved threshold $R_S$. This trigger indicates that the agent has mastered proficiency in its current distribution of scenes and is ready for more challenging tasks. On this trigger, hyperparameter $\Delta\mu_\theta$ is used to help induce a new curriculum that

corresponds to harder tasks. Curriculum learning can also be triggered if a maximum number of frames since the last curriculum update has been reached, which is dictated by hyperparameter $\Delta f_{sync}$. This prevents training stagnation if CURATE selects tasks that are too difficult for the agent, eventually leading the curriculum to better synchronize with the agent's capabilities after some time. In either case, the covariance used for the update is specified by hyperparameter $\Sigma_{\theta_u}$. Finally, the agent is evaluated on the target task $\mathcal{M}_{\theta_t}$ via EvaluateAgent to obtain task evaluation reward $R_t$. We typically conduct stochastic, rather than deterministic, policy evaluation. If the agent solves the target task ($R_t > R_S$), then training concludes successfully. Otherwise, training continues while the number of maximum training frames $f_{max}$ has not been reached.

### 9.4.2 Updating the Curriculum with Sample-Based Evaluations

Besides at the beginning of training, CURATE's curriculum update is triggered when the task distribution seen in training exceeds the solved threshold $R_S$, or when a prespecified number of training frames has been reached without a curriculum update. CURATE's curriculum update, UpdateCurriculum, is a sample-based nonlinear optimization within environment parameter space. This method conducts sample-based evaluations to probe the current proficiency of the agent.

Algorithm 9.2 describes this method. First, the initial parameter distribution for the curriculum policy $\pi_c$ is provided by $\mu_{\theta_0}$ and $\Sigma_{\theta_0}$, which becomes $(\mu_\theta, \Sigma_\theta)$. Then, for each of $N_r$ rounds, the agent draws $N_s$ parameter samples from the curriculum policy parameter distribution $(\mu_\theta, \Sigma_\theta)$. For each parameter sample $\theta_j$, the corresponding task $\mathcal{M}_{\theta_j}$ is generated, and the agent is evaluated on this task to yield reward $R_j$. However, this reward is not used directly for the curriculum learning reward $R_c$. Instead, it is assessed whether it exceeds the threshold $R_S$, i.e., the task is solved. If so, the agent receives zero curriculum reward for this task, as the agent has mastered this task. Otherwise, the curriculum reward is assessed as $R_j/R_S$. This reward signal induces the agent towards the easiest (i.e., highest return) task that has not yet been solved. Thereafter, the curriculum learning reward $R_c$ is regularized by $\lambda_\theta||\theta_j||_2$. This regularization addresses cases where samples consistently return zero reward (e.g., at the beginning of training, all tasks may be too difficult), leading to a small bias towards easier levels. If this regularization did not exist, then the curriculum would remain close to a uniform distribution if all sampled tasks return zero reward. The parameter samples $\theta_j$ and curriculum reward $R_c$ are appended to buffers, which are eventually passed to REPSUpdate. This function uses Relative Entropy Policy Search [152] to yield an updated Gaussian distribution that maximises the reward, subject to a information loss bound based on Kullback-Leibler divergence. Then, for domains where tasks are discrete, the continuous curriculum parameters $(\mu_\theta, \Sigma_\theta)$ are

**Algorithm 9.1:** CURATE: Curriculum Agent for Targeted Exploration

**Input:** target task $\mathcal{M}_{\theta_t}$, task solved threshold $R_S$, maximum number of training frames $f_{max}$, number of parallelized tasks $N_v$, curriculum advancement on solve $\Delta\mu_\theta$, curriculum covariance for update $\Sigma_{\theta_u}$, maximum frames between curriculum updates $\Delta f_{sync}$

**Initialize:** training indicator $train \leftarrow$ `True`, target task solved indicator $converged \leftarrow$ `False`, number of training frames $f \leftarrow 0$, control policy $\pi \leftarrow$ InitializeRandomPolicy(), curriculum policy and parameters $\pi_c, \mu_\theta, \Sigma_\theta \leftarrow$ InitializeRandomCurriculumPolicy(), previous curriculum update frame $f_{prev} \leftarrow 0$

---

`// Initial curriculum policy update`
$\pi_c, \mu_\theta, \Sigma_\theta \leftarrow$ UpdateCurriculum$(\pi_c, \pi, \mu_\theta, \Sigma_\theta)$
**while** *train* **do**
  `// Sample tasks from the curriculum`
  $\boldsymbol{\mathcal{M}_{\theta_c}} \leftarrow \emptyset$
  **for** $i = 1$ **to** $N_v$ **do**
    $\theta_i \sim \pi_c$
    $\mathcal{M}_{\theta_i} \leftarrow$ TaskGenerator$(\theta_i)$
    $\boldsymbol{\mathcal{M}_{\theta_c}} \xleftarrow{+} \mathcal{M}_{\theta_i}$
  **end**
  `// Collect experience`
  $\mathcal{D}, R_\mathcal{D} \leftarrow$ RolloutAgentOnParallelTasks$(\pi, \boldsymbol{\mathcal{M}_{\theta_c}})$
  `// Update policy`
  $\pi \leftarrow$ UpdateAgent$(\pi, \mathcal{D})$
  `// Update curriculum policy`
  **if** $R_\mathcal{D} > R_S$ **then**
    $\pi_c, \mu_\theta, \Sigma_\theta \leftarrow$ UpdateCurriculum$(\pi_c, \pi, \mu_\theta + \Delta\mu_\theta, \Sigma_{\theta_u})$
    $f_{prev} \leftarrow f$
  **if** $(f - f_{prev}) > \Delta f_{sync}$ **then**
    $\pi_c, \mu_\theta, \Sigma_\theta \leftarrow$ UpdateCurriculum$(\pi_c, \pi, \mu_\theta, \Sigma_{\theta_u})$
    $f_{prev} \leftarrow f$
  `// Evaluate agent on target task`
  $R_t \leftarrow$ EvaluateAgent$(\pi, \mathcal{M}_{\theta_t})$
  `// Determine whether to continue training`
  **if** $R_t > R_S$ **then**
    $train \leftarrow$ `False`
    $converged \leftarrow$ `True`
  $f = f +$ NumFrames$(\mathcal{D})$
  **if** $f > f_{max}$ **then**
    $train \leftarrow$ `False`
**end**

---

**Result:** control policy $\pi$, target task solved indicator *converged*, number of frames $f$

discretized to yield the updated curriculum policy $\pi_c$.

**Algorithm 9.2:** UpdateCurriculum: Curriculum Update for CURATE

**Input:** curriculum policy $\pi_c$, control policy $\pi$, initial curriculum policy mean $\mu_{\theta_0}$, initial curriculum policy covariance $\Sigma_{\theta_0}$, task solved threshold $R_S$, parameter regularization $\lambda_\theta$, number of rounds $N_r$, samples per round $N_s$, relative entropy bound $\epsilon$, minimum temperature $\eta$

**Initialize:** $\mu_\theta \leftarrow \mu_{\theta_0}$, $\Sigma_\theta \leftarrow \Sigma_{\theta_0}$

**for** $i = 1$ **to** $N_r$ **do**
  `// Reset buffers`
  $\boldsymbol{\theta_{eval}} \leftarrow \emptyset$
  $\boldsymbol{R_{eval}} \leftarrow \emptyset$
  **for** $j = 1$ **to** $N_s$ **do**
    `// Sample task`
    $\theta_j \sim \mathcal{N}(\mu_\theta, \Sigma_\theta)$
    $\mathcal{M}_{\theta_j} \leftarrow \text{TaskGenerator}(\theta_j)$
    `// Evaluate agent on sampled task`
    $R_j \leftarrow \text{EvaluateAgent}(\pi, \mathcal{M}_{\theta_j})$
    `// Determine reward for curriculum learning`
    **if** $R_j < R_S$ **then**
      $R_c \leftarrow R_j / R_S$
    **else**
      $R_c \leftarrow 0$
    $R_c = R_c - \lambda_\theta ||\theta_j||_2$
    `// Append to buffers`
    $\boldsymbol{\theta_{eval}} \xleftarrow{+} \theta_j$
    $\boldsymbol{R_{eval}} \xleftarrow{+} R_c$
  **end**
  `// Run REPS and update curriculum policy`
  $\mu_\theta, \Sigma_\theta \leftarrow \text{REPSUpdate}(\boldsymbol{\theta_{eval}}, \boldsymbol{R_{eval}}, \epsilon, \eta)$
  $\pi_c \leftarrow \text{DiscretizeGaussian}(\mu_\theta, \Sigma_\theta)$
**end**

**Result:** updated curriculum policy $\pi_c$, updated curriculum policy mean $\mu_\theta$, updated curriculum policy covariance $\Sigma_\theta$

### 9.4.3 Description of Hyperparameters

The performance of the CURATE algorithm is influenced by the selection of several algorithm hyperparameters. We discuss these hyperparameters in detail below.

The task solved threshold $R_S$ is assumed to be known and given as a definition of the task. It represents the return that an agent who has mastered the task would receive. In general, we expect the amount of training frames that CURATE would require will vary based on the selection of this threshold. A higher threshold would require more training frames, whereas lowering the threshold would lead to fewer frames. In practice, this setting is chosen to ensure the examined baselines (c.f., Sec. 9.5.1) can generally solve the target task in a reasonable amount of frames. If this threshold is too high, then low-performing baselines may be unable to solve the target task. Additionally, we note that it is possible in principle for CURATE to have two separate thresholds: one threshold for the target task, and one threshold for all other tasks (an intermediate task threshold). We are currently exploring the effects of reducing the intermediate task threshold while keeping the target task threshold unchanged, which would tend to induce the agent to advance more quickly in the curriculum than it otherwise would have. However, we leave analysis of this extension for future work.

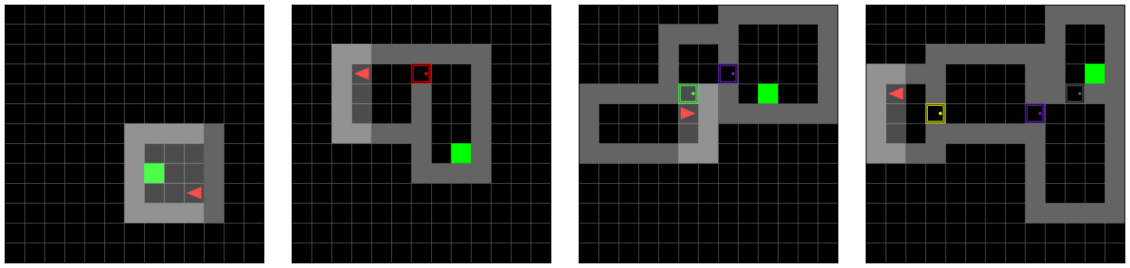
A curriculum learning update can be triggered if the task distribution is not solved in $\Delta f_{sync}$ frames. This hyperparameter prevents learning from stagnating indefinitely if the current task distribution is too difficult for the agent to make progress. For our initial experiments, this hyperparameter is set to run a curriculum learning update if the task distribution has not been solved in 128 PPO updates. In practice, this hyperparameter is most advantageous at the start of training, as it can update the curriculum if the initial task distribution is too difficult. Once the agent has reached some proficiency and can start solving tasks, then the curriculum learning update is more often triggered by the task distribution being solved. Setting the value of $\Delta f_{sync}$ too low generally means more curriculum learning updates will occur than necessary, which may increase algorithm runtime as the curriculum update is relatively expensive to run (in terms of time). Setting $\Delta f_{sync}$ too high, or disabling it altogether, may lead to undesirable training stagnation.

The parameter regularization hyperparameter $\lambda_\theta$ provides a bias towards less difficult tasks, under the assumption that task difficulty is proportional to environment parameters $\theta$. This capability induces a curriculum composed of easier tasks in the case that all tasks are too difficult for the agent and no reward is obtained. It is important that this hyperparameter is not too large, or else the curriculum reward $R_c$ would be offset by the regularization. In practice, we set $\lambda_\theta$ such that a maximum of 0.05 – 0.1 is reduced from $R_c$ for tasks with maximum $\theta$. For example, in a curriculum where the most difficult task occurs when $\theta$ is 4, setting $\lambda_\theta$ to 0.0125 would correspond to a maximum reduction of 0.05.

The task distribution is controlled in part by the initial setting of $\mu_\theta$ and $\Sigma_\theta$ prior to each curriculum update, as well as hyperparameters specific to REPS, such as the number of rounds $N_r$ and samples per round $N_s$. At the start of training prior to the initial curriculum learning update, the curriculum policy is initialized by the routine InitializeRandomCurriculumPolicy such that $(\mu_\theta, \Sigma_\theta)$ approximates a uniform

distribution. This approximate uniform distribution is determined by centering the initial mean through setting $\mu_\theta$ to $(\theta_{max} - \theta_{min})/2$ and choosing an initial covariance to spread out probability mass evenly. In practice, we find that setting this to $((\theta_{max} - \theta_{min})/3)^2$ suffices. For subsequent curriculum updates, the mean of the distribution begins at $\mu_\theta + \Delta\mu_\theta$ if the task distribution has been solved, or just $\mu_\theta$ if the update has been triggered by the $\Delta f_{sync}$ hyperparameter. However, the distribution covariance for subsequent curriculum updates always begins with a new covariance of $\Sigma_{\theta_u}$. The reason for this design choice (rather than using the existing $\Sigma_\theta$) is to ensure the task distribution does not completely collapse, as it may be beneficial to have a distribution of tasks for the agent to use for training. We generally set $\Sigma_{\theta_u}$ to be $((\theta_{max} - \theta_{min})/5)^2$, which was found to have acceptable task variance after a curriculum update with $N_r$ set to 2 and $N_s$ set to 8 for domains with few tasks, or 16 for domains with many tasks. For a given $\Sigma_{\theta_u}$, the task variance after a curriculum update will be lower if $N_r$ or $N_s$ is increased. Similarly, for a given $N_r$ and $N_s$, if a larger task variance is desired after the curriculum learning update, $\Sigma_{\theta_u}$ can be increased. Generally, it is desirable for the task distribution following the curriculum update to be centered around the easiest task not yet solved, with some variance to include nearby tasks. This may help with generalization and preventing catastrophic forgetting. However, if the variance is too large, then the effects of focusing on a small set of tasks will be lost. Lastly, we note that the curriculum learning update is a relatively expensive procedure. Thus, it is recommended to set $N_r$ and $N_s$ first, so that the curriculum update can take an acceptable amount of time, then size $\Sigma_{\theta_u}$ accordingly.

The curriculum advancement on solve hyperparameter $\Delta\mu_\theta$ is an optimization that can be used to bias the agent towards more difficult tasks after the current task distribution is solved. This hyperparameter shifts the initial parameter distribution $(\mu_\theta, \Sigma_\theta)$ at the start of the curriculum learning update. By default, this is set to 10% of the parameter space. The optimal selection of this parameter is likely task-dependent. If there is little difficulty change between adjacent tasks, the agent may be able to skip several intermediate tasks to find the next best task for learning. Thus, $\Delta\mu_\theta$ can be set to a relatively high value. Conversely, if a significant difficulty gap exists between adjacent tasks, then the next best task may be the task that immediately follows the task just solved. In this case, $\Delta\mu_\theta$ may be best set to either zero or the smallest parameter increment between tasks. We expect that this parameter has some tolerance and may not need to be set precisely, so long as it is in distribution with the optimal task. The curriculum update will ultimately settle on the best task distribution at the end of the curriculum update. However, if $\Delta\mu_\theta$ is set too high, then the curriculum update may fail to find the optimal task, thereby offering tasks that are too difficult to the agent. Should this occur, the agent's learning will likely stagnate until the curriculum can be synchronized if $\Delta f_{sync}$ frames occur without the task distribution being solved. As the curriculum shift $\Delta\mu_\theta$ only applies when the task distribution is solved, running a curriculum update from $\Delta f_{sync}$ may ultimately find the best tasks to use for training (although it may require multiple curriculum



(a) (b) (c) (d)

Figure 9.2: The MultiRoom-N4-Random domain [278]. This domain contains randomly generated corridors consisting of (a) 1 room, (b) 2 rooms, (c) 3 rooms, and (a) 4 rooms. To solve a scene, the agent must navigate from the first room to the goal in the last room. The size of each room is determined by randomly sampling two dimensions of length 4 – 7. The corridor is generated within a 13 by 13 grid. Navigating to an adjacent room requires the agent to open and proceed through the obstructing door. The target scenes for curriculum learning are the distribution of 4 rooms (d).

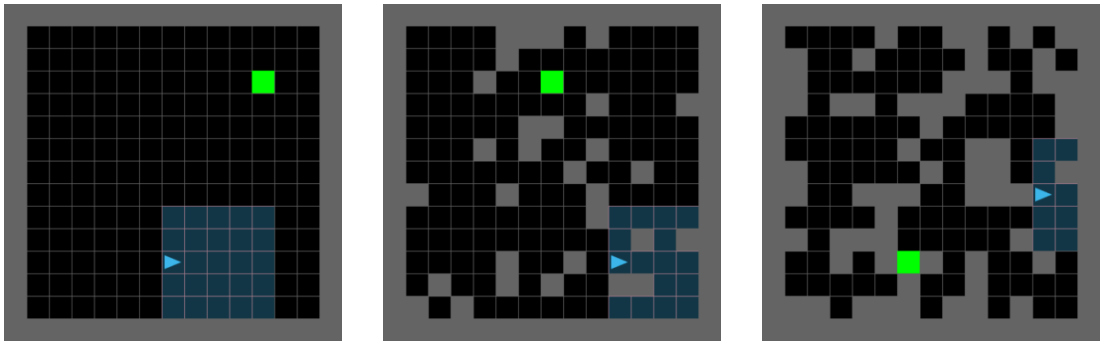
updates).

## 9.5 Experimental Results

We intend to evaluate how CURATE can yield automatic curriculum learning across a wide range of experimental domains for reinforcement learning agents and robots using both discrete and continuous action spaces. These domains include two MiniGrid domains [298], MultiRoom-N4-Random (Fig. 9.2) and ClutteredRoom-N60 (Fig. 9.3), where the agent must master grid-based navigation; three Procgen domains [243] (Fig. 9.4), where proficiency must be achieved over different procedurally generated games; and IndustReal [299] (Fig. 9.5), where a robot must demonstrate high-precision continuous control. At present, we present our experiments in MultiRoom-N4-Random (c.f., Sec. 9.5.2); the other domains are currently in progress.

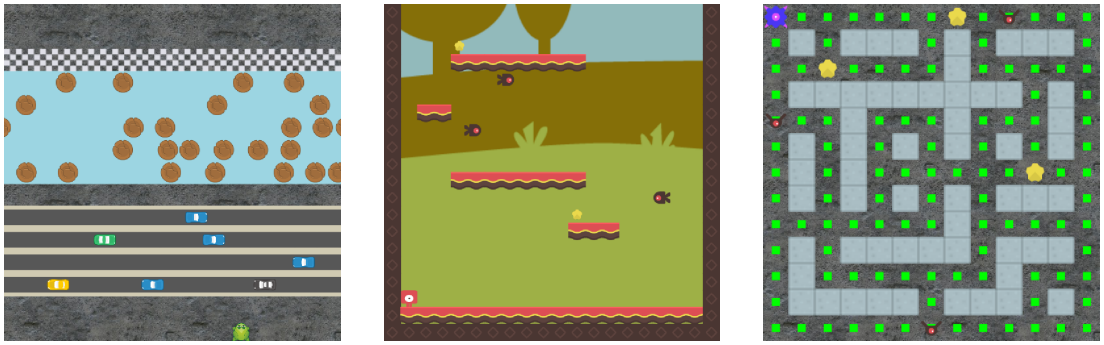
### 9.5.1 Experimental Procedure

Each domain features a curriculum learning problem, where the agent must determine how best to navigate learning within the space of tasks, as parameterized by environment parameters $\theta$. In general, increasing parameters $\theta$ corresponds to a more difficult task (and corresponding decrease in expected agent return). The ultimate



(a) (b) (c)

Figure 9.3: The ClutteredRoom-N60 domain [306]. This domain consists of a 15 by 15 room that contains a randomly selected number of blocks (from 0 to 60). Shown are representatives scenes consisting of (a) 0 blocks, (b) 30 blocks, and (c) 60 blocks. To solve a scene, the agent must navigate from the starting position to the goal. The target scenes for curriculum learning will be the distribution of rooms with 60 blocks (c).



(a) (b) (c)

Figure 9.4: The Procgen domain [243], which consists of procedurally generated games. (a) In Leaper, the agent must navigate from the bottom of the screen across road lanes and water lanes to the finish line. Curriculum learning will consist of varying the number of road lanes and water lanes. (b) In Climber, the agent must collect all the coins in the scene by jumping on platforms while avoiding enemies. Curriculum learning will consist of varying the number of platforms and number of enemies. (c) In Chaser, the agent must collect all of the orbs within the maze while avoiding enemies. If the agent consumes a large orb, the enemies can be defeated for a short period of time. Curriculum learning will consist of varying the maze size and the number of enemies.

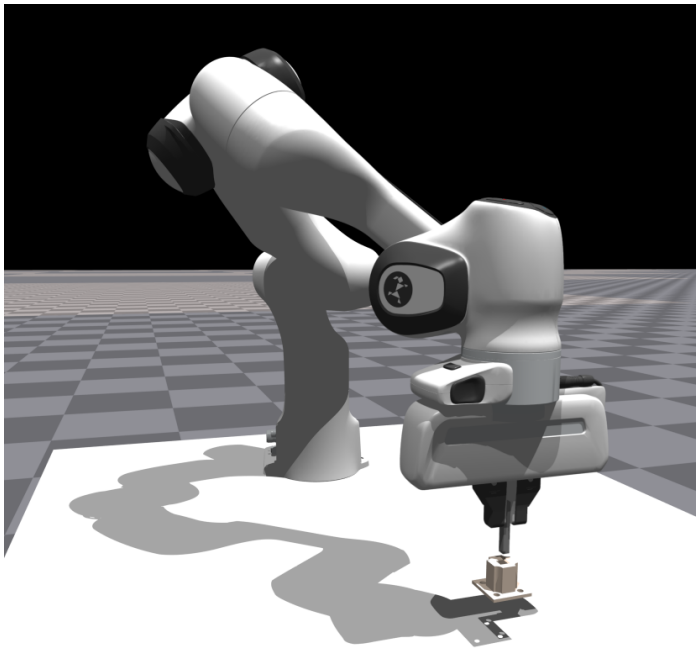


Figure 9.5: The IndustReal domain [299]. The robot must insert a peg into a socket, which requires highly precise control.

goal is to solve a particular target task $\mathcal{M}_{\theta_t}$ that is contained within the distribution of tasks within the domain. Usually, the target task is a distribution that contains the most difficult tasks within the domain. For each domain, we evaluate CURATE along with other UED algorithms and baselines.

**Training and test procedure.** All approaches use PPO [156] for training the control policy $\pi$. After every control policy update, the agent is evaluated on the target task. If the return achieved in the target task exceeds a predetermined threshold $R_S$, training concludes. Otherwise, training continues, up to a predetermined maximum allowable frames $f_{max}$.

**Baselines.** In total, we assess 6 approaches for each domain. Besides CURATE, we assess two UED algorithms, Robust PLR [307] and ACCEL [306], as well as curriculum baselines that include Domain Randomization, Hand Curriculum, and Target.

Robust PLR seeks to learn an implicit curriculum through judicious selection and replay of prior scenes through managing a replay buffer of tasks. Unlike in PLR [278], gradient updates only apply to replayed scenes in Robust PLR. ACCEL learns a curriculum through random mutations of tasks provided by the environment, starting from the most simple tasks. Task mutations are applied to the tasks contained in the replay buffer. Like Robust PLR, gradient updates only apply to replayed scenes.

Three curriculum learning baselines are also investigated. In Domain Randomiza-

tion, tasks are randomly sampled from the domain. In this way, Domain Randomization serves as a random curriculum. Hand Curriculum evaluates a straightforward curriculum, where the agent solves tasks in order of increasing difficulty. Once a task is solved above the solved threshold $R_S$, the agent advances to the next task. For some domains, Hand Curriculum can be seen as an approximate ground truth, but as shown in our previous work in ACL (Ch. 8), even a Hand Curriculum can suffer from training divergence when the agent changes tasks [36], and more sophisticated curriculum navigation may be necessary. Lastly, in Target, the agent only trains in the target task; no curriculum is used.

Finally, we note that although our work was greatly informed by Li et al. [272], which showed that solving tasks from easiest to hardest is optimal under certain conditions, we do not assess their algorithm, ROLLIN. This is primarily because of two reasons: 1) ROLLIN requires a near-optimal control policy initialization for solving the first task (such as an agent starting upon or immediately next to a goal), which is not the case in our domains; and 2) ROLLIN requires consecutive tasks to be sufficiently close, whereas our domains have some task-to-task differences that are not trivial (e.g., shifting from a maze of one room to two rooms requires learning how to open and proceed through the door that separates the rooms).

### 9.5.2 MiniGrid Corridor Navigation: MultiRoom-N4-Random

MultiRoom-N4-Random (Fig. 9.5.2) requires the agent to master grid-based navigation with partial observability. This domain was introduced by Jiang et al. [278] by extending a prior domain that was first introduced in MiniGrid [298]. In this domain, tasks consist of corridors of random numbers of rooms, from 1 room to 4 rooms. Each task is specified by the environmental parameter $\theta \in \{1, 2, 3, 4\}$ that specifies the number of rooms. The agent must navigate from the starting room to the goal, which is always contained in the last room. The target task $\mathcal{M}_{\theta_t}$ requires the agent to solve a distribution of corridors with 4 rooms. Specifically, the agent attempts 128 tasks using stochastic evaluation. For observations, the agent receives a low-dimensional encoding of the agent's viewpoint. This is a sparse reward setting, where the agent receives a time-discounted sparse reward upon solving the corridor, or zero otherwise. Each approach is evaluated over 10 trials with different random seeds. Generally, each PPO update occurs after 3072 frames, except in ACCEL, where 6144 frames occur before each PPO update when mutation occurs. All approaches have access to the full distribution of scenes (although some approaches, e.g., Robust PLR, only use a subset).

**Results.** Results for MultiRoom-N4-Random are shown in Tab. 9.1 and Figs. 9.6–9.7. Table 9.1 provides the summary statistics for each approach, where the corresponding sample efficiency (in terms of the median) is shown in Fig. 9.6. Represen-

| Approach | C. Type | Success Rate | Mean Frames ($\times 10^6$) | Median Frames ($\times 10^6$) |
|---|---|---|---|---|
| CURATE (ours) | Explicit | 100% | 6.114 ± 1.551 | 5.737 ± 2.006 |
| Robust PLR | Implicit | 100% | 18.280 ± 1.994 | 18.156 ± 3.190 |
| ACCEL | Implicit | 90% | 38.108 ± 10.122 | 34.265 ± 12.490 |
| Dom. Rand. | Random | 100% | 9.418 ± 1.806 | 9.136 ± 2.447 |
| Hand Curr. | Explicit | 100% | 4.750 ± 0.608 | 4.663 ± 0.726 |
| Target | None | 0% | 50.000 ± 0.000 | 50.000 ± 0.000 |

Table 9.1: Statistics for sample efficiency for MultiRoom-N4-Random. C. Type stands for curriculum type. 10 trials are evaluated for each approach. Mean Frames are shown with ± one standard deviation. Median Frames are shown with ± one interquartile range (IQR). Trials that do not solve the task still count towards summary statistics and are assessed the maximum allowable frames (50 million).

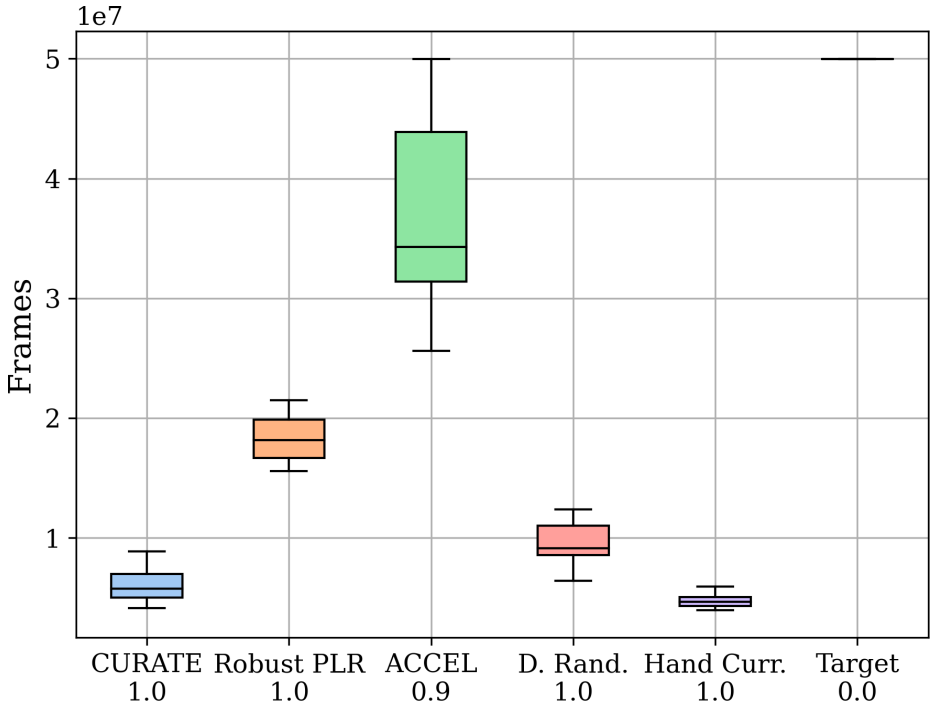
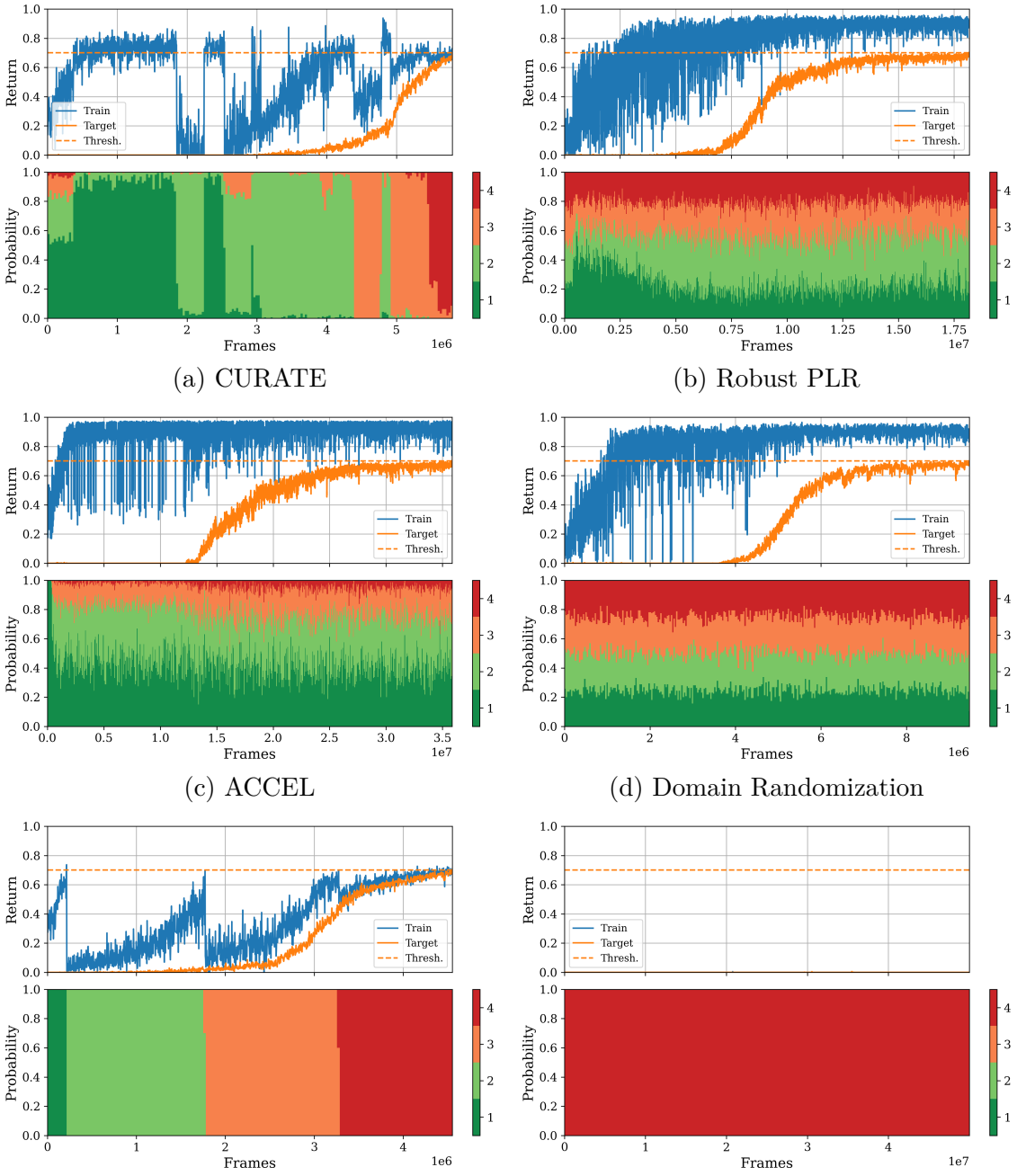



Figure 9.6: Median statistics for sample efficiency for MultiRoom-N4-Random. The approach success rate is displayed beneath each approach's name. D. Rand stands for Domain Randomization. Hand Curr. stands for Hand Curriculum. All trials for Target yielded the maximum allowable frames (50 million) with a 0% success rate.




(a) CURATE

(b) Robust PLR

(c) ACCEL

(d) Domain Randomization

(e) Hand Curriculum

(f) Target

Figure 9.7: Representative curriculum learning time histories for each approach. Each time history shows the trial that is closest to the median performance of all 10 trials for each approach. The top figure shows the time history of the return, shown for the training environments and the target task. The bottom figure shows the time history of the curriculum, with time discretization of 10 PPO updates to better show long-term trends.

tative trials for each approach are shown in Fig. 9.7.

In general, we see that CURATE outperforms all approaches except for Hand Curriculum. Figure 9.7a shows that CURATE tends to start curriculum learning in the easiest set of tasks, 1 room. Switching from 1 room to 2 rooms is a notable domain shift, requiring the agent to learn how to open and navigate through doors. Thus, the return tends to decrease upon switching, which may lead to a backwards step in the curriculum. However, generally, CURATE induces an easiest-to-hardest curriculum, leading to solving the target task when it has reached 4 rooms in the curriculum.

The performance gap between CURATE and the UED algorithms, Robust PLR and ACCEL, is relatively large. As shown in Fig. 9.7b and Fig. 9.7c, these algorithms do yield an increase in agent return, but such improvement is generally focused on the training environments. The increase in target task return is gradual, as there is otherwise no curriculum pressure for the agent to increase difficulty once the training reward is sufficiently great (above 0.8). Therefore, long-term training return is higher, but this does not necessarily yield gains to advancing the curriculum. There is evidence of an implicit curriculum that is learned in both Robust PLR and ACCEL, but such changes are not as task-directed as CURATE. ACCEL tends to yield a higher training return faster than Robust PLR, yet overall an easier curriculum (mostly 1 and 2 rooms), leading to longer generalization than Robust PLR.

Domain Randomization (Fig. 9.7d) provides a stronger performance than the UED algorithms, which likely is due to the relatively narrow possible values that the environment parameters can take. We also note that the improvement of Domain Randomization over Robust PLR is surprising; in Jiang et al. [278], PLR was shown to outperform Domain Randomization. We hypothesize that this difference is due to Domain Randomization using the entire distribution of scenes, whereas it and PLR were both constrained to 4,000 scenes in [278]. Robust PLR, ACCEL, and Domain Randomization all eventually achieve generalization, although indirectly (as compared to directly training in the same distribution as the target, as CURATE does.)

Hand Curriculum (Fig. 9.7e) performs the best out of all approaches, although it is just slightly more performant than CURATE. Its function is similar to CURATE, and we hypothesize is that CURATE tends to switch tasks more often, leading to greater accumulated frames than in Hand Curriculum. Like CURATE, generalization occurs directly when the training environment consists of the same distribution as the target task. Although this type of curriculum is successful in this domain, we do not expect that, in general, Hand Curriculum would always be ideal. In cases where training divergence may occur when switching tasks, a hand curriculum may suffer catastrophic forgetting, requiring more complex ways of managing distribution shifts than what a straightforward curriculum can provide.

Lastly, as shown in Fig. 9.7f, Target represents the performance without using a curriculum. Overall, performance is markedly poor: no trials were successful. The target task is too difficult to solve initially due to the exploration problem that agents

face when initially solving a task. CURATE addresses this exploration problem by dynamically changing the task to be more simple, leading to success early that can be bootstrapped into solving harder tasks.

## 9.6 Related Works

**Curriculum Learning.** As formalized by Bengio et al. [253], curriculum learning concerns how to meaningfully organize data for training machine learning models. The effectiveness of introducing concepts in an orderly fashion has support from cognitive psychology and machine learning research [273], although related work in the intersection of these fields has found differing evidence [292]. Therefore, as suggested by Bengio et al. [253], "some curriculum strategies work better than others," and thus how to optimally sequence training data for a given problem remains an open area of research. Thus, a diversity of approaches exist.

Graves et al. [267] introduce a general curriculum learning method based on a non-stationary multi-armed bandit algorithm, leading to a stochastic curriculum. Wang et al. [295, 296] show that curricula can emerge from co-evolving environments and agents. Portelas et al. [308] introduce a Gaussian mixture model in the parameter space of the environment, where the curriculum is driven by absolute learning progress. Algorithms from the Unsupervised Environment Design [300] and Dual Curriculum Design [307] frameworks yield implicit curricula that emerge from unsupervised learning. For the case of reinforcement learning, Li et al. [272] recently proposed that, under certain assumptions, solving tasks from easiest to hardest is optimal. Our algorithm, CURATE, is most similar to Portelas et al. [308] and Li et al. [272]. CURATE also maintains a distribution within the environment parameter space similar to Portelas et al. [308], but the curricula found by CURATE are driven by seeking out the easiest set of tasks that are not yet solved, leading to an approximately easiest-to-hardest curriculum that is similar to Li et al. [272].

**Unsupervised Environment Design and Dual Curriculum Design.** First introduced by Dennis et al. [300], the Unsupervised Environment Design (UED) paradigm provides a framework wherein parameters of an underspecified environment are varied by a teacher to produce distributions over environments for a student learner. This paradigm can support various teaching modes, such as domain randomization, minimax regret, or a "environment-generating adversary" [300] in the PAIRED algorithm, which learns to construct scenes adversarially to maximize the regret between the student learning and an antagonist who is allied with the adversary. Jiang et al. [307] unifies the UED framework with prior work on replaying experiences [278] to form the Dual Curriculum Design (DCD) framework, wherein the student learns from either the environment-generating teacher (as in UED) or from replaying past experiences. In so doing, Jiang et al. introduced REPAIRED (replay-augmented PAIRED) and an extension of PLR, Robust PLR (also stylized as

PLR$^{\perp}$), in which gradient updates only occur on replayed scenes, leading to improved theoretical guarantees and empirical performance. Later, Parker et al. [306] introduce ACCEL, an evolutionary-based algorithm that randomly mutates scenes starting from environments of minimal complexity. The first multi-agent extension of UED was introduced by Samvelyan et al. [309], which also proposed an algorithm (MAESTRO) for two-player, zero-sum settings. Mediratta et al. [310] revisited PAIRED to improve its performance through different mechanisms, such as a bonus term to prevent entropy collapse, different teacher optimizers, and behavior cloning. Recently, Beukman et al. [311] examined that an adversary optimizing for minimax regret may lead to learning stagnation, leading to the introduction of the Bayesian level-perfect minimax regret objective and corresponding algorithm (ReMiDi) that overcomes this limitation. Our algorithm, CURATE, is framed within the UED and DCD frameworks, where the teacher designs levels that are at the leading edge of the student's capabilities as determined by feedback from the student through sample-based evaluations. Whereas UED and DCD approaches typically yield an implicit curriculum, our approach handles how to navigate an explicit curricula (that is, within environment parameter space).

## 9.7 Conclusion

We present CURATE, an approach for automatic curriculum learning that learns how to train reinforcement learning policies. Our work frames curriculum learning as causal in nature, with agent selection of tasks expressed as interventions in sequencing its own training data. CURATE navigates a curriculum by conducting sample-based evaluations to establish the best task distribution for training. In a grid-based navigation task, we demonstrate that CURATE outperforms recent state-of-the-art algorithms from Unsupervised Environment Design and Dual Curriculum Design as well as other curriculum baselines, with a hand curriculum being slightly more performant. Although the hand curriculum slightly outperforms CURATE, this general method of solving easiest-to-hardest is leveraged by CURATE, and we expect CURATE to be competitive with or outperform a hand curriculum in domains where training divergence via catastrophic forgetting is more likely.

### 9.7.1 Extensions

As work on CURATE is still ongoing, we anticipate several extensions in the near future. First, we intend to demonstrate CURATE on other domains, including ClutteredRoom-N60, Procgen, and IndustReal. We also will continue to explore extensions of CURATE to further integrate with ACL (Ch. 8) and interleave greater causal structure and principles. We outline a few possible directions below.

- In ACL, children tended to not wait until solving a task before switching to a

harder task. CURATE uses a task solved threshold for determining when tasks should be selected for curriculum learning. Usually this threshold is the same as the target task solved threshold, but we can explore using a slightly lower threshold that mimics human behavior.

- In ACL, children learned how to control the agent without necessarily solving tasks they selected for curriculum learning. This capability is in stark contrast to reinforcement learning agents in sparse reward settings, where the learning signal to determine how to learn is only achieved after solving a task to receive a reward. Thus, we are currently exploring a learned *progress model* that can bootstrap agent behavior with an auxiliary reward, which ultimately "densifies" learning. We also showed in ACL how, if such an auxiliary reward was available to use, training divergence via catastrophic forgetting was either mitigated or recoverable. We have explored the use of Explore Like Experts [312], which has been shown to improve learning using a dataset of expert trajectories in the domain of NetHack. For our purposes, we would extend this methodology by first using trajectories that solve the initial task distribution, then gradually fine-tuning and generalizing the progress model as CURATE advances the curriculum.

- What if the environment parameters $\theta$ contain dimensions that are not relevant for the target task? For example, consider a new parameter for MultiRoom-N4-Random that controls the color of the corridor, but generalizing over color is not necessary for solving the target task. For these cases, we can imbue CURATE with greater causal principles by conducting causal reasoning in concert with curriculum learning, where probing the underlying causal structure can assess the effect of each parameter on agent performance to learn over which parameter axes the agent should generalize. Such capability would be advantageous for curriculum learning in open-world settings, which may have many possible axes, although only a few may be relevant.

## 9.8 Acknowledgments

We gratefully acknowledge support from the Manufacturing Futures Institute and the NVIDIA NVAIL program. We also gratefully acknowledge Jake Bruce and Minqi Jiang for their helpful feedback regarding Explore Like Experts and Procgen integration into the DCD codebase, respectively.

# VI

# Conclusion

# 10 Conclusion

With the contributions discussed, we now provide the conclusion for the thesis. We first start with a discussion of lessons learned from the thesis (Sec. 10.1). Then, we identify future research directions (Sec. 10.2) and present limitations and next steps (Sec. 10.3). Lastly, in Sec. 10.4, we conclude the thesis with a vision that we hope this thesis will bring to bear.

## 10.1 Discussion

We now present major themes and lessons learned from the completed works of the thesis.

### 10.1.1 The Generality of Causality

Causality is a fundamental property of not only processes within our world, but of information. The generality of causality leads to rich opportunities in imbuing a wide range of robot learning problems with greater reasoning and learning capabilities. This is evidenced by the various contributions within this thesis that pertain to causality, such as learning and transferring the causal structure of policies and skills from simulation to reality (Part III, CREST and SCALE) and better understanding and modeling dynamical systems from a causal perspective (Part IV, LBD and LMeshNet). Our concluding work in curriculum learning (Part V, ACL and CURATE) is currently exploring greater causal capabilities, towards our goal of causal curriculum learning. Even for contributions that did not specifically leverage causal structure (Part II, DREAM and FormNet), these methods can be readily extended by incorporating causal learning objectives. This diversity also suggests a litany of other problems that would benefit from the integration of causality (c.f., Sec. 10.2).

### 10.1.2 The Power of Causal Interventions and Counterfactuals

In 2018, Pearl and Mackenzie introduced the "Ladder of Causation" [22], in which association, intervention, and counterfactuals are framed as increasing levels of cognition within a hierarchy. Such a framework corresponds to the human capability of seeing, doing, and imagining: critical aspects of learning that demand greater levels of cognition. Currently, state-of-the-art correlational robot learning generally consists of learning from observations: the first rung of the ladder. However, human-like capabilities may demand higher levels of cognition through interventions and counterfactuals, which leverage the causal structure of information. This thesis explores in part how robots can achieve greater performance through harnessing interventions, such as causal feature selection in a simulated version of reality (Part III, CREST and SCALE). These interventions can be seen as answering counterfactual queries such as "What would the reward be, had one of the context components been different and the robot still used the same control inputs?" We believe this work is an important step for how robots can answer counterfactuals in a general sense (i.e., as used by the causal inference literature) for problems with richer causal structures and interrelationships. In this way, this thesis pioneers a trail for how robots can climb the Ladder of Causation towards human-like cognition.

### 10.1.3 Causal Robot Learning: A Timely Need

As discussed in Ch. 1, the bifurcation of deep learning realities suggests that, for certain problems, state-of-the-art statistical machine learning may be sufficient for solving tasks to an acceptable degree. Despite the significant progress made in this area for robot learning, the ubiquitous deployment of robot manipulators into the open world currently remains out of reach. Even with the rise of foundation models [313], generative models trained with vast quantities of data at scale, certain learning questions may not be sufficiently answered without a causal lens. Thus, robots would benefit from learning to reason about the processes that generate the data, beyond just the data alone. This thesis argues that causal robot learning would expedite robots breaking this barrier and operating seamlessly within the open world, ultimately achieving the "promise of robotics." [185]

## 10.2 Future Work

The completed works of this thesis suggest future directions in integrating the principles of causality for a vast set of robot learning problems. Some example research thrusts are discussed below. These thrusts build towards a unified endeavor: building a lifelong, causal robot learning system.

### 10.2.1 Active Causal Learning of Robot Manipulation Skills through Interactive Perception

*How can interactive perception facilitate active causal learning of robot skills?* Modeling actions as interventions, the robot can conduct experiments to probe the underlying data generating processes to identify and learn their structure. In the context of learning manipulation skills, this thrust extends the work of SCALE (Ch. 5). Instead of learning skills from batch data as in SCALE, such interventions can render the discovery of skills that are present, yet statistically uncommon. This work would frame interactive perception [96] through a causal lens, wherein the robot conducts information-seeking actions to perceive and learn the structure of the data generating processes.

### 10.2.2 TRACE: Structural Task Transfer

*How can a new task be learned from the structure of a previously solved task?* For general-purpose robots to be useful, they must be equipped to quickly and sample-efficiently transfer previous relevant experiences in order to solve a new task. For example, the robot could transfer its previous knowledge for opening an oven to a new task with a similar structure, such as opening a cabinet. This thrust, TRACE,[1]generalizes our previous work in structural sim-to-real transfer for robot control (Part III, CREST and SCALE) for structural sim-to-real transfer of task representations using causal discovery. As shown in Fig. 10.1, TRACE would explore *structural task transfer*: how the learning of a task can be bootstrapped from transferring the causal structure of a related task that the robot knows how to solve. This thrust would consider the question of which tasks, out of a library of solved tasks, would be the best starting point for transfer, with the important consideration that relevant task structures could exist in either real-world or simulated experiences.

[1]TRACE simultaneously refers to two concepts. First, the verb "trace" means to find or discover, much like this approach empowers the robot to find the structural causal model that relates a desired task with a task it knows how to solve. The second concept is that of a "stack trace" of a computer program. The knowledge learned by the robot is anchored with respect to some previous structural causal model, similar to how a stack trace is referenced by the preceding stack frame. Moreover, the original project name — **S**tructural **TA**sk and **C**ausal **K**nowledge **TRA**nsfer through **C**ausal discov**E**ry (STACK TRACE) — is best shortened to TRACE for the sake of brevity. We thank Prof. Kun Zhang, Prof. Oliver Kroemer, Prof. Chris Atkeson, and Prof. Wennie Tabib for their insights and conversations related to this particular thrust of TRACE, which was initially proposed for investigation as a part of this thesis.

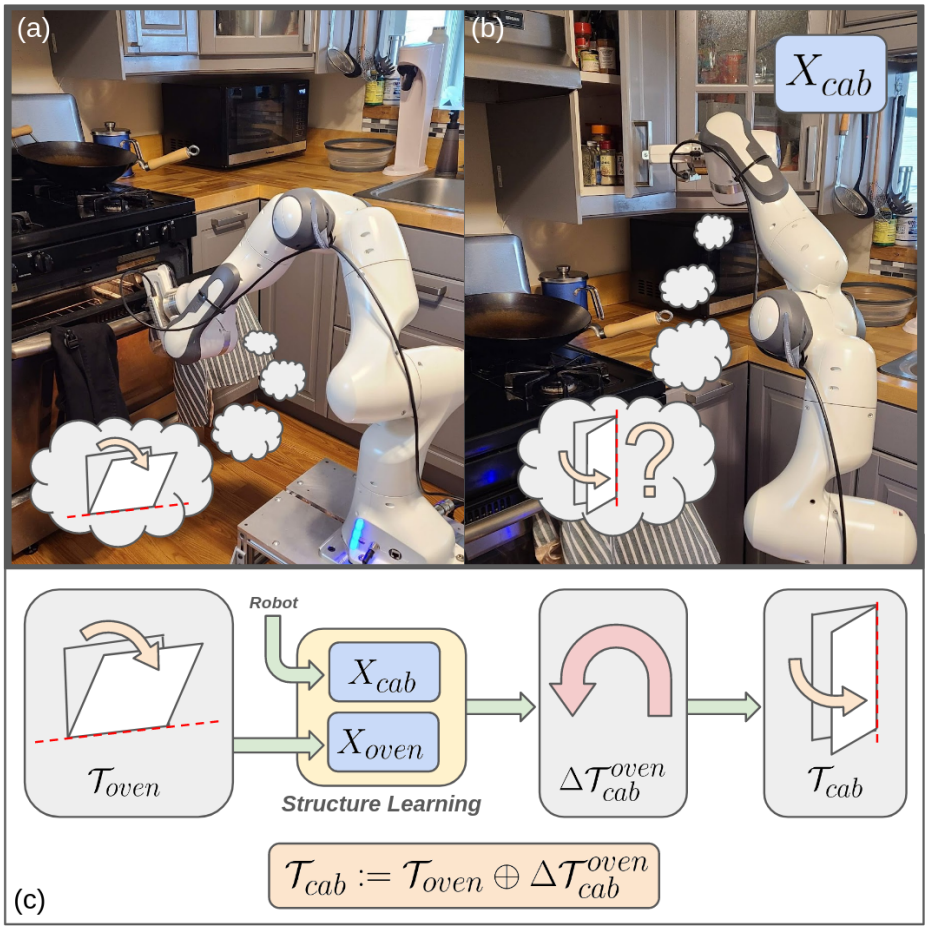



Figure 10.1: Proposed overview of TRACE. The robot starts with a structural task representation ($\mathcal{T}_{oven}$) for solving an oven opening task, shown in (a). The robot is now asked to complete a different, but related, task — opening a cabinet. As shown in (b), initially, the structural task representation of the cabinet task, $\mathcal{T}_{cab}$, is not known. To learn it, the robot collects in-scene data, $X_{cab}$, and then forms a dataset $\mathcal{D}$ that also includes $X_{oven}$, data generated by solving the oven opening task (either real-world data or synthetic data generated by an internal model). Using this dataset $\mathcal{D}$, structure learning discovers a structural model that relates the solution for opening a cabinet to the solution for opening an oven. This model is $\Delta\mathcal{T}_{cab}^{oven}$, where the notation expresses that this is a relationship defined between these two task representations. The task representation is then recovered: $\mathcal{T}_{cab} \coloneqq \mathcal{T}_{oven} \oplus \Delta\mathcal{T}_{cab}^{oven}$. The operator $\oplus$ represents the composition of these two task representations, analogous to how $SE(3)$ poses can be defined with respect to composition of other $SE(3)$ poses.

### 10.2.3 RVS: Causal Visual Servoing for Robust Image-Based Control

*How can causality endow robustness to* any *distribution shift in irrelevant parts of the image?* An important capability for robot manipulation is visual servoing [314, 315]: sequential direct control from images. By extending recent work in causal inference literature for learning counterfactually invariant predictors [316] and shortcut removal [317], the image can be decomposed into either regions that are causal or spurious to the image-based robot control task. Therefore, our goal for **R**obust **V**isual **S**ervoing (RVS)[2]is to achieve robustness to *any* distribution shift in irrelevant parts of the image. The key insight is that the image itself is produced by the data generation process of camera projective geometry. Whereas our previous work in LBD (Ch. 6) explored the similarities between control theory and causality for controlling a system, this thrust would explore similarities for *observing* a system.

### 10.2.4 Causal Discovery and Implicit Causal Models

*What are the implications for causal discovery with implicit representations of causal models?* Causal discovery, the process of learning causal structure from data, emerged in the context of learning causal models with graphical structure [27, 28]. These models can be considered *explicit* causal models, as their structure is explicitly encoded by the graph. Yet, causal models known to humans may be encoded *implicitly*: in a representation that does not require an obvious graph, where causal reasoning and counterfactual queries can nonetheless still be conducted. With such implicit representations of knowledge, what would be the role of causal discovery?[3] Is the underlying structure discovered prior to the model being learned, or refined afterward?

### 10.2.5 Causality and Human-Robot Interaction

*How can causal reasoning augment learning from humans?* The vision of a general-purpose robot is to operate seamlessly alongside humans in open-world environments, such as homes, restaurants, and hospitals. Given the presence of humans, approaches from human-robot interaction [318] would benefit from integrating causality. For example, if a human were to give a demonstration for completing a task, can the robot use causal reasoning to determine the relevant features of the demonstration for use in imitation learning [319]? Moreover, can the robot pose counterfactual queries in the form of natural language to the human demonstrator to avoid spurious

[2]We thank Joseph Gleason and Eric Dixon (Lockheed Martin Corporation) for their discussions and feedback for this particular thrust of RVS, which was briefly investigated during the course of this thesis.

[3]We thank Sarvesh Patil for his insights and conversations for this thrust.

correlations and understand preferences? Specifically, for a demonstration of putting pasta sauce away in a cabinet, the query "If the adjacent cabinet door was closed, would the demonstration still be the same?" would allow the human to communicate to the robot whether changes to that part of the environment would be causally important.

### 10.2.6 Unifying Foundation Models with Causality

*How can generative foundation models be unified with model-based causal reasoning?* Despite the impressive capabilities of today's *foundation models* [313], generative models trained on broad data, they are generally opaque mechanisms [320, 321]. For the case of large language models (LLMs), these generative natural language models have shown state-of-the-art performance on multiple causal benchmarks in causal discovery, counterfactual reasoning, and understanding necessary and sufficient causes [322]. Yet, others suggest that LLMs may only be learning "correlations between causal facts in natural language," rendering such models nothing more than "causal parrots" [323]. On the other hand, structural causal models are inherently interpretable and offer model-based explanations [16]. Indeed, in the words of Pearl and Mackenzie [22]:

> "...if we ever want robots to answer "Why?" questions or even understand what they mean, we must equip them with a causal model..." (Pearl and Mackenzie [22])

This thrust would seek to unify foundation models with a causality model, where each side complements the other. For example, a foundation model could generate rich, counterfactual scenarios that are grounded by modeling assumptions and mechanisms provided by the causality model. In this way, such a unified learning system would be akin to the lateralization of the human brain, which works harmoniously as one system with the popular mythos of left-side propensity for logic and order and right-side specialization for intuition and creativity [324, 325].

### 10.2.7 CASIE: The Lifelong, Causal Robot Learning System

*What are the implications of causal learning over the operational lifetime of an embodied system?* There are important questions for embodied systems continuously learning over their operational lifetime: What tasks are important to learn? When is learning no longer needed? Which knowledge concepts would be important for learning, and where would they be found? Such questions are *causal* in nature, for they concern reasoning over data generating processes (beyond just data). This systems thrust would leverage all previous thrusts to develop CASIE (**C**ausally **A**ugmented

**S**ystem **I**n **E**mbodiment),[4] a robot that works alongside humans to learn a variety of useful manipulation tasks over an extended operational lifetime. This thrust would explore not only the integration of causal robot learning algorithms and capabilities into one holistic system, but also their embodiment in hardware.

## 10.3 Limitations and Next Steps

In this thesis, we argue for causal robot learning: the integration of the principles of causal inference and causal representation learning into robot learning. Moreover, we posit that the capabilities demonstrated by causal-based learning algorithms would benefit robots in open-world settings, such as reasoning over which factors are relevant and exhibiting robustness to distribution shifts. For the individual contributions of this thesis, we focused on demonstrating capabilities for representative tasks that could be tested within a laboratory setting, while making assumptions for capabilities that could be available in the future. For example, in CREST (Ch. 4), we demonstrated that policies learned and constructed from causal structure would create networks that scale with only the relevant variables, and we assessed this capability for a block stacking task with many distractor blocks. Among some of the assumptions for this work, we assumed a simulator was available that the robot could use to perform causal interventions. Although the assumptions made throughout our works may be strong, we believe these works offer compelling evidence towards further research in this area and the relaxation of these assumptions.

To future researchers who wish to continue the ideas of this thesis, we recommend pursuing experimentation in open-world environments, either in reality or in photo-realistic simulation. Towards this end, we believe that the development of CASIE (Sec. 10.2.7), the causal robot learning system, would serve as a "North Star" towards this end. The construction and realization of such a system would address autonomous structuring of learning problems in the open world and better identify which assumptions must be addressed towards the realization of such a system in practice.

## 10.4 Towards Causal Embodied Intelligence

In closing, we leave the reader with a vision that extends well past this thesis: a vision for causal embodied intelligence. In this third age of artificial intelligence (AI), it is remarkable that, as Roy et al. [15] state, many of the challenges facing robot learning have not changed in 30 years. Indeed, as Brooks argues,

[4]CASIE's name is gender inclusive, so that everyone, regardless of gender, may identify with them.

> "Just about every successful deployment of AI has either one of two expedients: It has a person somewhere in the loop, or the cost of failure, should the system blunder, is very low." (Brooks [1])

This "human-in-the-loop" aspect is fundamental towards unlocking the next age of AI. In this age of deep representation learning, impressive progress has been made. However, as Schölkopf et al. [25] argue, this progress has mostly been a consequence of engineering away nuisance factors until the particular problem can be mostly solved via modern statistical learning algorithms on suitably independent and identically distributed data.

In this sense, we have made little progress towards removing the human from the autonomy loop. Whereas in the second age of AI, where humans imbued systems with domain expertise in the form of rule-based models, humans now imbue systems with domain expertise through neural network architectural choices and training data. Moreover, when the agent performs inconsistently with the human designer's intent, the human intervenes, updates the autonomy (model or data), and tries again. This closes the autonomy feedback loop for imbuing agents with domain expertise until the particular problem can be solved by the agent, without human intervention.

We posit that the key towards the next age of AI is addressing this "human-in-the-loop" problem. Instead of humans performing interventions upon the robot, the robot must perform *self-interventions* in order to achieve *causal self-learning*. Doing so requires some notion of the robot assessing what it knows, what it needs to learn, testing hypotheses through real-world experiments, and integrating knowledge gained from causal discovery. Without such principles of causal learning, it is not immediately clear how a robot's knowledge base could ever exceed what a human domain expert knows. It is the vision of this thesis that robots can ultimately *learn from experiments* for *any* task. Yet, these ideas have broader implications for not only robot manipulators, but for *any* embodied agent. In fact, a fully realized *causal, self-learning embodied intelligence* many in fact behave in a manner that could be commonly described as having Artificial General Intelligence (AGI).

To be clear: it will take a concerted, herculean effort across many decades of researchers across many fields — computer science, machine and robot learning, causal inference, statistics, cognitive psychology and neuroscience, human-robot interaction, vision, sensing, planning, control, mechanical and electrical engineering, natural language and linguistics, philosophy, AI ethics, and many more — to solve the "human-in-the-loop" problem once and for all. That being said, even if this thesis is a small step towards this lofty goal, the thesis author will be forever grateful for the contributions that arose because of the collaborators of these thesis works and other researchers in this field.

# VII

# Appendices

# A Appendix for FormNet

## A.1 Algorithm for Computing Articulation from Motion Residual Flow

**Algorithm A.1:** Compute Articulation Type and Parameters from Predicted Motion Residual Flow

**Input:** Depth image $I_D \in \mathbb{R}^{W \times H}$, binary part segmentation mask $I_S \in \mathbb{Z}_2^{W \times H}$, motion residual flow $I_F \in \mathbb{R}^{W \times H \times 3}$, small thresholds $\epsilon_0, \epsilon_1$.

**Output:** Articulation Type (AT) $\in$ {FIXED, PRISM, REV} and articulation parameters if PRISM or REV.

**if** $\|I_f\|_2 < \epsilon_0$ **then**
  **return** FIXED
**end if**
Set original point cloud $P \leftarrow$ DEPROJECT$(I_D[I_S])$
Set estimated displaced point cloud $P' \leftarrow P + I_F[I_S]$
Set pre-motion plane and normals $\pi, \hat{n} \leftarrow$ RANSAC$(P)$
Set post-motion plane $\pi', \hat{n}' \leftarrow$ RANSAC$(P')$
**if** $\hat{n}^\top \hat{n}' > 1 - \epsilon_1$ **then**
  Find mean flow $d \leftarrow \frac{1}{\sum_{w,h} I_S[w,h]} \sum_{w,h} I_F[w,h]$
  Normalize into direction $\hat{d} \leftarrow \frac{d}{\|d\|_2}$
  **return** PRISM, $\hat{d}$
**else**
  Find intersecting line $\mathbf{l} \leftarrow$ INTERSECT$(\pi, \pi')$
  **return** REV, $\mathbf{l}$
**end if**

## A.2 Summary of Public Datasets of Meshes

| ***Dataset*** | ***Categories*** | ***Objects*** | ***Info*** |
|---|---|---|---|
| **RBO** [120] | 14 | 14 | Y |
| **ShapeNet** [121] | 3315 | 220K | N |
| **PartNet** [122] | 24 | 26.6K | N |
| **Shape2Motion** [110] | 45 | 2.4K | Y |
| **PartNet-Mobility** [109] | 46 | 2.3K | Y |

Table A.1: This table summarizes the different public datasets of meshes on their number of object categories, number of object models, and whether it contains articulation information between object parts. Column **info** represents articulation information (y/n).

## A.3 Extension of FormNet Performance on Object Categories

See Figure A.1 for an extension of Figure 3.4 with all the object categories.

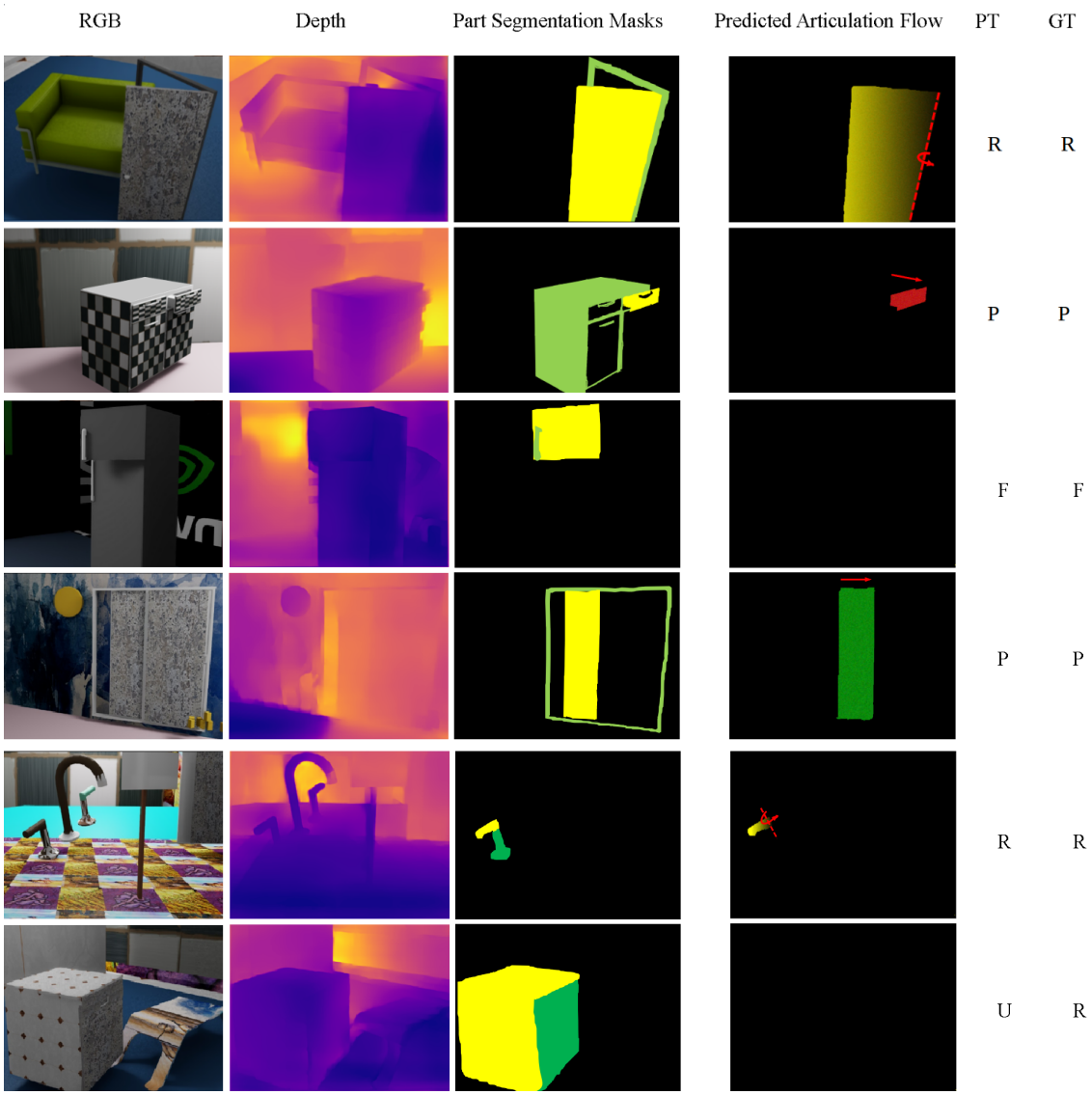



Figure A.1: Extension of Figure 3.4 with all object categories.

# B Appendix for CREST

## B.1 Summary of CREST

Which state features are important for learning a control policy? Our approach, CREST, addresses this question through causal feature selection. CREST selects the relevant state variables for a given control policy, which apply over the policy's preconditions. The assumptions for CREST are that an internal model (i.e., an approximate task simulation) exists, the context space representation of the internal model facilitates causal interventions (e.g., disentangled variables), and the (parameterized) control policy and its preconditions are known. Through structure and transfer learning, CREST enables learning of policies that are compact, avoiding unnecessary state features. By construction, policies built using CREST are robust to distribution shifts in irrelevant variables, whereas baseline methods may yield policies with spurious correlations that are brittle. Such distribution shifts could arise from transfer between the internal model and reality, due to variations in dynamics or context distributions not encountered during pretraining with the internal model.

## B.2 CREST Analysis on Math Environment

We now provide a greater description of the manipulation environment described in Sec. 4.5.3. The toy environment, `MathManipEnv`, approximates the mathematics of a controller for goal-based manipulation. For simplicity, the low-level control policy simply perturbs the state $s \in \mathbb{R}^{|S|}$ by an input of $\theta = a \in \mathbb{R}^{|A|}$ in a manner specific to whether the system is linear or non-linear. Additionally, we consider the context $c \in \mathbb{R}^{|S|}$ to be the initial state, $s_0$. For this evaluation, we considered cases where $|S| = |A|$ ("Dim." in Table 4.1).

The reward for this task is

$$\begin{aligned} r &= -\|\boldsymbol{g}_a - \boldsymbol{g}_d\| \\ &= -\|Gs_a - \boldsymbol{g}_d\| \\ &= -\|G(s_0 + A(\theta)) - \boldsymbol{g}_d\| \end{aligned}$$

where $\boldsymbol{g}_a \in \mathbb{R}^{|g|}$ is the goal vector that was obtained after execution of the controller to yield achieved state $s_a$, and $\boldsymbol{g}_d \in \mathbb{R}^{|g|}$ is the desired goal. The goal vector is calculated from a *goal selection matrix* $G \in \mathbb{R}^{|g| \times |S|}$, which is a one-hot encoding matrix where the columns indicate the elements of the state vector that are used. In practice, $G$ is formed by first randomly selecting $N_\tau$ relevant context variables from the total set of $c$ to form $\tau$. Then, each $\tau$ is randomly allocated to a separate dimension of the goal vector, i.e., row of $G$. Here, $G$ represents that, in some goal-based problems, the goal is calculated from only a subset of the state vector (e.g., relative to the position of a particular object).

The process of the system is either linear or non-linear, where $A(\theta) = \Delta s + w_a$ and $w_a \sim \mathcal{N}(0, \sigma_a^2)$ is the action noise. In the linear case, the controller $A \in \mathbb{R}^{|S| \times |S|}$ is a matrix with randomly selected coefficients, so $\Delta s = A\theta$. The non-zero coefficients of $A$ indicate mappings of $\tau_j$ to $\theta_j$. In the non-linear case, the controller $A$ is a list of size $|A|$, where each element of the list specifies randomly selected functions (exponential, sigmoid, sine, cosine) that transform input $a_j$ into the resulting $\Delta s_j$.

Each trial of this environment randomly selects different $\tau$, $\boldsymbol{g}_d$, $s_0$, $G$, and $A$. The goal vector dimensionality $|g|$ is fixed for each run and is typically equal to $|S|$.

## B.3 Task Representation: Block Stacking

We now provide additional detail for the block stacking task described in Sec. 4.7.1. Figure B.1 illustrates some context variables and the policy trajectory. In this task, the robot must stack the source block (block 0) upon the target block (block 1) using a sequential straight-line skill with control policy $\pi(a|s, \theta_b)$ and known preconditions.

**Policy.** The policy parameters $\theta_b = [\, \theta_{\Delta x},\, \theta_{\Delta y},\, \theta_{\Delta z} \,]^{\mathrm{T}} \in \mathbb{R}^3$ define three waypoints that the robot is sequentially commanded to via impedance control. Specifically, let $y_p$ represent a vertical position above the table and blocks. After the block is grasped, the executed policy is therefore:

1. Vertically lift to $y_p$.
2. Move $(\theta_{\Delta x}, \theta_{\Delta z})$ at fixed $y_p$.
3. Vertically move $\theta_{\Delta y} - y_p$.

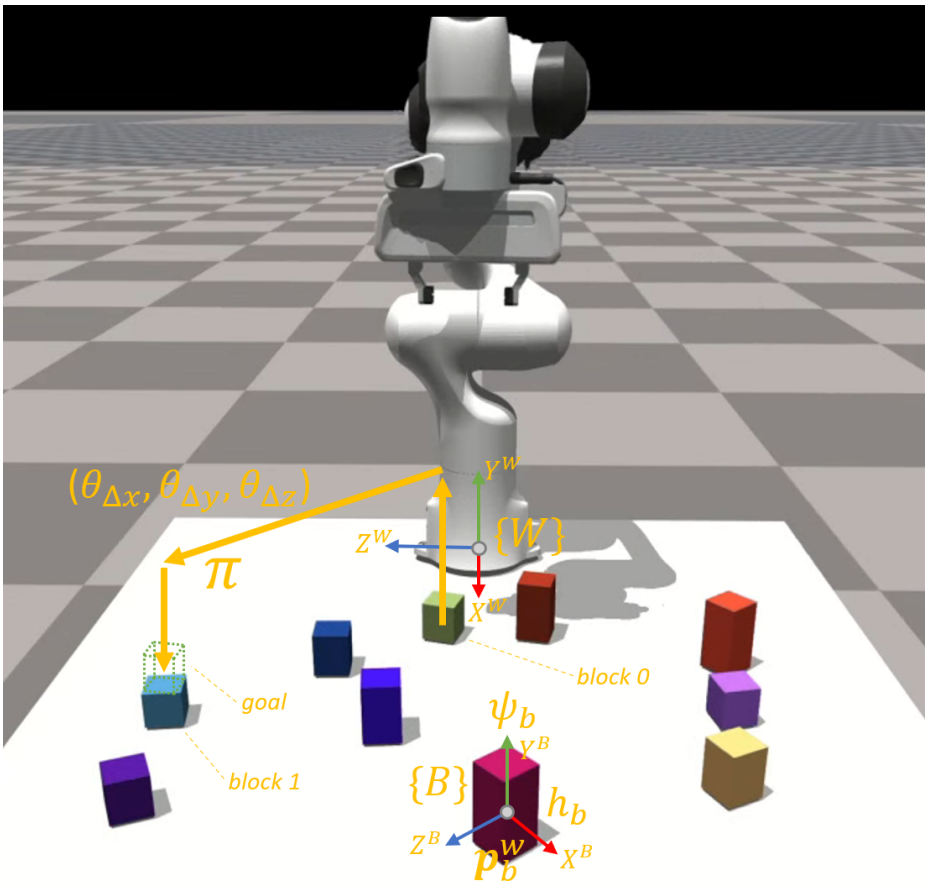



Figure B.1: Diagram of the block stacking task. The world coordinate frame $\{W\}$ is defined at the base link of the robot. Each block coordinate frame $\{B\}$ is defined at the block's centroid.

The vertical lift to $y_p$ avoids obstructions to moving the block, so the preconditions of this skill are always satisfied.

**Reward.** The reward function for this task is

$$r = -\alpha \|\boldsymbol{p}^w_{0,a} - \boldsymbol{p}^w_g\|$$

where $\alpha = 1$ is the reward weight, $\boldsymbol{p}^w_g = [\, x^w_g,\, y^w_g,\, z^w_g \,]^{\mathrm{T}}$ is the goal position of block 0, and $\boldsymbol{p}^w_{0,a} = [\, x^w_{0,a},\, y^w_{0,a},\, z^w_{0,a} \,]^{\mathrm{T}}$ is the final (i.e., achieved) position of block 0 at the end of the policy execution. In this task, the goal is to stack block 0 upon block 1. Therefore, $x^w_g = x^w_1$, $y^w_g = \frac{1}{2}h_0 + \frac{1}{2}h_1 + y^w_1$, and $z^w_g = z^w_1$.

Expressing the reward function in terms of the task context variables elucidates which variables are considered relevant to the task policy. The optimal low-level policy parameters $\theta^*_b = [\, \theta^*_{\Delta x},\, \theta^*_{\Delta y},\, \theta^*_{\Delta z} \,]^{\mathrm{T}} \in \mathbb{R}^3$ are

$$\begin{aligned}
\theta^*_{\Delta x} &= x^w_g - x^w_0 \\
&= x^w_1 - x^w_0 \\
\theta^*_{\Delta y} &= y^w_g - y^w_0 \\
&= \tfrac{1}{2}h_0 + \tfrac{1}{2}h_1 + y^w_1 - y^w_0 \\
&= \tfrac{1}{2}h_0 + \tfrac{1}{2}h_1 + \tfrac{1}{2}h_1 - \tfrac{1}{2}h_0 \\
&= h_1 \\
\theta^*_{\Delta z} &= z^w_g - z^w_0 \\
&= z^w_1 - z^w_0
\end{aligned}$$

where the reduction of the block $y$-position variables arise from the blocks being initially constrained to the table.

The above derivation demonstrates that only certain variables are needed to generalize the policy across different contexts where the preconditions also hold true. Moreover, certain variables are only influential in certain policy parameters. We express this more concretely by formalizing what variables are needed for each parameter, which is where the ground truth mappings for CREST (Sec. 4.7.1) arise:

$$\begin{aligned}
\theta^*_{\Delta x} = f(x^w_0, x^w_1) &\rightarrow \tau^*_{\Delta x} = [x^w_0, x^w_1]^T \\
\theta^*_{\Delta y} = f(h_1) &\rightarrow \tau^*_{\Delta y} = [h_1] \\
\theta^*_{\Delta z} = f(z^w_0, z^w_1) &\rightarrow \tau^*_{\Delta z} = [z^w_0, z^w_1]^T
\end{aligned}$$

$$\theta^*_b = f(x^w_0, x^w_1, h_1, z^w_0, z^w_1) \rightarrow \tau^* = [x^w_0, x^w_1, h_1, z^w_0, z^w_1]^T$$

Here, $f$ is the model, which for our work we characterize using a neural network (although for this specific task, a linear model would also suffice). In causality terms, this is equivalent to modeling each individual policy parameter ($\theta_j$) as a structural causal model, where $f$ is a function with parent variables given by $\tau_j$.

## B.4 Sim-to-Real Block Stacking Experiment

The sim-to-real block stacking experiment (Sec. 4.7.1) demonstrates that our proposed approach works in practice on a real robot system (Fig. 4.2c). As it is experimentally difficult to realize all possible values within the context distributions (e.g., creating blocks of precise height and color for each sample), we instead conduct the experiment on a slightly reduced distribution range. Specifically, we conduct this experiment using 10 blocks, where each block has a different color and two possible heights (5.7 cm or 7.6 cm). Before each trial, all block positions and rotations are shuffled by hand. Additionally, a random number generator selects the height of each block, as well as the enumeration of the blocks (and therefore which blocks are the source and target). The length and width of each block is 4.2 cm, which does not change during the experiment and is known from manual measurement (i.e., not perception).

**Perception.** We use a Microsoft Azure Kinect RGB-D camera to estimate each block's position, rotation, and color through a model-based perception algorithm utilizing the Open3D library [326]. Figure B.2 shows an example of the block perception. The perception algorithm is as follows:

1. Crop to region bounded by the table blue tape (Fig. 4.2c).
2. Removal of hidden points via Katz [327], i.e., points expected to be occluded from the camera viewpoint.
3. Fit plane to table via random sample consensus (RANSAC) and remove any points below this plane.
4. Detect remaining clusters with DBSCAN [328], a density-based clustering algorithm. Proceed only if the numbers of clusters is $N_B = 10$, or reject the perception sample and try again.
5. For each cluster (block), determine the best position and angle that fits a cube of known dimensions to the cluster via least-squares optimization. This step yields an estimate of each block's position and rotation.
6. Estimate block color by averaging the colors of all points within a cluster (block).

Due to difficulties with accurately estimating block height from depth, the block height is provided by manual input instead. Manual checks are also completed prior to executing the control policy to ensure block perception results are reasonable. For example, if one cluster was not a block, but part of the blue tape, the perception sample would be rejected and attempted again. Prior to running the perception system, we obtain the extrinsics of the camera via a target-based calibration procedure, and we use the intrinsics as reported directly from the camera.

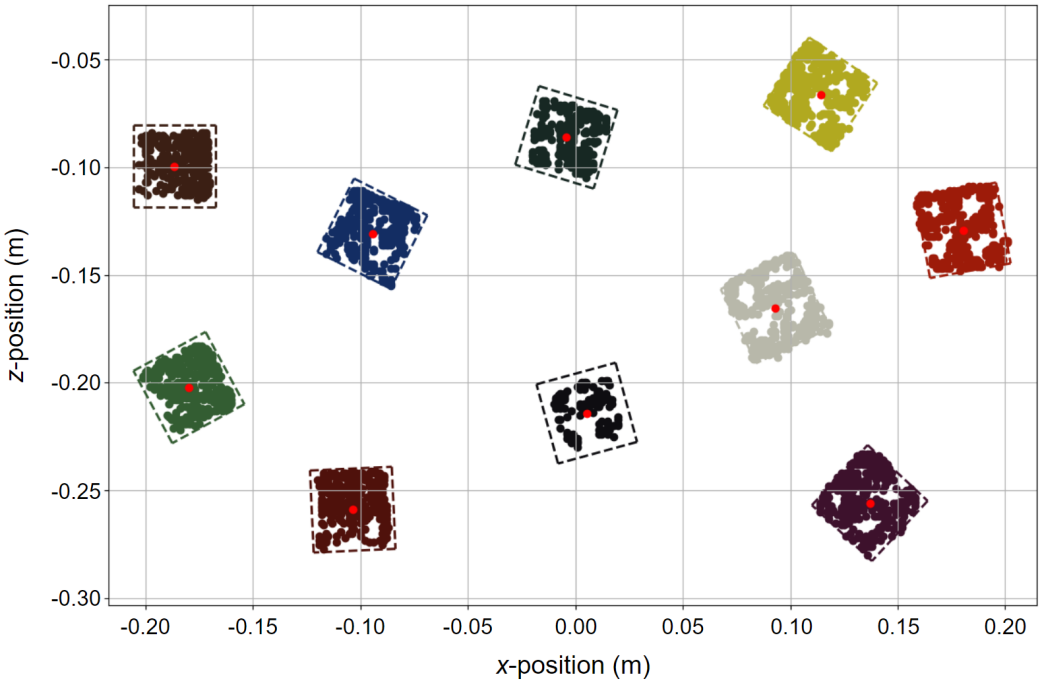



Figure B.2: Block state estimation used for the sim-to-real experiments using RGB-D perception. The perception algorithm takes as input a colored point cloud, and outputs a position, rotation, and color for 10 blocks. The red point in each block represents the block centroid, and the dashed lines indicate the block length and width (known *a priori*). The best-fit position and rotation angle for each point cloud cluster yields a pose estimate for each block.

**Control.** We use the `FrankaPy` library [329] that implements impedance control for the Franka Emika Panda robot.

## B.5 Task Representation: Crate Opening

This section provides more detail of the crate opening task (Sec. 4.7.2), where the objective is to open a crate in the presence of distractor objects using a circular arc skill with control policy $\pi(a|s, \theta_a)$ and known preconditions. Figure B.3 shows some context variables and the policy trajectory, which emerges from the crate grasp point.

This task has a larger modeling difference between the internal model and the target domain, which could also contribute to why our partitioned networks (PMLP, PMLP-R) were less successful than our non-partitioned network (RMLP). In addition to the dynamics domain difference discussed in Sec. 4.7.2, the $y$-position of the grasp point, $y_g^C$, is also slightly different. For the internal model, $y_g^C$ exists in the same plane as the crate, but for the target, $y_g^C$ is slightly above the crate because of the protruding grasp point.

**Policy.** The policy parameters $\theta_a = [\,{\theta_{\boldsymbol{p}_a^w}}^{\mathrm{T}},\, \theta_{\Delta\gamma},\, \theta_{\Delta\phi}\,]^{\mathrm{T}} \in \mathbb{R}^5$ define a circular arc that is composed of $N_T$ waypoints. The robot is commanded to the crate grasp point, then the robot executes the policy by sequentially following each waypoint via impedance control. The crate cannot open into the blocks below, so the preconditions are always satisfied.

**Reward.** The reward function for this task is

$$r = -\|[\,\alpha_a \Delta\Theta,\, \alpha_k e_k\,]^{\mathrm{T}}\|$$

In this function, $\Delta\Theta = \Theta_a - \Theta_o$ is the difference between the achieved ($\Theta_a$) and goal ($\Theta_o$) crate angles, and $\alpha_a$ and $\alpha_k$ are reward weights. The term $e_k$ is the kinematic error in the policy trajectory, which is intended to induce robot trajectories that are safe (physically realizable and low force) in the target domain given articulated motion of the crate. For this work, $\alpha_a = 1$ and $\alpha_k$ is 5 for the internal model and 0 for the target domain (because the robot realizes the trajectory it can actually achieve on the target due to the crate's articulated motion).

Specifically, $e_k = \frac{1}{N_T} \sum_{t=1}^{N_T} \|\boldsymbol{p}_{a,t}^w - \boldsymbol{p}_{d,t}^w\|$, where $\boldsymbol{p}_{d,t}^w$ is the desired position of a waypoint in the trajectory and $\boldsymbol{p}_{a,t}^w$ is the kinematically realizable position of that same waypoint, both at timestep $t$. This is determined by projecting the desired waypoint onto the plane formed by rotating the grasp point about the crate hinge, obtaining the resulting crate angle, and using this angle to compute the realized grasp point.

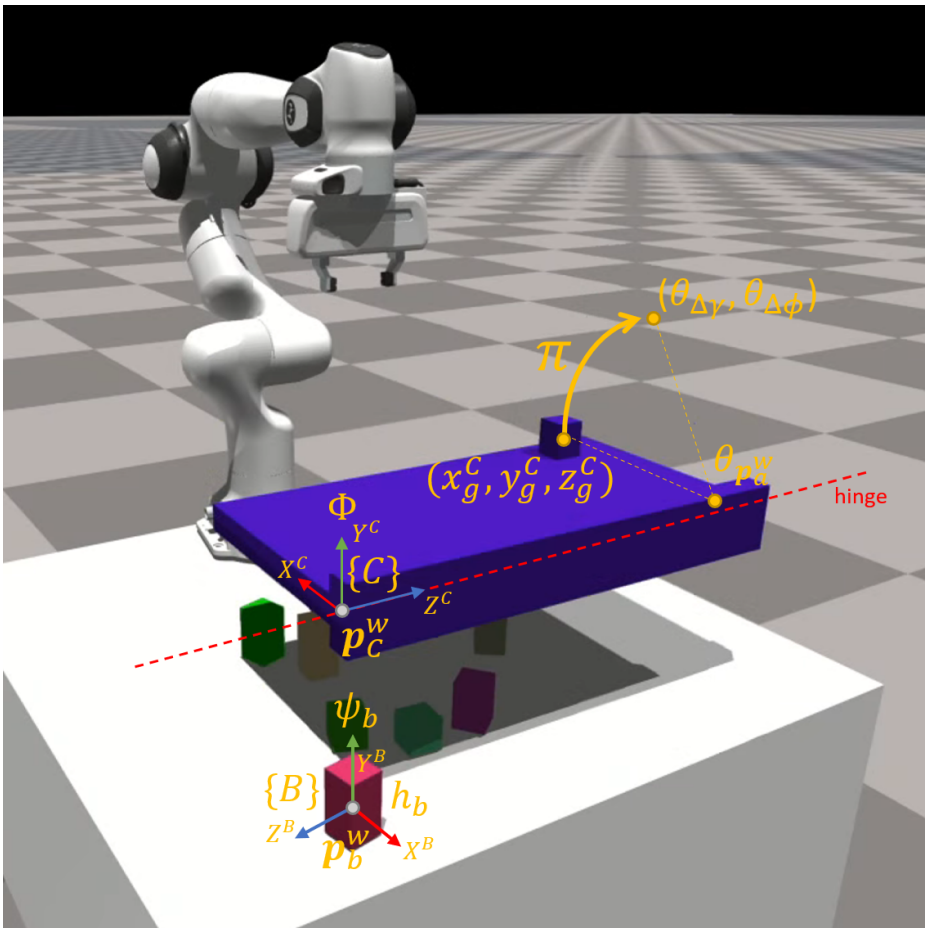



Figure B.3: Diagram of the crate opening task. The world coordinate frame $\{W\}$ (not shown) is defined at the base link of the robot, similar to the block stacking task (Fig. B.1). The $z$-axis of the crate coordinate frame $\{C\}$ is coincident with the crate hinge. There are 10 distractor blocks, each with coordinate frame $\{B\}$.

Table B.1: Transfer results for a distribution shift in 33 context variables that are irrelevant for the crate opening policy. Variables are 3-tuple RGB colors of the crate and 10 blocks in the scene. Light crate stiffness.

| Network | IM Updates (k-Samples) | Target Updates (k-Samples), no shift | Target Updates (k-Samples), shift |
|---|---|---|---|
| MLP | 36.10 ± 3.11<br>(18.48 ± 1.59) | 11.10 ± 2.30<br>(5.68 ± 1.18) | 17.30 ± 4.22<br>(8.86 ± 2.16) |
| RMLP (ours) | 36.20 ± 5.23<br>(18.53 ± 2.68) | 3.40 ± 0.66<br>(1.74 ± 0.34) | 3.50 ± 0.67<br>(1.80 ± 0.34) |
| PMLP (ours) | 48.50 ± 7.88<br>(24.80 ± 4.03) | 8.50 ± 2.06<br>(4.35 ± 1.06) | 8.30 ± 1.10<br>(4.25 ± 0.56) |
| PMLP-R (ours) | 53.00 ± 7.80<br>(27.14 ± 3.99) | 9.50 ± 1.91<br>(4.86 ± 0.98) | 9.60 ± 2.01<br>(4.92 ± 1.03) |

## B.6 Crate Opening Distribution Shift in Irrelevant Contexts

As mentioned in Sec. 4.7.2, we also conducted a crate opening experiment with distribution shifts in irrelevant parts of the context space, similar to the experiment in the block stacking task (Table 4.3). As before, we pretrain on the entire context space, except for color of the crate and blocks, where only half of the color space is used. For testing, we transfer to two cases: 1) the same color space seen in training (no shift), and 2) the opposite color space (complete shift with no overlap). This experiment uses the "light" crate stiffness.

Table B.1 shows the results of this experiment. As expected, our policies are robust to distribution shifts of this type, whereas the baseline MLP incurs approximately 55% more target updates to overcome these irrelevant distribution shifts. Unlike the version of this experiment for block stacking, no policies achieved zero-shot transfer. However, this is because of the previously described domain shift in dynamics between the internal model and target domain.

# C Appendix for SCALE

## C.1 SCALE and Appendices Overview

Fundamentally, SCALE is a causal learning algorithm for discovering compact, diverse skills through interventions in simulation. Figure C.1 provides an overview of the approach.

**Structure of appendices.** These appendices are structured as follows. Appendix C.2 describes how SCALE connects to related work in intuitive physics. Appendix C.3 provides greater details into the formalization of the simulator and its role as a causal reasoning engine. Appendix C.5 formalizes the SCALE algorithm using nomenclature introduced in App. C.4. A discussion of higher-dimensional context spaces and SCALE is then provided in App. C.6. Next, App. C.7 provides a toy experiment that is designed to convey greater intuition and visualization of the mechanisms that underlie SCALE. Appendix C.8 presents additional experimental details of the block stacking experiment presented in Sec. 5.7.1. Following this, Apps. C.9 and C.10 provides two additional experiments in the block stacking domain: a sim-to-real transfer experiment and a downstream task evaluation experiment, respectively. The next two appendices concern the peg-in-hole insertion domain. Appendix C.11 details additional experimental details first presented in Sec. 5.7.2, and App. C.12 presents an additional experiment that shows the robustness of SCALE under a task domain shift. Lastly, Appendix C.13 contains a primer on causality for readers who are new to this area of research.

## C.2 Related Work for Intuitive Physics

This appendix describes the connections between SCALE and the intuitive physics literature. Intuitive physics is the ability to approximately predict and model the physical world without explicit understanding of the underlying dynamics [330]. Lit-

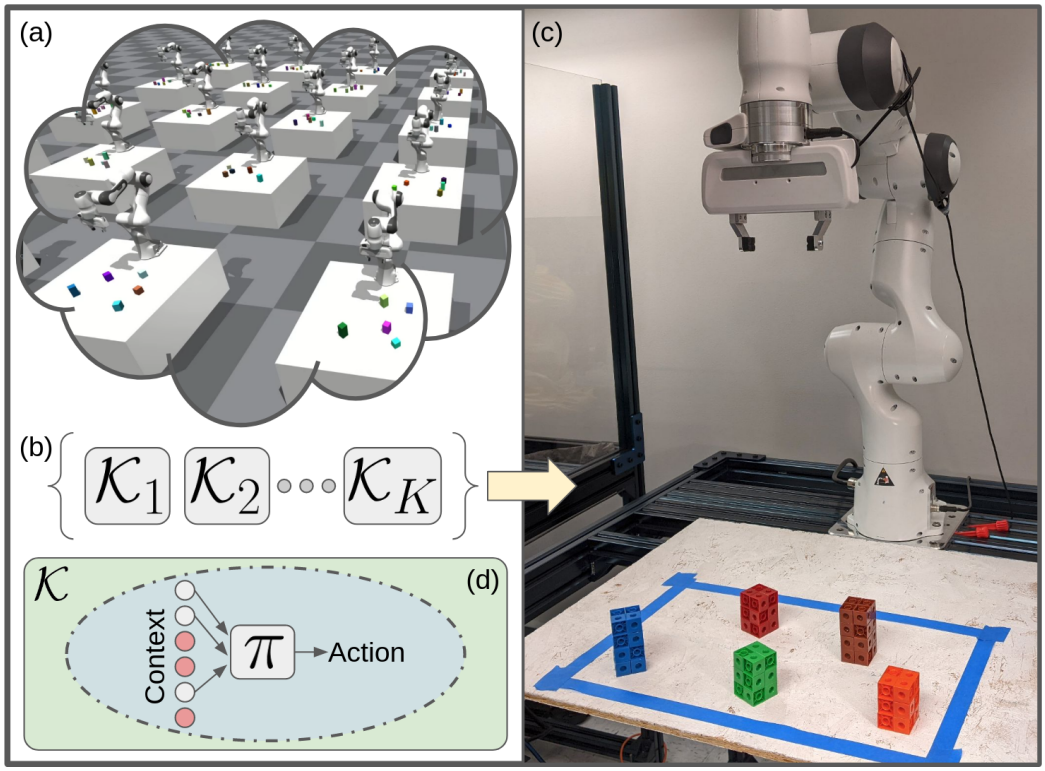
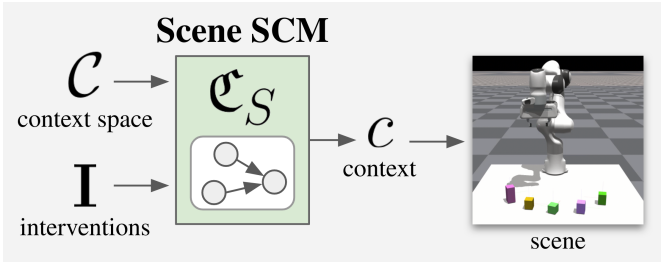
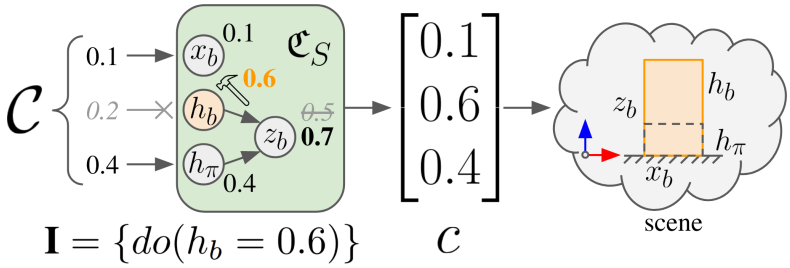



Figure C.1: In SCALE, the robot discovers skills in simulation using causal learning. (a) The simulation is used to solve task instances and conduct interventions to determine causally relevant context variables. (b) Simulation data are used to train a library of skills, (c) which are suitable for sim-to-real transfer learning. (d) Each skill that is learned is parameterized by the relevant variables selected in simulation. Here, red context variables are unnecessary for the skill policy and can be safely ignored. The boundary encircling the policy represents the skill DGR and precondition, which are also learned.




(a)




(b)

Figure C.2: Illustrations of the scene structural causal model used in the simulator $\mathcal{W}$. (a) From context space $\mathcal{C}$ and robot interventions $\mathbf{I}$, the scene SCM $\mathfrak{C}_S$ generates a context vector $c$ that represents a particular *scene* that defines objects and their properties. (b) In this block example, $\mathfrak{C}_S$ is defined using scene variables $\mathbf{\Psi} \coloneqq \mathbf{C} \cup z_b$ and context variables $\mathbf{C} \coloneqq \{x_b, h_b, h_\pi\}$, where $x_b$ is block x-position, $h_b$ is block height, $h_\pi$ is table height upon which the block rests, and $z_b \coloneqq \frac{1}{2}h_b + h_\pi$ is block z-position. Normally, values of $\mathbf{C}$ are sampled from context space $\mathcal{C}$, but the robot performs an intervention $\mathbf{I} = \{do(h_b = 0.6)\}$ to force the value of $h_b$ to be 0.6. As a result, the dependent variable $z_b$ is determined as 0.7 using this intervened value. Lastly, the scene is constructed and represented as context vector $c = [0.1, 0.6, 0.4]^{\mathrm{T}}$.

erature in cognitive psychology has suggested that humans develop mental intuitive physics models to support fast prediction and understanding of complex physical scenes which enables physical reasoning [150]. Computational learning of intuitive physics have been successful, enabling reinforcement learning and planning applications owing to the models ability for forward prediction [331–333]. In our work, our causal reasoning engine can be viewed as an internal model that uses interventions to elicit the physical mechanisms by which the data arise.

## C.3 Simulation as a Causal Reasoning Engine

This appendix provides greater discussion of the simulator formalization used by SCALE. The simulator model, $\mathcal{W} \coloneqq (\mathfrak{C}_S, T)$, is formalized as follows:

1. a scene structural causal model $\mathfrak{C}_S$ (Fig. C.2) that, given context space $\mathcal{C}$ and interventions $\mathbf{I}$, instantiates a scene that can be represented as a context vector, $c \in \mathcal{C}$;

2. the transition model $T$ that captures the domain forward dynamics as the robot interacts with the world through $\theta$ starting from the scene initialized from $\mathfrak{C}_S$.

A structural causal model (SCM) [16, 17] can be represented as a directed acyclic graph that is driven by exogenous variables (functional inputs of the graph) that produces the solution for all variables within the graph. These two components of the simulator capture the spatial structure inherent to the scene itself ($\mathfrak{C}_S$), and the spatiotemporal structure of the robot interacting with the world ($T$). The simulator model $\mathcal{W}$, including the scene SCM and transition function, is provided for the robot to use. In principle, the scene SCM could be learned via causal representation learning [25], e.g., a world models approach that admits causal interventions.

The scene SCM $\mathfrak{C}_S$ is defined by structural equations with scene variables $\mathbf{\Psi}$, where $\mathbf{C} \subseteq \mathbf{\Psi}$. In the graph induced by $\mathfrak{C}_S$, the scene variables are the nodes, and context variables $\mathbf{C}$ are the root nodes and exogenous variables (functional inputs) of the SCM. The value of the context variables is given by interventions $\mathbf{I} = \{do(C_i = c_i)\}$ if specified, or otherwise sampled from the context space $\mathcal{C}$. The robot only conducts interventions with respect to $\mathbf{C}$ that would yield a steady-state solution and are physically realizable, excluding physically invalid scenes (e.g., object penetration).

The transition model $T$ is the same as typical simulators. The forward dynamics are simulated through the initial state $s_0$, obtained from the scene created by $\mathfrak{C}_S$, and $\theta$, the inputs to the low-level controller $\pi_l$. With these inputs, the system temporally evolves as usual until the end of the episode, where reward $R_f$ is obtained and compared to a threshold $R_S$ to determine if the task was solved.

## C.4 Nomenclature

Table C.1 summarizes the nomenclature used in this paper and, in particular, the SCALE algorithm (c.f., App. C.5). Note the use of italics and bold type to disambiguate certain symbols. For example, $\mathbf{X}$ is a set of random variables, but $\boldsymbol{X}$ refers to a dataset matrix. The notation for a variable and its instantiation as a scalar may also be overloaded depending on the context.

Table C.1: Table of nomenclature.

| Symbol | Meaning |
|---|---|
| $\mathbf{X}$ | set of $d$ random variables, i.e., $\mathbf{X} \coloneqq \{X_1, \ldots, X_d\}$ |
| $\mathcal{X}$ | space of $\mathbf{X}$, i.e., $\mathcal{X} \coloneqq [\mathcal{X}_1, \ldots, \mathcal{X}_d]^{\mathrm{T}}$ |
| $x$ | vector instantiation of $\mathbf{X}$ i.e., $x \coloneqq [x_1 \in X_1 \subseteq \mathcal{X}_1, \ldots, x_d \in X_d \subseteq \mathcal{X}_d]^{\mathrm{T}}$ |
| $\boldsymbol{\mathcal{K}}$ | set of $k$ robot skills, i.e., $\boldsymbol{\mathcal{K}} \coloneqq \{\mathcal{K}_1, \ldots, \mathcal{K}_k\}$ |
| $\mathcal{D}$ | dataset containing ${}^{\mathbf{A}}\boldsymbol{X} \in \mathbb{R}^{m \times n}$ samples from set $\mathbf{A}$ with size $n$ and ${}^{\mathbf{B}}\boldsymbol{Y} \in \mathbb{R}^{m \times p}$ labels from set $\mathbf{B}$ with size $p$ |

## C.5 SCALE Algorithm

As explained in Sec. 5.6, the SCALE algorithm (Alg. C.1) describes how the skills are learned through batch dataset collection and skill training. The procedure for batch dataset collection used by SCALE (SkillTrainData) is described in Alg. C.2.

Note that the number of skills is not a hyperparameter of the SCALE algorithm. Rather, the skill quantity emerges from SplitIntoSkillDatasets from groups of highly-occurring CREST results, where each group becomes the dataset for a particular skill.

## C.6 SCALE and Higher-Dimensional Context Spaces

The SCALE algorithm scales linearly with the dimensionality of the context space, i.e., $O(|\mathbf{C}|)$, due to the necessity of performing interventions on each context variable. In the experiments examined in this work, the dimensionality of the context space was 36 and 8 for the block stacking and peg insertion domains, respectively. For other applications where the context space is very large, heuristics can be incorporated to first downselect the context space into a smaller candidate space that can be provided to SCALE. Example heuristics could include a distance metric (objects closer to the goal may be more likely to be relevant than those further away) or using other approaches such as meta-level priors [334].

**Algorithm C.1:** SCALE: Skills from Causal Learning

**Input:** causal reasoning engine $\mathcal{W}$, context space $\mathcal{C}$, controller $\pi_l$, reward solved threshold $R_S$, number of samples $n$, skill policy function $f_\pi$, number of evaluations $m$, skill timestep $T_f$

**Initialize:** skills $\boldsymbol{\mathcal{K}} \leftarrow \emptyset$

---

`// Collect training data`
$(\mathcal{D}_1, \ldots, \mathcal{D}_k) \leftarrow \textsc{SkillTrainData}(\mathcal{W}, \mathcal{C}, \pi_l, n)$
`// Train skills`
**for** $j = 1$ **to** $k$ **do**
  $({}^{C}\boldsymbol{X}, {}^{\theta}\boldsymbol{Y}, \mathbf{A}, \mathbf{D}) \leftarrow \mathcal{D}_j$
  `// Train DGR`
  ${}^{D}\boldsymbol{X} \leftarrow \textsc{ReduceDims}({}^{C}\boldsymbol{X}, \mathbf{D})$
  $\mathscr{D} \leftarrow \textsc{TrainDGR}({}^{D}\boldsymbol{X})$
  `// Train Policy`
  ${}^{A}\boldsymbol{X} \leftarrow \textsc{ReduceDims}({}^{C}\boldsymbol{X}, \mathbf{A})$
  $({}^{A}\boldsymbol{X}^{+}, {}^{\theta}\boldsymbol{Y}^{+}) \leftarrow \textsc{DGRInliers}(\mathscr{D}, {}^{A}\boldsymbol{X}, {}^{D}\boldsymbol{X}, {}^{\theta}\boldsymbol{Y})$
  $\pi_u \leftarrow \textsc{TrainPolicy}(f_\pi, {}^{A}\boldsymbol{X}^{+}, {}^{\theta}\boldsymbol{Y}^{+})$
  $\pi \leftarrow \pi_l \pi_u$
  `// Train Preconditions`
  $({}^{C}\boldsymbol{X}_e, {}^{R}Y_e) \leftarrow \textsc{EvaluatePolicy}(\mathcal{W}, \mathcal{C}, \pi, m)$
  $\text{Pre} \leftarrow \textsc{TrainPrecondition}({}^{C}\boldsymbol{X}_e, {}^{R}Y_e, R_S)$
  `// Set Termination Conditions`
  $\beta \leftarrow T_f$
  `// Construct Skill`
  $\boldsymbol{\mathcal{K}} \xleftarrow{+} (\pi, \text{Pre}, \beta, \mathscr{D})$
**end**

---

**Result:** learned skills $\boldsymbol{\mathcal{K}}$

**Algorithm C.2:** SkillTrainData

**Input:** causal reasoning engine $\mathcal{W}$, context space $\mathcal{C}$, controller $\pi_l$, reward solved threshold $R_S$, number of samples $n$, local region fraction $f$, minimum dataset size $d$

**Initialize:** batch dataset $\mathcal{D}_B \leftarrow \emptyset$

```
// Collect training data
for i = 1 to n do
    c ← SampleValidScene(W, C)
    (θ, R_f) ← TryToSolveTask(W, c, π_l)
    TaskSolved ← R_f > R_S
    if TaskSolved then
        A ← CREST(W, c, π_l, θ, R_f, fC)
        D ← CREST(W, c, π_l, θ, R_f, C)
        D_B ←+ (c, θ, A, D)
    end
end
// Separate into k skill datasets
(D_1, ..., D_k) ← SplitIntoSkillDatasets(D_B, d)
```

**Result:** skill training data $(\mathcal{D}_1, \ldots, \mathcal{D}_k)$

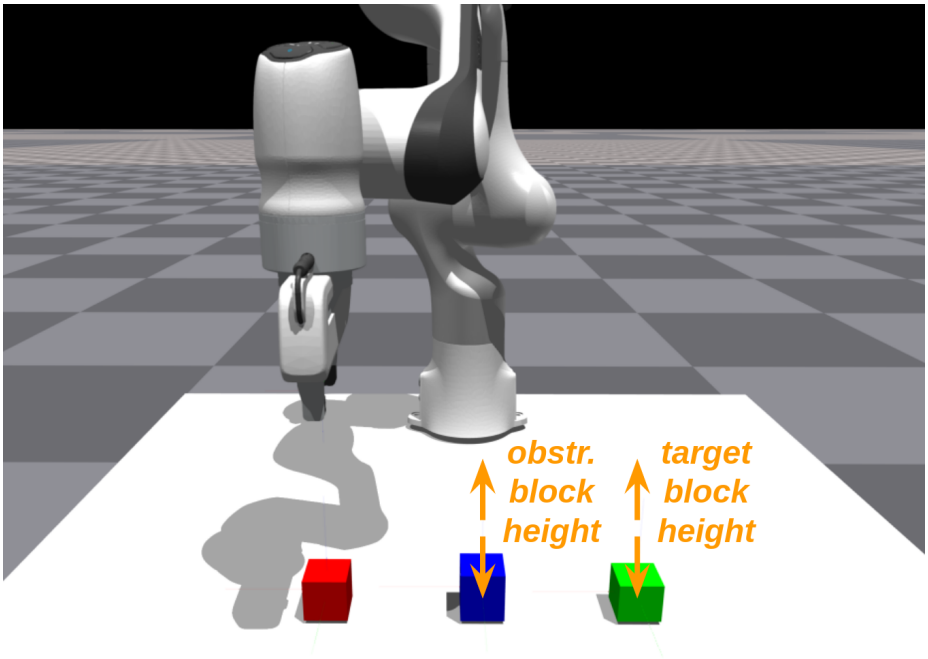



Figure C.3: The *Height-Height* experiment is an intuitive example for SCALE in the block stacking domain. In this experiment, only two context variables can vary: the height ($z$-dimension) of the obstructing block ($h_o$) and the height of the target block ($h_t$). All others variables (e.g., features of the source block) do not change throughout this experiment.

## C.7 Block Stacking Intuitive Example

To provide greater intuition for SCALE and the causal skill learning problem, we present the *Height-Height* experiment (Fig. C.3): a simple example in the block stacking domain that can be easily visualized.

**Task and policy description.** The *Height-Height* experiment contains 3 blocks: 1) a source block; 2) a target block; and 3) an obstructing block between the source and target block. As in Sec. 5.7.1, the task is to place the source block on top of the target block. The same controller is used as in Sec. 5.7.1, which is parameterized by $\theta \in \mathbb{R}^4$. Specifically, each parameter of the controller is defined as follows:

1. $\theta_{\Delta x}$: the distance the source block is moved along the world coordinate frame's $+x$-axis once it is picked up.

2. $\theta_{\Delta y}$: the distance the source block is moved along the world coordinate frame's $+y$-axis once it is picked up.

3. $\theta_{\Delta z_u}$: the distance the source block is lifted (moved along the world coordinate frame's $+z$-axis) during the pick-up motion.

4. $\theta_{\Delta z_d}$: the distance the source block descends (moved along the world coordinate frame's $-z$-axis) during the set-down motion.

The controller behaves as follows:

1. Move robot end-effector to source block and grasp it.

2. Lift up the source block according to $\theta_{\Delta z_u}$.

3. Move the source block in the $x$-$y$ plane according policy parameters $\theta_{\Delta x}$ and $\theta_{\Delta y}$.

4. Set down the source block according to $\theta_{\Delta z_d}$.

5. Ungrasp the source block.

The context space of this experiment is just 2 variables, $h_t$ and $h_o$, facilitating 2-dimensional visualizations. For greater clarity, we refer to block properties by whether they belong to the target block ($t$) or the obstructing block ($o$), instead of their index (as in Sec. 5.7.1). For this experiment, only linear approaches are considered.

**Skill learning results.** The SCALE results for the *Height-Height* experiment are shown in Tab. C.2 and Fig. C.4. The dataset size for skill learning was 569 samples, from an original size of 581. The remaining 12 samples consisted of CREST results that occurred rarely (2.07%), and thus they were not used for skill learning. Additionally, Fig. C.5 visualizes the policy parameters of the dataset. Two primary behaviors were learned: *free motion* ($\mathcal{K}_{free}$), and *obstructed motion* ($\mathcal{K}_{obstr}$). These behaviors emerge *because of* the causal relationships between context variables.

When the obstructing block is shorter than the target block (i.e., $h_t > h_o$), then the obstructing block height can safely be ignored in the robot action (thus, $h_o \notin \mathbf{A}$ for $\mathcal{K}_{free}$). This is reflected by the values of $\theta_{\Delta z_u}$ and $\theta_{\Delta z_d}$ in Fig. C.5. In the region corresponding to $\mathcal{K}_{free}$, $\theta_{\Delta z_u}$ varies linearly with respect to the target block height, but not with the obstructing block height. Thus, $\theta_{\Delta z_d}$ is generally 0. The result is that the robot tends to lift the block to a value that depends on the target block height, and no set-down motion ($\theta_{\Delta z_d}$) is needed.

However, when the obstructing block is taller than the target block (i.e., $h_t < h_o$), the obstructing block's geometry interferes with the robot's motion, and the robot must take this into account when taking action. Specifically, the robot must first lift the source block over the obstructing block. After it moves laterally, the robot must descend to set the source block down; dropping the block would typically lead to inadequate reward to solve the task. Because both the heights of these blocks are needed to perform this action, $\{h_t, h_o\} \subseteq \mathbf{A}$ for $\mathcal{K}_{obstr}$. In Fig. C.5, the effect of $h_o$ appears in the $\theta_{\Delta z_u}$ parameter values, where the variation in the $\mathcal{K}_{obstr}$ region

Table C.2: Skills $\boldsymbol{\mathcal{K}}_{HH}$ that were discovered for the *Height-Height* experiment. **A** and **D** are the variables used for the skill's policy and DGR, respectively. Data is the quantity of data used for each skill (from a batch dataset of 581 samples, 569 samples were used to train skills). These samples are used to train a linear policy (Bayesian ridge regression) using the features from variables in **A**. Task Solve % is the rate of task solves over the entire context space using only that skill.

| Skill | **A** | **D** | Data | Task Solve % |
|---|---|---|---|---|
| $\mathcal{K}_{free}$ | $\{h_t\}$ | $\{h_t, h_o\}$ | 253 (43.55%) | 55.63% (178) |
| $\mathcal{K}_{obstr}$ | $\{h_t, h_o\}$ | $\{h_t, h_o\}$ | 316 (54.39%) | 57.50% (184) |

arises because of needing to lift above the obstructing block height, $h_o$ (and thus, this parameter no longer depends on $h_t$). However, for $\theta_{\Delta z_d}$, *both* $h_t$ and $h_o$ are needed, as the distances the robot descends through $\theta_{\Delta z_d}$ arises from the difference between $h_t$ and $h_o$. Thus, the gradient here shows components for both $h_t$ and $h_o$.

These two skills encode the two distinct data generating processes within this context space. These processes — the reason why the data are generated a certain way — fundamentally depend on whether the obstructing block is shorter or taller than the target block. Whether a condition holds for a given context requires the value of both of the blocks heights, so both block heights are needed to define each skill's data generating region (i.e., $\{h_t, h_o\} \subseteq \mathbf{D}$).

Note that neither skill can robustly solve the entire task space (55.63% for $\mathcal{K}_{free}$ and 57.50% for $\mathcal{K}_{obstr}$). However, when using the entire library $\boldsymbol{\mathcal{K}}_{HH} = \{\mathcal{K}_{free}, \mathcal{K}_{obstr}\}$ (Tab. C.3), the success rate becomes 100.00%, with each skill being selected at approximately 50% chance (49.38% for $\mathcal{K}_{free}$, and 50.62% for $\mathcal{K}_{obstr}$). This is expected because the relationship $h_t > h_o$ holds for half of the context space and $\mathcal{K}_{free}$ should be used, whereas $h_t < h_o$ ($\mathcal{K}_{obstr}$) holds for the other half.

**Baseline comparisons.** In addition to scale-lin, Tab. C.3 shows comparisons against several baselines. The "monopolicy" baselines are monolithic policies (without skills). The "-sk" and "-all" suffixes denote whether the monolithic policy uses the same data as the SCALE library ("-sk", 569 samples) or the entire batch dataset ("-all", 581 samples). Given the similar amount of data, it is unsurprising that monopolicy-lin-sk and monopolicy-lin-all are essentially the same up to the stochasticity of the simulator (±2%). Note that, unlike in Sec. 5.7.1 and Sec. 5.7.2, CREST monopolicy baselines are not examined in this experiment; they are functionally equivalent to the monopolicy approaches because the most common CREST result is $\{h_t, h_o\}$, which is the same as the entire context space used for the monopolicy baselines.

As shown in Tab. C.3, the skill library obtained by SCALE vastly outperforms

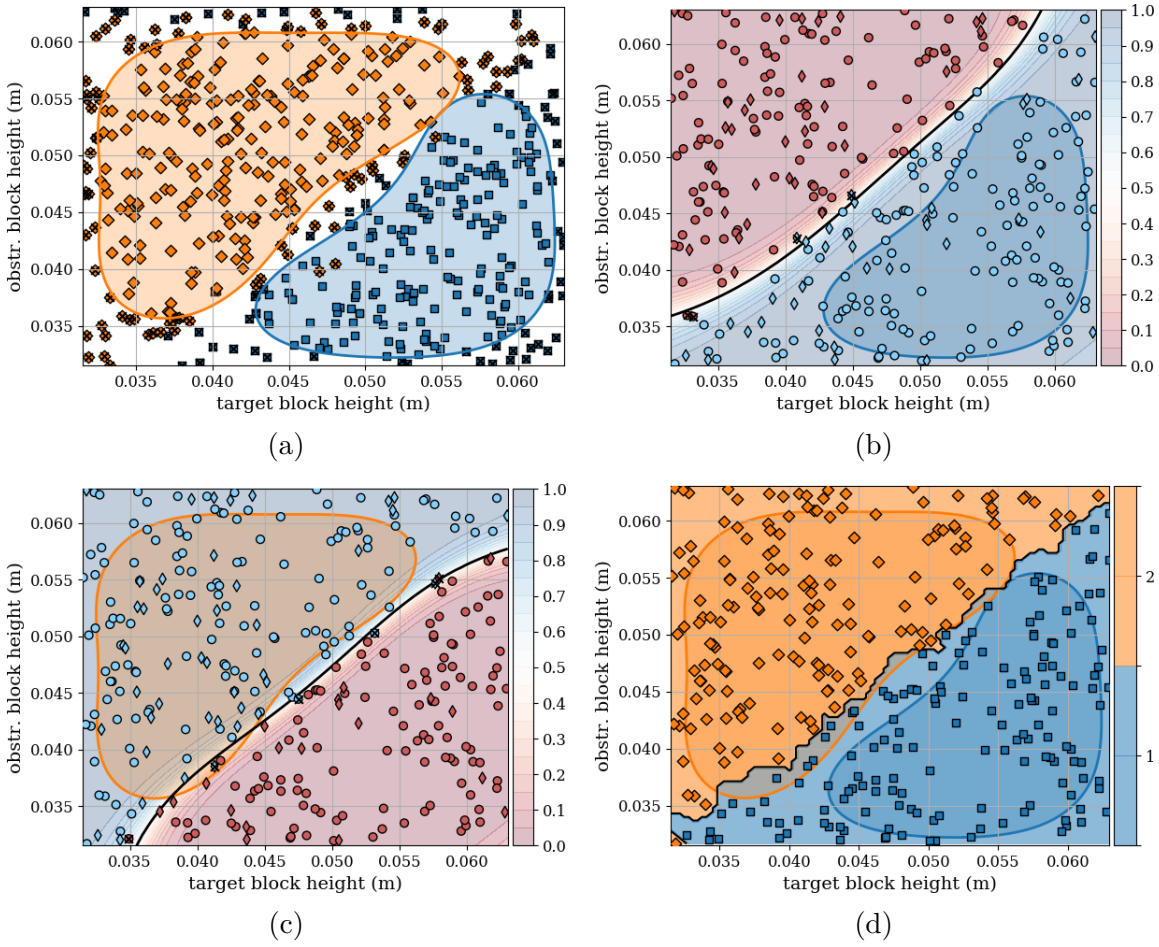



Figure C.4: SCALE results for the *Height-Height* experiment. Two skills were found: $\mathcal{K}_{free}$ (free block motion), stylized in blue with rectangular markers, and $\mathcal{K}_{obstr}$ (obstructed block motion), stylized in orange with diamond markers. (a) Learned data generating regions. Each datapoint is a result from CREST. Datapoints that are crossed out are considered outliers and not used for training the policy for that skill. (b–c) Preconditions for $\mathcal{K}_{free}$ and $\mathcal{K}_{obstr}$, respectively. The black line is the decision boundary for the prediction of whether the task would or would not be solved with that skill. Note that each skill's DGR generally falls within the positive precondition boundary. Training and test data for learning the preconditions are indicated by circle and thin diamond markers, respectively. Datapoints that result in a different prediction than observed are crossed out. (d) Task evaluation when using the skill library $\{\mathcal{K}_{free}, \mathcal{K}_{obstr}\}$ to solve the task. The marker and color of each datapoint indicate which skill was selected for completing the task based on the skill preconditions (i.e., the skill with the highest probability of success). Note that the separation between selecting $\mathcal{K}_{free}$ and $\mathcal{K}_{obstr}$ is consistent with each skills' underlying precondition and DGR. Datapoints that were not solved by the chosen skill are crossed out.

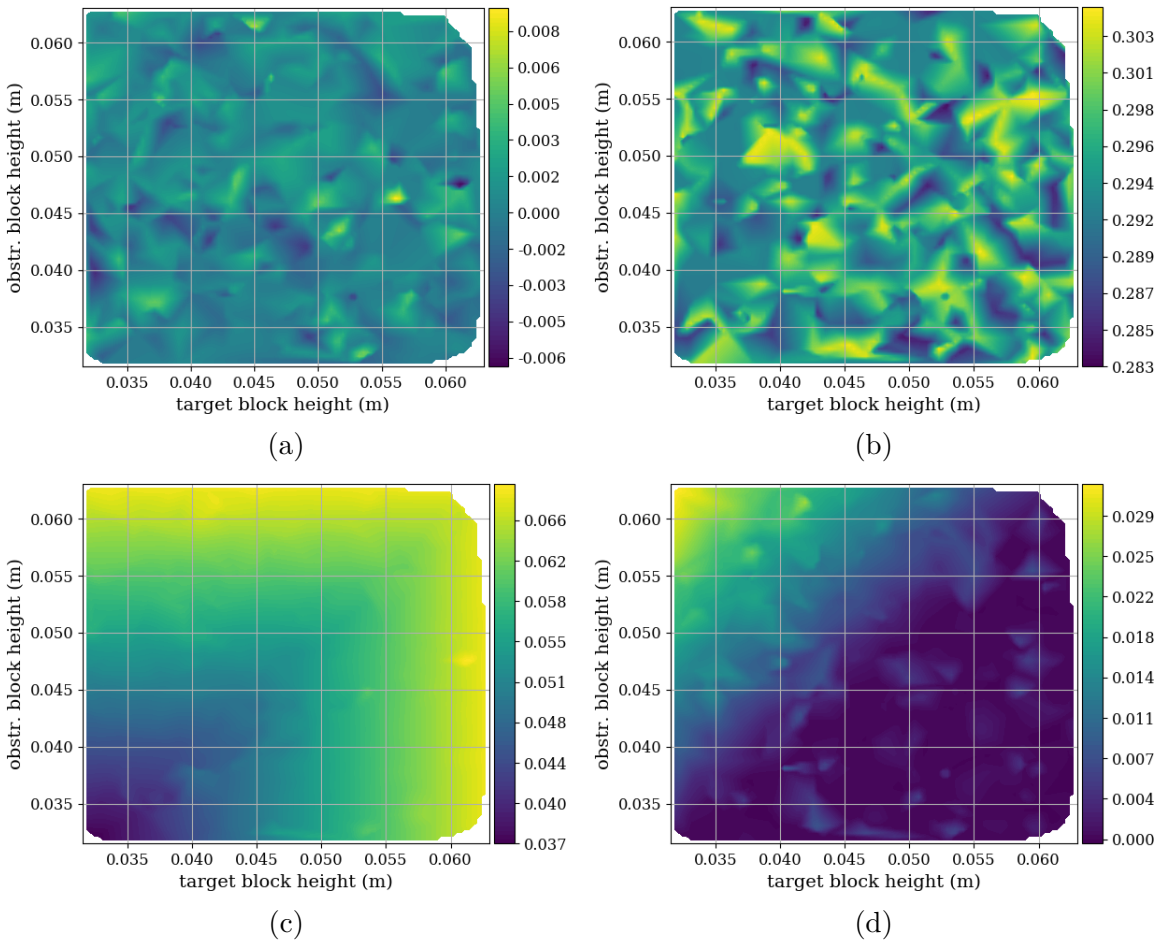



Figure C.5: Policy parameters for the *Height-Height* experiment (shown as interpolated across the 569 dataset samples to better visualize the gradients). The units of the parameters are in meters. The parameters $\theta_{\Delta x}$ (a) and $\theta_{\Delta y}$ (b) are generally constant as they are unaffected by the variation in context variables. The notable variations occur in $\theta_{\Delta z_u}$ (c) and $\theta_{\Delta z_d}$ (d). Specifically, the relationship changes whether the obstructing block is taller or shorter than the target block (above or below the $h_t - h_o = 0$ line, respectively).

Table C.3: Task evaluation results for using the skill library $\boldsymbol{\mathcal{K}}_{HH}$ for the block stacking task. Ctrl. is the approach control (skills or one monolithic policy). Fn. Cl. is the approach's function class. Linear approaches use Bayesian ridge regression. Task Solve % is the rate of task solves over the entire context space using the approach. Methods within ±2% (the stochasticity of the simulator) of the best approach are bold. $|\mathbf{A}|$ is the quantity of input variables used for the approach's policy. Data is the amount of training data used for the approach. A ground truth policy is also shown, using all context variables and additional domain knowledge.

| Approach | Ctrl. | Fn. Cl. | Task Solve % | $\lvert\mathbf{A}\rvert$ | Data |
|---|---|---|---|---|---|
| scale-lin (ours) | 2 skills | Linear | **100.00%** (320) | 1/1 | 569 |
| monopolicy-lin-sk | 1 policy | Linear | 64.06% (205) | 2 | 569 |
| monopolicy-lin-all | 1 policy | Linear | 62.19% (199) | 2 | 581 |
| ground-truth-policy | 1 policy | Nonlin. | 100.00% (320) | * | – |

the baselines, providing task evaluation performance similar to that of a ground truth policy. This outcome is possible because SCALE learns underlying regions of similar causal structure within the data, whereas monolithic policies ignore such structure. As shown in Fig. C.5c–C.5d, this domain is nonlinear, but can be represented by two smaller linear regions ($h_t > h_o$ and $h_t < h_o$). Learning to regress to both regions with a monolithic linear policy is not possible, but SCALE can solve this domain with separate linear skills, one per region.

**Summary.** Our approach for SCALE — learning skills that encode distinct causal processes — empowers the robot with a diversity of specialized behaviors to use, depending on the context. Generalization of the context space can be achieved then through the composition of these behaviors, rather than attempting to learn a monolithic skill or policy that can capture the entire variation. In this example, two skills each with a linear policy is sufficient for generalization with SCALE, whereas a monolithic approach would require a nonlinear policy.

## C.8 Additional Details for Block Stacking Experiment

This appendix provides greater information for the block stacking experiment first presented in Sec. 5.7.1.

**Context.** Note that the block vertical position $z_b^w \in \boldsymbol{\Psi}$ is not part of the context, as we only consider cases where the scene can be initialized into a steady state condition. Thus, $z_b^w \coloneqq \frac{1}{2}h_b + h_\pi$.

**Reward function.** The reward function for the task is $R = R_B - \alpha_L L - \alpha_e e - \alpha_d d$, where $R_B = 10$ is a bonus term obtained when the block is successfully stacked, $L$ is

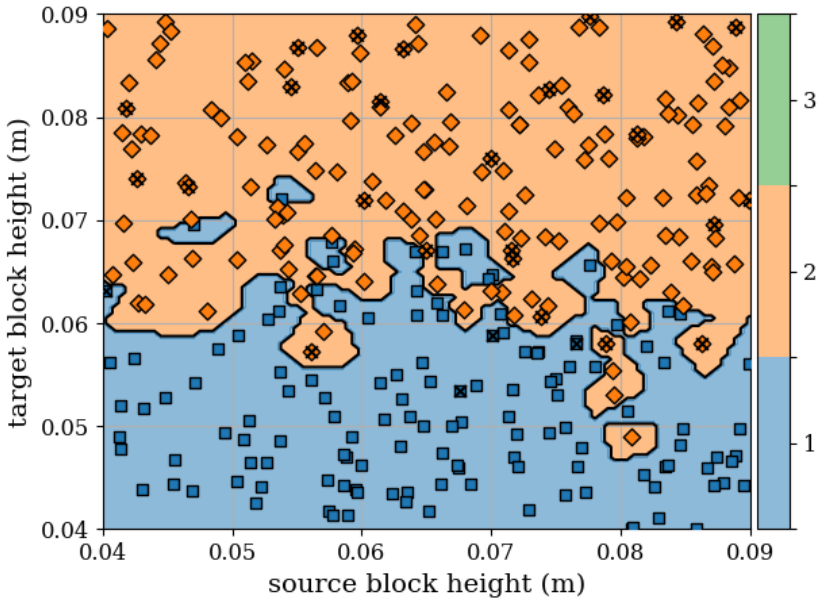



Figure C.6: Skill selection for the scale-lin approach for the block stacking task. Skill $\mathcal{K}_1$ is generally selected when $h_2$ is short, whereas taller $h_2$ values perform better with $\mathcal{K}_2$ because $h_2 \subseteq \mathbf{A}$. Skill $\mathcal{K}_3$ is dominated by the other two skills and is not selected. Datapoints that were not solved are crossed out.

the total end-effector path of the robot ($\alpha_L = 1$), $e$ is the L2 norm error between the source block at the time of release and the goal ($\alpha_e = 1$), and $d$ is the distance the source block travels between the point it was ungrasped to its final position ($\alpha_d = 1$). The task is considered solved if the final reward $R_f$ exceeds solved threshold $R_S = 5$.

**SCALE skill selection.** In all SCALE approaches, the skills were complementary; using the entire skill library afforded greater coverage (greater task solve rate) than any single skill alone. For scale-lin, the skill selection distribution was almost even between $\mathcal{K}_1$ (43.28%) and $\mathcal{K}_2$ (56.72%), with $\mathcal{K}_3$ never being chosen. The skill $\mathcal{K}_3$ is dominated by the other two skills for this task, but $\mathcal{K}_3$ could nonetheless be useful for a different downstream task. Empirically, it was observed that $\mathcal{K}_1$ was chosen for shorter target block heights, whereas $\mathcal{K}_2$ was used elsewhere (see Fig. C.6). In the nonlinear case, only $\mathcal{K}_2$ was selected.

**Policy and training data ablations.** We provide additional experiments to investigate the effect of different policy functions and training data usage. The results are shown in Tab. C.4, which expands Tab. 5.2. For the linear function class, we conduct experiments with Bayesian ridge regression (B. ridge reg.) and ordinary least squares linear regression (OLS lin. reg.). Both linear policy functions used an intercept term and were trained using unnormalized data. For the nonlinear function class, we

conduct experiments with a multilayer perceptron (MLP, 16x16x16 architecture using ReLU activations) and support vector regression with a radial basis function (RBF) kernel (SVR (RBF)). The nonlinear policy functions were trained with normalized data. Additionally, we present ablations in terms of training data usage. Methods ending in "-all" use the entire batch dataset. For the full-dimensional monopolicy approaches, the "-sk" ablation uses same training data as used by the SCALE skills (340 samples). For the CREST baselines, the "-subs" ablation randomly downselects the batch dataset to the same number of samples used by SCALE (340 samples).

In general, we see that SCALE generally outperforms the full-dimensional monopolicy methods and matches the performance of the CREST baselines in most (but not all) cases. We see that increasing the amount of training data available for the baselines usually improves performance. For the linear function class, both Bayesian ridge regression and ordinary least squares linear regression produced capable approaches. For ordinary least squares linear regression, SCALE (scale-lin-ols) outperforms the full-dimensional monopolicy on a sample-adjusted basis. For the nonlinear function class, the performance of approaches was lower overall. The similarity in performance of scale-nonlin to the full-data CREST baseline is strictly due to sample size; on a sample-adjusted basis, scale-nonlin is slightly more performant. However, for support vector regression with a RBF kernel, although SCALE (scale-nonlin-svr-rbf) exceeds the performance of the full-dimensional monopolicy approaches, the CREST approaches perform more strongly (although modestly overall). Thus, we see some sensitivity for the nonlinear function class to the selection of policy function used for this task.

## C.9 Sim-to-Real Block Stacking Experiment

In this appendix, we demonstrate that the skills learned by SCALE are suitable for sim-to-real transfer. As skills are constructed using only the relevant causal variables, this is a form of *structural* sim-to-real transfer. For this experiment, we evaluate the skill library $\boldsymbol{\mathcal{K}}_{blocks}$ for a real block stacking domain with a Franka Emika Panda robot manipulator (Fig. 5.2c). This experiment is generally similar to task evaluation in simulation, except with a smaller subset of the context space. We assess the SCALE approaches, scale-lin and scale-nonlin, against their monopolicy counterparts. We only consider the "-all" monopolicy approaches, as they were generally better performing.

### C.9.1 Experimental Setup

For this experiment, a smaller subset of the context space is varied, as compared to the variation across the entire context space as tested in Tab. 5.2. From a pool of 20 blocks, 5 were randomly chosen to be used for each experimental trial. The 20

Table C.4: Task evaluation results for using the skill library $\mathcal{K}_{blocks}$ for the block stacking task for a variety of policy functions and training data ablations. This table expands upon Tab. 5.2. Ctrl. is the approach control (skills or one monolithic policy). Fn. Cl. is the approach's function class. P. Fn. is the policy function. Task Solve % is the rate of task solves over the entire context space using the approach. Methods within ±2% (the stochasticity of the simulator) of the best approach are bold. $|\mathbf{A}|$ is the quantity of input variables used for the approach's policy. Data is the amount of training data used for the approach. A ground truth policy is also shown, using all context variables and additional domain knowledge. The abbreviation "mp" stands for monopolicy.

| Approach | Ctrl. | Fn. Cl. | P. Fn. | Task Solve % | $\lvert\mathbf{A}\rvert$ | Data |
|---|---|---|---|---|---|---|
| scale-lin (ours) | 3 skills | Linear | B. ridge reg. | **90.49%** (276) | 4/5/6 | 340 |
| monopolicy-lin-sk | 1 policy | Linear | B. ridge reg. | 80.72% (247) | 36 | 340 |
| monopolicy-lin-all | 1 policy | Linear | B. ridge reg. | 85.95% (263) | 36 | 585 |
| crest-monopolicy-lin-subs | 1 policy | Linear | B. ridge reg. | **89.87%** (275) | 5 | 340 |
| crest-monopolicy-lin-all | 1 policy | Linear | B. ridge reg. | **89.87%** (275) | 5 | 585 |
| scale-lin-ols (ours) | 3 skills | Linear | OLS lin. reg. | **90.85%** (278) | 4/5/6 | 340 |
| monopolicy-lin-ols-sk | 1 policy | Linear | OLS lin. reg. | 83.33% (255) | 36 | 340 |
| monopolicy-lin-ols-all | 1 policy | Linear | OLS lin. reg. | **90.16%** (275) | 36 | 585 |
| crest-monopolicy-lin-ols-subs | 1 policy | Linear | OLS lin. reg. | **90.52%** (277) | 5 | 340 |
| crest-monopolicy-lin-ols-all | 1 policy | Linear | OLS lin. reg. | **90.20%** (276) | 5 | 585 |
| scale-nonlin (ours) | 3 skills | Nonlin. | MLP | **63.40%** (194) | 4/5/6 | 340 |
| monopolicy-nonlin-sk | 1 policy | Nonlin. | MLP | 1.31% (4) | 36 | 340 |
| monopolicy-nonlin-all | 1 policy | Nonlin. | MLP | 10.13% (31) | 36 | 585 |
| crest-monopolicy-nonlin-subs | 1 policy | Nonlin. | MLP | 58.17% (178) | 5 | 340 |
| crest-monopolicy-nonlin-all | 1 policy | Nonlin. | MLP | **60.78%** (186) | 5 | 585 |
| scale-nonlin-svr-rbf (ours) | 3 skills | Nonlin. | SVR (RBF) | 19.61% (60) | 4/5/6 | 340 |
| monopolicy-nonlin-svr-rbf-sk | 1 policy | Nonlin. | SVR (RBF) | 1.63% (5) | 36 | 340 |
| monopolicy-nonlin-svr-rbf-all | 1 policy | Nonlin. | SVR (RBF) | 7.19% (22) | 36 | 585 |
| crest-mp-nonlin-svr-rbf-subs | 1 policy | Nonlin. | SVR (RBF) | 41.64% (127) | 5 | 340 |
| crest-mp-nonlin-svr-rbf-all | 1 policy | Nonlin. | SVR (RBF) | **56.86%** (174) | 5 | 585 |
| ground-truth-policy | 1 policy | Nonlin. | – | 95.75% (293) | * | – |

blocks consisted of variations of 10 different colors and 2 different heights (5.7 cm or 7.6 cm). The length and width of the blocks were 4.2 cm. The 5 randomly chosen blocks were placed into the Panda robot workspace and randomly shuffled, producing variation in block $x$-position, $y$-position, and orientation. The table height $h_\pi$ was determined from manual measurement and was not varied for this experiment.

**Perception.** An Intel RealSense camera mounted to the robot wrist provided RGB-D perception of the $x$-position, $y$-position, and orientation of the blocks in the workspace. A depth observation was collected by commanding the robot above the workspace. This point cloud was then processed to yield five clusters via hidden point removal [327], RANSAC-based table plane fitting, and density-based clustering using DBSCAN [328]. Averaging the colors within each cluster yielded the block color. A least-squares optimization procedure fit a cuboid of known length and width to each cluster, yielding the position and orientation of the blocks. Block height was provided by manual input because of inaccuracies with estimation from depth alone. The camera extrinsics were obtained via computer-aided design models of the Panda robot and wrist mount, which were confirmed via manual measurement. The camera intrinsics were used as directly reported by the camera.

**Control.** The `FrankaPy` library [329] is used to provide impedance-based control of the Panda robot.

### C.9.2 Experimental Results

Table C.5 presents the results. For each function class, the skill library learned by SCALE outperforms the full-dimensional monopolicy baseline and is generally comparable to or slightly outcompetes the CREST monopolicy baseline. The ground truth policy matched the linear SCALE approach and is only slightly better than the nonlinear SCALE approach. Compared to the task solve rate in simulation (Tab. 5.2), scale-lin performed consistently, and scale-nonlin had slightly better performance. All baseline approaches generally matched their evaluation in simulation, except for monopolicy-lin-all, which had a marked degradation. This may arise from domain differences between simulation and reality. Full-dimensional approaches are more susceptible to domain shifts due to their reliance on the entire context space (all 36 variables), whereas SCALE approaches are compressed, using only a minimal subset. Error was only loosely correlated with task solve rate, and likely explains the poor performance of monopolicy-nonlin-all. Even though their errors were similar, it was observed that monopolicy-lin-all tended to underpredict the height needed to clear the target block as compared to scale-lin. This caused the target block to be pushed away from where it should have been for the goal position, leading to block stacking failures.

For both scale-lin and scale-nonlin, skill $\mathcal{K}_2$ was always chosen, as its precondition was on average greater than that of the other skills. Specifically, for scale-lin, the average preconditions were 58.88% for $\mathcal{K}_1$, 75.77% for $\mathcal{K}_2$, and 36.99% for $\mathcal{K}_3$. As the

Table C.5: Sim-to-real evaluation results for using the skill library $\boldsymbol{\mathcal{K}}_{blocks}$ for a real block stacking domain. Table columns are as described in Tab. 5.2. Task Solve % is the rate of successful block stacks. Error is the mean error ($\pm 1$ standard deviation) in meters between the block position when the block is ungrasped and the goal position determined at the beginning of the trial.

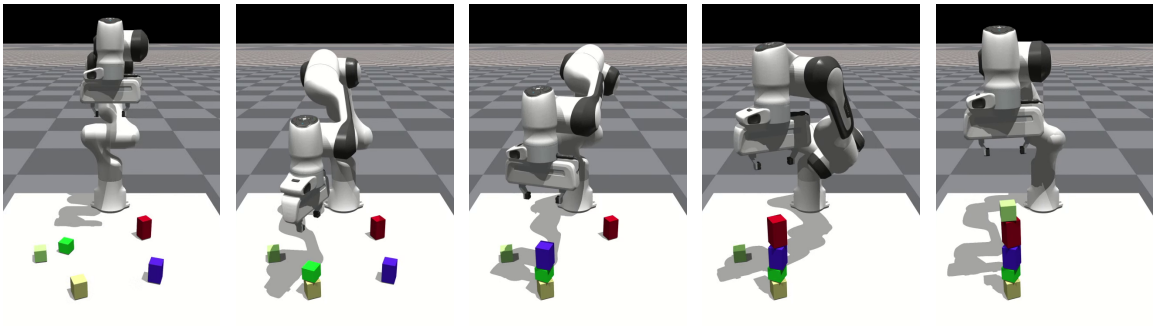
| Approach | Ctrl. | Fn. Cl. | Task Solve % | Error | $\lvert\mathbf{A}\rvert$ |
|---|---|---|---|---|---|
| scale-lin (ours) | 3 skills | Linear | **90.00%** (9) | 0.010 ± 0.003 | 4/5/6 |
| monopolicy-lin-all | 1 policy | Linear | 50.00% (5) | 0.008 ± 0.003 | 36 |
| crest-monopolicy-lin-all | 1 policy | Linear | **90.00%** (9) | 0.004 ± 0.001 | 5 |
| scale-nonlin (ours) | 3 skills | N.L. | **80.00%** (8) | 0.007 ± 0.002 | 4/5/6 |
| monopolicy-nonlin-all | 1 policy | N.L. | 10.00% (1) | 0.093 ± 0.040 | 36 |
| crest-monopolicy-nonlin-all | 1 policy | N.L. | 70.00% (7) | 0.013 ± 0.012 | 5 |
| ground-truth-policy | 1 policy | N.L. | 90.00% (9) | 0.002 ± 0.003 | * |

block heights used were only 5.7 cm and 7.6 cm, it is reasonable to expect that skill $\mathcal{K}_1$ would have been chosen more for shorter target block heights (per Fig. C.6). For scale-nonlin, the average preconditions were $\mathcal{K}_1$: 20.17%, $\mathcal{K}_2$: 51.84%, $\mathcal{K}_3$: 1.21%.

## C.10 Skill Library Use in a Downstream Task: Stacking a Block Tower

To demonstrate the utility of re-using skills learned by SCALE, a follow-up experiment is conducted wherein the skill library $\boldsymbol{\mathcal{K}}_{blocks}$ is used for a task in which it was not specifically trained: stacking a block tower (Fig. C.7). This long-horizon task can be decomposed into a number of sequential actions that must be performed correctly, so an approach that can capture the essence of a large problem and re-use smaller, modular components should perform best. Moreover, we do not perform any additional training or fine-tuning; we intentionally use the skills off-training data to test their generalization capability. This is a challenging task: in addition to the long-horizon precision involved, the skills are being evaluated increasingly out-of-distribution at each step, as the effective block heights increase beyond what is seen in training.

**Experimental setup.** For this experiment, we assume that the robot has access to a planner and additional domain knowledge as a part of this downstream task. We assume that the robot understands that at any step, the target block should be adjusted in the following manner. First, the target block's $x$- and $y$-position should be substituted with the bottom-most block's $x$- and $y$-position. Then, the target block's height should be substituted with the sum of all heights of the previous blocks, plus a small offset (1.5 cm). Effectively, this can be seen as treating each new step as stacking upon one, increasingly taller block. We leave the development of such a



(a) (b) (c) (d) (e)

Figure C.7: The block tower task. As previously, five blocks are initially available to the robot. However, after each stack attempt, the task does not reset. Instead, the block enumeration changes, so that the previous source block becomes the new target block. This happens four times, after which the task resets. The robot must complete each of the four individual steps successfully, as failure in any step renders the entire block tower task a failure. (a) Initial task scene. (b – d) Successful block stacks for intermediate attempts. (e) A successfully stacked block tower.

planner that can provide this additional information for future work, but it suffices for this experiment that this information is available.

**Block tower results.** Table C.6 shows the results for stacking the block tower. For this experiment, we use the same linear and nonlinear approaches and baselines from Sec. 5.7.1, including the training data ablations. Included is a ground truth policy with access to oracle information.

Overall, we see that the scale-lin approach does best for stacking a tower with five blocks, although a notable gap exists between the ground truth policy. However, a block tower success rate of 48.29% is not unreasonable, given that even the ground truth policy fails almost 30% of the time. The linear approaches are all comparable for the first stacking step, and for the second step with a $N_B = 3$ tall tower, three baseline methods slightly outperform scale-lin. However, for the last two steps, baseline approaches become markedly less performant, leading to scale-lin emerging as the best overall approach despite modest performance in an absolute sense. Each step requires successively greater extrapolation out of the training data, so an approach that can capture the smaller process well should perform best, assuming that this process also holds outside the training data. For the case of the block tower, this is generally true, so the skills learned by scale-lin are best suited for this downstream task despite the challenge of generalization to yet-unseen data.

For the nonlinear function class, performance across all approaches suffers beyond the first stacking step, where the CREST baselines outperform scale-nonlin. The challenge of extrapolation for nonlinear functions is evident here; the best linear approach for each step was better performing than any nonlinear approach (and

markedly so for taller towers). Thus, out-of-distribution generalization is not observed for any nonlinear approach, whereas scale-lin exhibits modest performance in this area.

For SCALE approaches, the skill selection rate is intriguing. The skill $\mathcal{K}_1$ does not contain the target block height, which is likely why it was only selected during the first block stack attempts. However, $\mathcal{K}_2$ continues to demonstrate its robustness, as it was used for all remaining block stack attempts in the linear case and for all attempts in the nonlinear case. Its inclusion of target block height in $\mathbf{A}_{\mathcal{K}_2}$ is in fact the reason this skill can extrapolate to taller towers. Like $\mathcal{K}_2$, $\mathcal{K}_3$ also contains the block height, but this skill was generally dominated, and thus it is not surprising it was not selected.

In summary, in addition to the benefits of SCALE described previously for task learning, the capability for SCALE to learn smaller, modular skills is evident in this experiment. Although out-of-distribution generalization was not observed in the nonlinear function class, we see that in principle SCALE does offer these benefits under certain conditions, such as in the linear case. We suggest that this aspect of causal learning is often overlooked for experiments that only concern single-task learning. However, the benefits of modularity become advantageous for re-using behaviors for downstream tasks at a later time in the robot's operational lifetime.

## C.11 Additional Details for Sensorless Peg-in-Hole Insertion Experiment

This appendix serves to provide greater detail for the peg insertion experiment that was described in Sec. 5.7.2.

**Reward function.** Our reward function consists of two terms: 1) a penalty based on the Euclidean distance of the peg from the hole, and 2) a bonus of 10 for successful insertion. We also add a regularization term based on the norm of the policy parameters. The task is considered solved if the final reward $R_f$ exceeds solved threshold $R_S = 8$.

**SCALE skill $\mathcal{K}_1$.** Unlike the other skills in $\boldsymbol{\mathcal{K}}_{peg}$ that were discovered by SCALE, skill $\mathcal{K}_1$ has an empty set of relevant variables. This is surprising as it is difficult to solve this task reliably without taking the help of one of the walls, in which case the wall should show up as a relevant variable. However, we observed that $\mathcal{K}_1$ actually localizes against 2 walls instead of just 1. Hence, when SCALE intervenes on any one of the two walls, the skill is still able to complete the assembly by taking advantage of the other wall. In other words, our assumption that the context space is disentangled does not hold in this case which leads to this erroneous relevant variable set. However, the precondition would limit where this skill would be applied, as skills $\mathcal{K}_{2-5}$ are generally more performant.

Table C.6: Results for re-using learned behaviors in a representative downstream task: stacking a block tower. The task solve percentage is shown for stacking a tower of at least $N_B$ blocks tall. The sequence is executed in one attempt, so a fully stacked tower ($N_B = 5$) requires 4 successful block stacking attempts. Methods within ±2% (the stochasticity of the simulator) of the best approach at each step are bold. For SCALE approaches, the skill selection rate at each step (not cumulative) is also shown. The abbreviation "mp" stands for monopolicy.

| Approach | | $N_B = 2$ | $N_B = 3$ | $N_B = 4$ | $N_B = 5$ |
|---|---|---|---|---|---|
| scale-lin (ours) | | **92.20%** (272) | 80.73% (222) | **65.23%** (167) | **48.29%** (113) |
| | $\mathcal{K}_1$ | 15.59% (46) | 0.00% (0) | 0.00% (0) | 0.00% (0) |
| | $\mathcal{K}_2$ | 84.07% (248) | 100.00% (275) | 100.00% (256) | 100.00% (234) |
| | $\mathcal{K}_3$ | 0.34% (1) | 0.00% (0) | 0.00% (0) | 0.00% (0) |
| monopolicy-lin-sk | | **93.22%** (275) | **87.23%** (239) | 55.08% (141) | 1.27% (3) |
| monopolicy-lin-all | | **93.56%** (276) | 76.36% (210) | 2.33% (6) | 0.00% (0) |
| crest-mp-lin-subs | | **93.20%** (274) | **85.40% (234)** | 5.84% (15) | 0.00% (0) |
| crest-mp-lin-all | | **93.92%** (278) | **85.51% (236)** | 5.84% (15) | 0.00% (0) |
| scale-nonlin (ours) | | 67.46% (199) | 2.55% (7) | 0.00% (0) | 0.00% (0) |
| | $\mathcal{K}_1$ | 0.00% (0) | 0.00% (0) | 0.00% (0) | 0.00% (0) |
| | $\mathcal{K}_2$ | 100.00% (295) | 100.00% (275) | 100.00% (256) | 100.00% (235) |
| | $\mathcal{K}_3$ | 0.00% (0) | 0.00% (0) | 0.00% (0) | 0.00% (0) |
| monopolicy-nonlin-sk | | 2.72% (8) | 0.00% (0) | 0.00% (0) | 0.00% (0) |
| monopolicy-nonlin-all | | 11.86% (35) | 0.00% (0) | 0.00% (0) | 0.00% (0) |
| crest-mp-nonlin-subs | | **84.75%** (250) | **27.37%** (75) | **0.78%** (2) | 0.00% (0) |
| crest-mp-nonlin-all | | 75.59% (223) | 11.31% (31) | 0.00% (0) | 0.00% (0) |
| ground-truth-policy | | 96.25% (282) | 90.48% (247) | 83.14% (212) | 69.96% (163) |

**SCALE skill selection.** For scale-lin, skills $\mathcal{K}_2$ (48.44%) and $\mathcal{K}_5$ (51.56%) were chosen nearly equally. Conversely, the skill selection was more distributed for the nonlinear case: $\mathcal{K}_2$: 46.48%, $\mathcal{K}_3$: 35.16%, $\mathcal{K}_4$: 3.91%, $\mathcal{K}_5$: 14.45%. For both approaches, $\mathcal{K}_1$ was not chosen as it was dominated by the other skills.

**Policy and training data ablations.** As with the block stacking domain, we conducted experiments with several policy functions and training data ablations. Table C.7 details the experimental results, which expand upon Tab. 5.4. In the linear function class, two policy functions were investigated: Bayesian ridge regression (B. ridge reg.) and ordinary least squares linear regression (OLS lin. reg.). An intercept term was used for both approaches, and the training data were unnormalized. In the nonlinear function class, experiments were conducted with a multilayer perceptron (MLP, 16x16x16 architecture using ReLU activations) and support vector regression with a radial basis function (RBF) kernel (SVR (RBF)). For the nonlinear policy functions, the training data were normalized. Methods with the "-all" suffix use the entire batch dataset. For the full-dimensional monopolicy approaches, the "-sk" suffix indicates that the same training data as SCALE was used (168 samples). The "-subs" suffix for the CREST baselines denotes that the batch dataset was randomly downselected to the same number of samples used by SCALE (168 samples).

Overall, we observe that the SCALE skills are highly performant across function class and policy function type. Moreover, SCALE significantly outperforms both the full-dimensional monopolicy approaches and the CREST baselines. Indeed, SCALE exceeds the performance of the baselines by around 30% for each policy function type. The success of SCALE is attributed to capturing the four modes in the data — localizing against each of the four walls — found by exploiting the underlying causal structure. The baselines, which are agnostic to such structure, do not leverage this property and are therefore limited. Unlike in the block stacking domain, we see that the effect of training data size does not necessarily yield an increase in performance for the baseline approaches.

## C.12 Sensorless Peg-in-Hole Insertion: Domain Shift Experiment

We evaluate the generalization capability of SCALE by evaluating it under a domain shift. All tasks are generated by uniformly sampling the relative position of the center of each wall with respect to the hole from a given range. The ranges used to generate the training and test tasks are specified in Tab. C.8. We transfer all the policies zero-shot to the test distribution. However, we do re-learn the preconditions of the scale-lin policies for the test distribution.

The evaluation results are summarized in Tab. C.9. All approaches witness a sharp drop in performance. This is expected as (a) the test tasks are not guaranteed

Table C.7: Task evaluation results for using the skill library $\boldsymbol{\mathcal{K}}_{peg}$ for peg insertion for a variety of policy functions and training data ablations. This table expands upon Tab. 5.4. Ctrl. is the approach control (skills or one monolithic policy). Fn. Cl. is the approach's function class. P. Fn. is the policy function. Task Solve % is the rate of task solves over the entire context space using the approach. Methods within ±2% (the stochasticity of the simulator) of the best approach are bold. $|\mathbf{A}|$ is the quantity of input variables used for the approach's policy. Data is the amount of training data used for the approach. The abbreviation "mp" stands for monopolicy.

| Approach | Ctrl. | Fn. Cl. | P. Fn. | Task Solve % | $\lvert\mathbf{A}\rvert$ | Data |
|---|---|---|---|---|---|---|
| scale-lin (ours) | 5 skills | Linear | B. ridge reg. | **96.48%** (247) | 0/1/1/1/1 | 168 |
| monopolicy-lin-sk | 1 policy | Linear | B. ridge reg. | 67.19% (172) | 8 | 168 |
| monopolicy-lin-all | 1 policy | Linear | B. ridge reg. | 62.50% (160) | 8 | 210 |
| crest-monopolicy-lin-subs | 1 policy | Linear | B. ridge reg. | 66.80% (171) | 1 | 168 |
| crest-monopolicy-lin-all | 1 policy | Linear | B. ridge reg. | 62.89% (161) | 1 | 210 |
| scale-lin-ols (ours) | 5 skills | Linear | OLS lin. reg. | **96.88%** (248) | 0/1/1/1/1 | 168 |
| monopolicy-lin-ols-sk | 1 policy | Linear | OLS lin. reg. | 50.78% (130) | 8 | 168 |
| monopolicy-lin-ols-all | 1 policy | Linear | OLS lin. reg. | 67.19% (172) | 8 | 210 |
| crest-monopolicy-lin-ols-subs | 1 policy | Linear | OLS lin. reg. | 63.67% (163) | 1 | 168 |
| crest-monopolicy-lin-ols-all | 1 policy | Linear | OLS lin. reg. | 60.55% (155) | 1 | 210 |
| scale-nonlin (ours) | 5 skills | Nonlin. | MLP | **88.67%** (227) | 0/1/1/1/1 | 168 |
| monopolicy-nonlin-sk | 1 policy | Nonlin. | MLP | 18.36% (47) | 8 | 168 |
| monopolicy-nonlin-all | 1 policy | Nonlin. | MLP | 12.89% (33) | 8 | 210 |
| crest-monopolicy-nonlin-subs | 1 policy | Nonlin. | MLP | 56.64% (145) | 1 | 168 |
| crest-monopolicy-nonlin-all | 1 policy | Nonlin. | MLP | 55.47% (142) | 1 | 210 |
| scale-nonlin-svr-rbf (ours) | 5 skills | Nonlin. | SVR (RBF) | **94.53%** (242) | 0/1/1/1/1 | 168 |
| monopolicy-nonlin-svr-rbf-sk | 1 policy | Nonlin. | SVR (RBF) | 53.52% (137) | 8 | 168 |
| monopolicy-nonlin-svr-rbf-all | 1 policy | Nonlin. | SVR (RBF) | 58.20% (149) | 8 | 210 |
| crest-mp-nonlin-svr-rbf-subs | 1 policy | Nonlin. | SVR (RBF) | 57.81% (148) | 1 | 168 |
| crest-mp-nonlin-svr-rbf-all | 1 policy | Nonlin. | SVR (RBF) | 60.94% (156) | 1 | 210 |

Table C.8: Training and test distributions of the domain shift experiment in the sensorless peg-in-hole domain. The relative position of the center of each of the 4 walls is uniformly sampled from the given (min, max) range. The ranges used to generate test tasks are more than double the ranges used to generate training tasks in the domain shift experiment. All values are in meters.

| | Train | | | | Test | | | |
|---|---|---|---|---|---|---|---|---|
| | $x$-min | $x$-max | $y$-min | $y$-max | $x$-min | $x$-max | $y$-min | $y$-max |
| Wall 1 | 0.01 | 0.05 | -0.02 | 0.02 | -0.04 | 0.10 | -0.07 | 0.07 |
| Wall 2 | -0.02 | 0.02 | -0.05 | -0.01 | -0.07 | 0.07 | -0.10 | 0.07 |
| Wall 3 | -0.02 | 0.02 | 0.01 | 0.05 | -0.07 | 0.07 | -0.04 | 0.10 |
| Wall 4 | -0.05 | -0.01 | -0.02 | 0.02 | -0.10 | -0.04 | -0.07 | 0.07 |

Table C.9: Task evaluation results under domain shift for sensorless peg-in-hole insertion. We evaluate only linear policies as nonlinear policies perform worse in this domain. Table columns are as described in Tab. 5.4.

| Approach | Ctrl. | Fn. Cl. | Task Solve % | $\lvert\mathbf{A}\rvert$ |
|---|---|---|---|---|
| scale-lin (ours) | 5 skills | Linear | **64.84%** | 0/1/1/1/1 |
| monopolicy-lin-all | 1 policy | Linear | 44.92% | 8 |
| crest-monopolicy-lin-all | 1 policy | Linear | 39.83% | 1 |

to be feasible and (b) the ranges used to generate the test task are more than double those used in training. However, our multi-skill approach scale-lin performs much better than the baselines. This highlights a key benefit of learning multiple skills. A skill may perform well on the training distribution, but it can be rendered invalid due to an unforeseen domain shift. Having a repertoire of different skills allows the robot to still complete the task by switching to a different skill. This makes our multi-skill approach more robust than single-skill approaches.

## C.13 A Primer on Causality

For readers who are unfamiliar with causality, this appendix serves as a gentle "on-ramp" for understanding SCALE.

*What's a data generating process?* A data generating process (DGP) is a dynamical process that generates data in a physical system. The process is usually described by variables that characterize the system. Consider the following examples: turning a light switch on a lamp to illuminate the lightbulb; inserting a car key into an ignition and turning the starter to start a vehicle; rain showers causing rainfall. These examples can be considered data generating processes if system variables were

instrumented, such as instrumenting a rain gauge to measure rainfall.

*What's a Structural Causal Model?* A Structural Causal Model (SCM) [16, 17] is a representation of a data generating process. Usually, the SCM consists of variables of a system, a graph (which is usually directed with no cycles) that describes how the variables depend on each other, and functions that describe how each variable is characterized based on that variable's causes. These functions are also called structural equations or functional equations, and each function will have its own noise variable. Noise variables (also referred to as exogenous variables) are generally jointly independent.

*What's an example of an SCM, and how can it be used?* Consider the following example of SCM $\mathfrak{C}_1$:

- $X \coloneqq N_X$
- $Y \coloneqq 2X + N_Y$

Here, $X$ and $Y$ are variables of our SCM, and $N_X$ and $N_Y$ are the noise terms. This SCM can also be characterized by its underlying graph, where $X \to Y$ because $X$ is a cause of $Y$. For this example, consider that $N_X$ and $N_Y$ are (independently) sampled from the uniform distribution from -10 to +10. Then, if $N_X = 2$ and $N_Y = -3$, then by the mechanics of the SCM, $X = 2$ and therefore $Y = 1$.

We now introduce the concept of an intervention, where we *set* the value of a variable to be a particular value (usually regardless of its causes or noise variables), holding all other variables equal. We can formalize this using the *do* operator [16]. Thus, an intervention $do(Y = 5)$ means that no matter what value $N_X$, $X$, or $N_Y$ take, $Y = 5$. In the previous example, under this intervention, if $N_X = 2$, then $X = 2$, but $Y = 5$ (and not 1). This type of intervention is called "hard" since it induces a structural change; other intervention types are possible, such as "soft" interventions where the functional equation of a variable changes (but not its parents).

*What's the difference between a DGP and an SCM?* In the case where the SCM captures the DGP exactly, there is no difference. However, often times we wish to learn the data generating process, and the SCM encodes the knowledge of the DGP that is currently known. In these cases, the SCM is an approximation of the underlying DGP in the physical world.

*In SCALE, what's the Data Generating Region and how does it differ from a DGP?* The Data Generating Region (DGR) introduced by SCALE provides *locality* to the data generating process. Consider a physical system where SCM $\mathfrak{C}_1$ co-exists with the following new SCM, $\mathfrak{C}_2$:

- $X \coloneqq 3N_X$
- $Z \coloneqq -X + N_Z$

However, it is also noticed that according to a fourth variable $A$, when $A < 0$, $\mathfrak{C}_1$ applies, whereas when $A > 0$, $\mathfrak{C}_2$ applies. The condition where these causal models

apply is equivalent to how the DGR specifies *where* particular skills are defined in the context space. Note that $X$, $Y$, and $Z$ are not needed to define where the models apply (only $A$). A learning algorithm could use all four variables to specify where the models apply, but a minimal, compressed representation only requires one ($A$).

Moreover, $Z$ and $A$ are not needed to specify the mechanics of $\mathfrak{C}_1$ (similarly, $Y$ and $A$ for $\mathfrak{C}_2$). This is similar to how SCALE learns which variables of the context space to use for modeling the skill policy. Even though a learner could potentially use all four variables it knows about, irrelevant variables are not needed in a minimal representation.

*I'm interested in learning more about causality. Where should I start?* There are many important and useful textbooks in this area. We use Pearlian causality and SCMs as the basis of our formalism, so we recommend the reader reviewing *Causality* (Pearl 2009, 2nd edition) [16], in particular, chapters 1–3. Then, we recommend the reader reviewing *Elements of Causal Inference* (Peters, Janzing, and Schölkopf, 2017) [17], in particular, chapters 1, 3, and 6.

# D Appendix for Learning By Doing

## D.1 More Details on Track CHEM

The following provides additional details on Track CHEM of the competition.

**Background on chemical reactions** Parts of the following text are taken from Peters et al. [193]. A general reaction (as given in Wilkinson [335], for example) takes the form

$$m_1R_1 + m_2R_2 + \ldots + m_rR_r \to n_1P_1 + n_2P_2 + \ldots + n_pP_p,$$

where $r$ is the number of reactants and $p$ the number of products. Both $R_i$ and $P_j$ can be thought of as molecules and are often called species. The coefficients $m_i$ and $n_j$ are positive integers, called stoichiometries.

In mass-action kinetics [201], one usually considers the concentration $[X]$ of a species $X$, the square parentheses indicating that one refers to the concentration rather than to the integer amount of a given species. The concentration $[X]$ changes over time (but to simplify notation, we sometimes omit the notational dependence on $t$). The law of mass-action allows one to convert the above equations into a system of ODEs over the concentrations of species. Formally, it states: *The instantaneous rate of each reaction is proportional to the product of each of its reactants raised to the power of its stoichiometry.* To better understand how this can be applied to transform reaction equations into a system of ODEs, it may help to consider an example. The Lotka-Volterra predator-pray model [336] can be expressed in terms of reactions of the form

$$A \xrightarrow{k_1} 2A \tag{D.1}$$

$$A + B \xrightarrow{k_2} 2B \tag{D.2}$$

$$B \xrightarrow{k_3} \emptyset, \tag{D.3}$$

where $A$ and $B$ describe abundance of prey and predators, respectively. In this model, the prey reproduce by themselves (D.1), but the predators require abundance of prey

for reproduction, see (D.2). After some time, also the predators die (D.3). The coefficients $k_1, k_2$, and $k_3$ indicate the rates, with which the reactions happen (the larger the rates, the faster the reactions). Applying the law of mass-action yields the following system of ordinary differential equations (ODEs)

$$\tfrac{\mathrm{d}}{\mathrm{d}t}[A] = k_1[A] - k_2[A][B] \tag{D.4}$$
$$\tfrac{\mathrm{d}}{\mathrm{d}t}[B] = k_2[A][B] - k_3[B]. \tag{D.5}$$

**Chemical reactions of the data-generating process** The data-generating process is illustrated in Figure 6.1. The corresponding chemical reactions are given by

$$\begin{aligned}
Z_1 + Z_2 &\xrightarrow{k_1} Z_9 & Z_{10} &\xrightarrow{k_5} Y & Z_{13} &\xrightarrow{k_9} Z_1 + Z_2\\
Z_3 + Z_4 &\xrightarrow{k_2} Z_{10} & Z_{11} &\xrightarrow{k_6} Y & Z_{14} &\xrightarrow{k_{10}} Z_5 + Z_6.\\
Z_5 + Z_6 &\xrightarrow{k_3} Z_{11} & Z_9 + Y &\xrightarrow{k_7} Z_{13} & &\\
Z_7 + Z_8 &\xrightarrow{k_4} Z_{12} & Z_{12} + Y &\xrightarrow{k_8} Z_{14} & &
\end{aligned}$$

This system can be converted using the law of mass action resulting in the following ODE system.

$$\begin{aligned}
\tfrac{\mathrm{d}}{\mathrm{d}t}[Z_1] &= -k_1[Z_1][Z_2] + k_9[Z_{13}]\\
\tfrac{\mathrm{d}}{\mathrm{d}t}[Z_2] &= -k_1[Z_1][Z_2] + k_9[Z_{13}]\\
\tfrac{\mathrm{d}}{\mathrm{d}t}[Z_3] &= -k_2[Z_3][Z_4]\\
\tfrac{\mathrm{d}}{\mathrm{d}t}[Z_4] &= -k_2[Z_3][Z_4]\\
\tfrac{\mathrm{d}}{\mathrm{d}t}[Z_5] &= -k_3[Z_5][Z_6] + k_{10}[Z_{14}]\\
\tfrac{\mathrm{d}}{\mathrm{d}t}[Z_6] &= -k_3[Z_5][Z_6] + k_{10}[Z_{14}]\\
\tfrac{\mathrm{d}}{\mathrm{d}t}[Z_7] &= -k_4[Z_7][Z_8]\\
\tfrac{\mathrm{d}}{\mathrm{d}t}[Z_8] &= -k_4[Z_7][Z_8]
\end{aligned}$$

$$\begin{aligned}
\tfrac{\mathrm{d}}{\mathrm{d}t}[Z_9] &= k_1[Z_1][Z_2] - k_7[Z_9][Y]\\
\tfrac{\mathrm{d}}{\mathrm{d}t}[Z_{10}] &= k_2[Z_3][Z_4] - k_5[Z_{10}]\\
\tfrac{\mathrm{d}}{\mathrm{d}t}[Z_{11}] &= k_3[Z_5][Z_6] - k_6[Z_{11}]\\
\tfrac{\mathrm{d}}{\mathrm{d}t}[Z_{12}] &= k_4[Z_7][Z_8] - k_8[Z_{12}][Y]\\
\tfrac{\mathrm{d}}{\mathrm{d}t}[Z_{13}] &= k_7[Z_9][Y] - k_9[Z_{13}]\\
\tfrac{\mathrm{d}}{\mathrm{d}t}[Z_{14}] &= k_8[Z_{12}][Y] - k_{10}[Z_{14}]\\
\tfrac{\mathrm{d}}{\mathrm{d}t}[Y] &= k_5[Z_{10}] + k_6[Z_{11}] - k_7[Z_9][Y] - k_8[Z_{12}][Y]
\end{aligned}$$

**Evaluation** For each of the systems, $i = 1, \ldots, 12$, partipants were asked to provide control input for 50 initial values. Participants' control inputs were evaluated by running the data-generating process for each of the provided controls to compute the following loss for each system[1]

$$J_i := \frac{1}{50}\sum_{k=1}^{50}\left(\sqrt{\frac{1}{40}\int_{40}^{80}\left(Z_{15}^{i,k}(t) - y_*^{i,k}\right)^2 dt} + c\cdot\sqrt{\frac{||u^{i,k}||_2^2}{8}}\right), \tag{D.6}$$

where $c = \frac{1}{20}$ and $u^{i,k} \in \mathbb{R}^p$ is the control input provided by the participant corresponding to the $k$th initial condition in the $i$th system. The process $Y^{i,k}$ of course

[1] The integrals are approximated numerically.

depends on the provided input, $u^{i,k}$, even though this is not made explicit in the notation.

### D.1.1 CHEM results

The following table summarizes the results from Track CHEM. The keywords describing participants' solutions were chosen by the organizers based on participants' summaries of their solutions. *Oracle* corresponds to a solution using access to the true data generating process. *Oracle*$^e$ corresponds to a solution generated with access to the true data generating process, but using only the expensive controls (see Section 6.4). *Zero* corresponds to a solution choosing $U \equiv 0$ for every system and initial condition.

| Team name | Score | Place | Keywords |
|---|---|---|---|
| *Oracle* | 0.0872 | | |
| Ajoo | 0.0890 | 1st | Sparse estimation of graph<br>Direct estimation of a function in $\mathcal{F}$ |
| *Oracle*$^e$ | 0.1450 | | |
| TeamQ | 0.3385 | 2nd | Neural network prediction of target |
| GuineaPig | 0.3386 | 3rd | Neural network prediction of target |
| *Zero* | 0.9686 | | |

## D.2 More Details on Track ROBO

**Evaluation** For each evaluated system $(F^i, \theta^i, A^i),\ i \in \{1, \ldots, 24\}$ and repetition $k \in \{1, \ldots, 10\}$, running the system using the participants' controller and comparing the realized end-effector trajectory against a target process $z_*^{i,k} : [0, 2] \to \mathbb{R}^2$ the following loss is computed[1]

$$J_{i,k} := b_{i,k} \cdot \int_0^2 ||Z^{i,k}(t) - z_*^{i,k}(t)||_2^2 dt + c_{i,k} \cdot \int_0^2 U^{i,k}(t)^\top U^{i,k}(t) dt, \qquad \text{(D.7)}$$

where $b_{i,k}$ and $c_{i,k}$ are scaling constants which are selected such that $J_{i,k} = 100$ when no controls are applied and $J_{i,k} = 1$ if an oracle LQR-controller is used, that is, an LQR-controller using the true robot dynamics and interface function. If $J_{i,k}$ is smaller than 1, it is improving on the oracle LQR-controller; if it is larger than 100 the performance is worse than when doing nothing. We clip all scores at 100 before averaging them. The scaling is meant to ensure that losses are comparable across each repetition.

For the (preliminary) leaderboard, which was updated during the competition, only 12 systems were evaluated, and the mean loss across those systems (and all corresponding repetitions) was shown on the leaderboard. For the final ranking, the

| robot | dynamics ($F$) | specification ($\theta$) | interface ($A$) |
|---|---|---|---|
| great-devious-beetle | Rotational2 | $\theta^{\text{gr-be}}$ | $A^{\text{de}} = \mathbf{1}_{2\times 2}$ |
| great-vivacious-beetle | | $\theta^{\text{gr-be}}$ | $A^{\text{vi}} \in \mathbb{R}^{2\times 2}$ |
| great-mauve-beetle | | $\theta^{\text{gr-be}}$ | $A^{\text{ma}} \in \mathbb{R}^{2\times 3}$ |
| great-wine-beetle | | $\theta^{\text{gr-be}}$ | $A^{\text{wi}} \in \mathbb{R}^{2\times 4}$ |
| rebel-devious-beetle | | $\theta^{\text{re-be}}$ | $A^{\text{de}} = \mathbf{1}_{2\times 2}$ |
| rebel-vivacious-beetle | | $\theta^{\text{re-be}}$ | $A^{\text{vi}} \in \mathbb{R}^{2\times 2}$ |
| rebel-mauve-beetle | | $\theta^{\text{re-be}}$ | $A^{\text{ma}} \in \mathbb{R}^{2\times 3}$ |
| rebel-wine-beetle | | $\theta^{\text{re-be}}$ | $A^{\text{wi}} \in \mathbb{R}^{2\times 4}$ |
| talented-ruddy-butterfly | Rotational3 | $\theta^{\text{ta-bu}}$ | $A^{\text{ru}} = \mathbf{1}_{3\times 3}$ |
| talented-steel-butterfly | | $\theta^{\text{ta-bu}}$ | $A^{\text{st}} \in \mathbb{R}^{3\times 3}$ |
| talented-zippy-butterfly | | $\theta^{\text{ta-bu}}$ | $A^{\text{zi}} \in \mathbb{R}^{3\times 4}$ |
| talented-antique-butterfly | | $\theta^{\text{ta-bu}}$ | $A^{\text{an}} \in \mathbb{R}^{3\times 6}$ |
| thoughtful-ruddy-butterfly | | $\theta^{\text{th-bu}}$ | $A^{\text{ru}} = \mathbf{1}_{3\times 3}$ |
| thoughtful-steel-butterfly | | $\theta^{\text{th-bu}}$ | $A^{\text{st}} \in \mathbb{R}^{3\times 3}$ |
| thoughtful-zippy-butterfly | | $\theta^{\text{th-bu}}$ | $A^{\text{zi}} \in \mathbb{R}^{3\times 4}$ |
| thoughtful-antique-butterfly | | $\theta^{\text{th-bu}}$ | $A^{\text{an}} \in \mathbb{R}^{3\times 6}$ |
| great-piquant-bumblebee | Prismatic | $\theta^{\text{gr-bu}}$ | $A^{\text{pi}} = \mathbf{1}_{2\times 2}$ |
| great-bipedal-bumblebee | | $\theta^{\text{gr-bu}}$ | $A^{\text{bi}} \in \mathbb{R}^{2\times 2}$ |
| great-impartial-bumblebee | | $\theta^{\text{gr-bu}}$ | $A^{\text{im}} \in \mathbb{R}^{2\times 3}$ |
| great-proficient-bumblebee | | $\theta^{\text{gr-bu}}$ | $A^{\text{pr}} \in \mathbb{R}^{2\times 4}$ |
| lush-piquant-bumblebee | | $\theta^{\text{lu-bu}}$ | $A^{\text{pi}} = \mathbf{1}_{2\times 2}$ |
| lush-bipedal-bumblebee | | $\theta^{\text{lu-bu}}$ | $A^{\text{bi}} \in \mathbb{R}^{2\times 2}$ |
| lush-impartial-bumblebee | | $\theta^{\text{lu-bu}}$ | $A^{\text{im}} \in \mathbb{R}^{2\times 3}$ |
| lush-proficient-bumblebee | | $\theta^{\text{lu-bu}}$ | $A^{\text{pr}} \in \mathbb{R}^{2\times 4}$ |

Table D.1: Overview of the 24 robot systems used in Track ROBO. Here, $\theta^*$ refers to the robot specification (link lengths and masses, moments of inertia, friction coefficients, and locations of link center of masses) and $A^* \in \mathbb{R}^{q\times p}$ parametrizes the linear interface function; values are chosen at random, while the above table indicates which properties where shared across which robot systems. We refer to Appendix D.2.1 for details on the 2- and 3-link rotational robots' dynamics and to Appendix D.2.2 for details on the 2-link prismatic robots' dynamics.

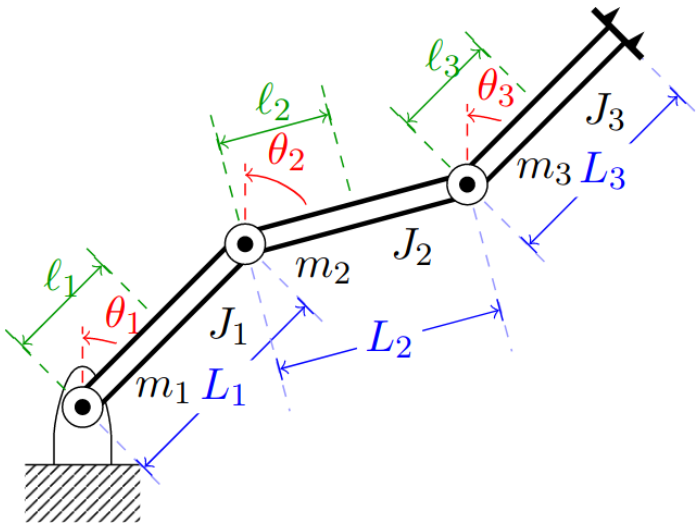



Figure D.1: Diagram of a 3-link rotational robotic arm.

average loss across the 12 held-out systems (and all corresponding repetitions) was used.

### D.2.1 Rotational robots

We consider two types of rotational robotic manipulators: open chain planar manipulators with three (cf. Figure D.1) and two revolute joints. Joints can be controlled by applying a voltage signal to a DC motor located in the joint, which creates a torque.

We begin with discussing the 3-link manipulator and then show the simplified version for the 2-link variant. Let $Z(t) = [\theta_1(t), \theta_2(t), \theta_3(t), \omega_1(t), \omega_2(t), \omega_3(t)]^{\mathrm{T}} \in \mathbb{R}^6$ be the state of the robotic arm, consisting of joint angles $(\theta_1, \theta_2, \theta_3)$ and corresponding angular velocities $(\omega_1, \omega_2, \omega_3)$. Let $U(t) = [\tau_1(t), \tau_2(t), \tau_3(t)]^{\mathrm{T}} \in \mathbb{R}^3$ be the input joint torques $(\tau_1, \tau_2, \tau_3)$.

The robotic arm is characterized by the following properties for the links, $i \in \{1, .., 3\}$:

- $m_i$ is the link mass,
- $J_i$ is the link rotational moment of inertia,
- $L_i$ is the link length,
- $\ell_i$ is the location of the link center of mass, and
- $c_i$ is the joint rotational friction coefficient.

The second-order dynamic system of the robotic arm is expressed through the following set of first-order equations [337]:

$$\begin{aligned}
\tfrac{\mathrm{d}}{\mathrm{dt}}[\theta_1] &= \omega_1 \\
\tfrac{\mathrm{d}}{\mathrm{dt}}[\theta_2] &= \omega_2 \\
\tfrac{\mathrm{d}}{\mathrm{dt}}[\theta_3] &= \omega_3 \\
\tfrac{\mathrm{d}}{\mathrm{dt}}[\omega_1] &= \alpha_1 \\
\tfrac{\mathrm{d}}{\mathrm{dt}}[\omega_2] &= \alpha_2 \\
\tfrac{\mathrm{d}}{\mathrm{dt}}[\omega_3] &= \alpha_3
\end{aligned}$$

The joint acceleration terms $\alpha = [\alpha_1, \alpha_2, \alpha_3]^{\mathrm{T}}$ are determined via:

$$\alpha = M^{-1}\left(\tau - C\omega - N\right) \tag{D.8}$$

where the inertia matrix $M$, Coriolis matrix $C$, and external force vector $N$ are:

$$M = \begin{bmatrix} M_{11} & M_{12}\cos(\theta_2 - \theta_1) & M_{13}\cos(\theta_3 - \theta_1) \\ M_{12}\cos(\theta_2 - \theta_1) & M_{22} & M_{23}\cos(\theta_3 - \theta_2) \\ M_{13}\cos(\theta_3 - \theta_1) & M_{23}\cos(\theta_3 - \theta_2) & M_{33} \end{bmatrix},$$

$$C = \begin{bmatrix} 0 & C_{12}\sin(\theta_2 - \theta_1)\omega_2 & C_{13}\sin(\theta_3 - \theta_1)\omega_3 \\ C_{21}\sin(\theta_2 - \theta_1)\omega_1 & 0 & C_{23}\sin(\theta_3 - \theta_2)\omega_3 \\ -C_{13}\sin(\theta_3 - \theta_1)\omega_1 & C_{32}\sin(\theta_3 - \theta_2)\omega_2 & 0 \end{bmatrix}, \text{ and}$$

$$N = \begin{bmatrix} N_1\sin(\theta_1) + c_1\omega_1 \\ N_2\sin(\theta_2) + c_2\omega_2 \\ N_3\sin(\theta_3) + c_3\omega_3 \end{bmatrix}$$

with coefficients

$$
\begin{aligned}
M_{11} &= m_1\ell_1^2 + J_1 + (m_2 + m_3)L_1^2 \\
M_{12} &= (m_2\ell_2 + m_3L_2)L_1 \\
M_{13} &= m_3\ell_3L_1 \\
M_{22} &= m_2\ell_2^2 + J_2 + m_3L_2^2 \\
M_{23} &= m_3\ell_3L_2 \\
M_{33} &= m_3\ell_3^2 + J_3 \\
C_{12} &= -(m_2\ell_2 + m_3L_2)L_1 \\
C_{13} &= -m_3\ell_3L_1 \\
C_{21} &= (m_2\ell_2 + m_3L_2)L_1 \\
C_{23} &= -m_3\ell_3L_2 \\
C_{32} &= m_3\ell_3L_2 \\
N_1 &= -(m_1\ell_1 + (m_2 + m_3)L_1)g \\
N_2 &= -(m_2\ell_2 + m_3L_2)g \\
N_3 &= -m_3\ell_3g
\end{aligned}
$$

and $g$ is gravitational acceleration.

For the 2-link robot, we omit all terms that correspond to the third joint. That is, the acceleration $\alpha = [\alpha_1, \alpha_2]^{\mathrm{T}}$ is still given by (D.8), but we now need to adapt $M$, $C$, and $N$. When only considering two joints, we have

$$
\begin{aligned}
M &= \begin{bmatrix} M_{11} & M_{12}\cos(\theta_2 - \theta_1) \\ M_{12}\cos(\theta_2 - \theta_1) & M_{22} \end{bmatrix} \\
C &= \begin{bmatrix} 0 & C_{12}\sin(\theta_2 - \theta_1)\omega_2 \\ C_{21}\sin(\theta_2 - \theta_1)\omega_1 & 0 \end{bmatrix} \\
N &= \begin{bmatrix} N_1\sin(\theta_1) + c_1\omega_1 \\ N_2\sin(\theta_2) + c_2\omega_2 \end{bmatrix}
\end{aligned}
$$

with coefficients

$$
\begin{aligned}
M_{11} &= m_1\ell_1^2 + J_1 + m_2L_1^2 \\
M_{12} &= m_2\ell_2L_1 \\
M_{22} &= m_2\ell_2^2 + J_2 \\
C_{12} &= -m_2\ell_2^2L_1 \\
C_{21} &= m2\ell_2L_1 \\
N_1 &= -(m_1\ell_1 + m_2L_1)g \\
N_2 &= -m_2\ell_2g.
\end{aligned}
$$

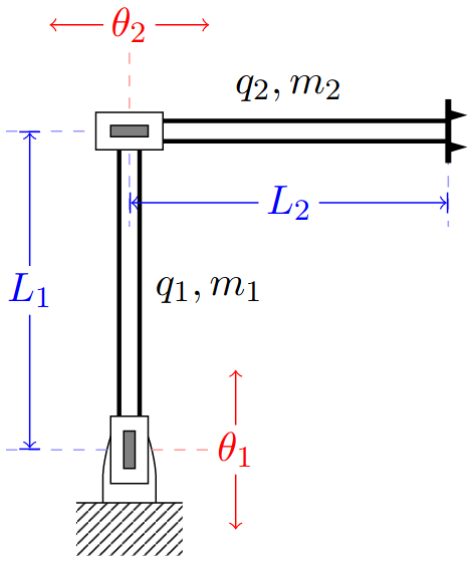



Figure D.2: Diagram of the 2-link prismatic robot arm.

*Remark.* Note that in the open source code, we define the joint angles of the 2-link manipulator with respect to each other instead of with respect to the vertical axis (Murray et al. [338]). Hence, the equations slightly differ from the ones presented above. However, the dynamics are the same as described herein.

### D.2.2 Prismatic robot

Besides the two versions of rotational robots, we also consider a 2-link prismatic robot arm (Figure D.2). This idealized prismatic robot is actuated by prismatic joints that change the link lengths, such that $L_i = q_i + \theta_i$, where $q_i$ represents the link length at zero joint input. Although the link length changes, we assume the link mass $m_i$ remains constant.

Due to the lateral instead of rotational movements, the dynamics of this robot are considerably simpler. The joint acceleration terms $\alpha = [\alpha_1, \alpha_2]^{\mathrm{T}}$ are given by

$$\alpha = M^{-1}\left(\tau - N\right). \tag{D.9}$$

The equation is similar to (D.8), but without the Coriolis term since there are only lateral movements. Mass matrix and external force vector are $M = \begin{bmatrix} m_1 + m_2 & 0 \\ 0 & m_2 \end{bmatrix}$ and $N = \begin{bmatrix} g(m_1 + m_2) \\ 0 \end{bmatrix}$, respectively.

### D.2.3 ROBO results

The results from Track ROBO are summarized below. The keywords were chosen by the organizers based on the participants' descriptions of their approaches. The score function is standardized such that a value of 1 corresponds to the performance of an oracle LQR-controller using the true system (optimizing only the trajectory, not the cost). A score of 100 corresponds to the zero solution, $U \equiv 0$.

| Team name | Score | Place | Keywords |
|---|---|---|---|
| Ajoo | 0.918 | 1st | Estimation of robot dynamics |
| TeamQ | 16.121 | 2nd | Neural network prediction |
| jmunozb | 29.539 | 3rd | Linear system approximation<br>Regression with polynomial features |

# E Appendix for ACL

## E.1 Related Works

Our work relates to prior work in cognitive science, curriculum learning, and active causal learning.

**Automated Learning in Children** Related work in cognitive science suggests that child learning resonates with the Goldilocks principle: children opt for information that is neither too easy nor too complex, but "just right" and moderately predictable [339–341]. Furthermore, children as learners seem capable of monitoring the "zone of proximity" between their current capabilities and the goal at hand; enabling them to progress from what they cannot do to what they can learn to do with the interventions of an adult or a teacher [342]. Infants allocate their visual attention based on surprise, predictability and learning progress of the environmental structure [250]. 4- to 6-year-old children use their improvement over time to decide whether to persist on a challenging goal on their own [343]. And by age 7, children ask questions that yield high information gain and more constraint-seeking questions when the problems are difficult [344]. However, there are no systematic studies showing that young children can indeed construct an appropriate curriculum in order to master complex goal-directed tasks, nor studies that would allow a comparison between children and artificial agents.

**Curriculum Learning** [253] introduced the concept of curriculum learning and proposed that an effective approach to learning involves the provision of examples that strike a balance between being neither overly simple nor excessively challenging. This concept is further supported by theoretical proofs in reinforcement learning [272]. Various metrics have been proposed to measure task difficulty, including the transferability of models trained on other tasks to the current task [345], complexity-driven progress, and loss-driven progress [267]. [252, 254] proposed a framework for automatic curriculum learning through self-play, while [255] proposed a framework utilizing automatic goal generation. [300] introduced the Unsupervised Environment Design paradigm, formalizing variation in environments by parameters, along with

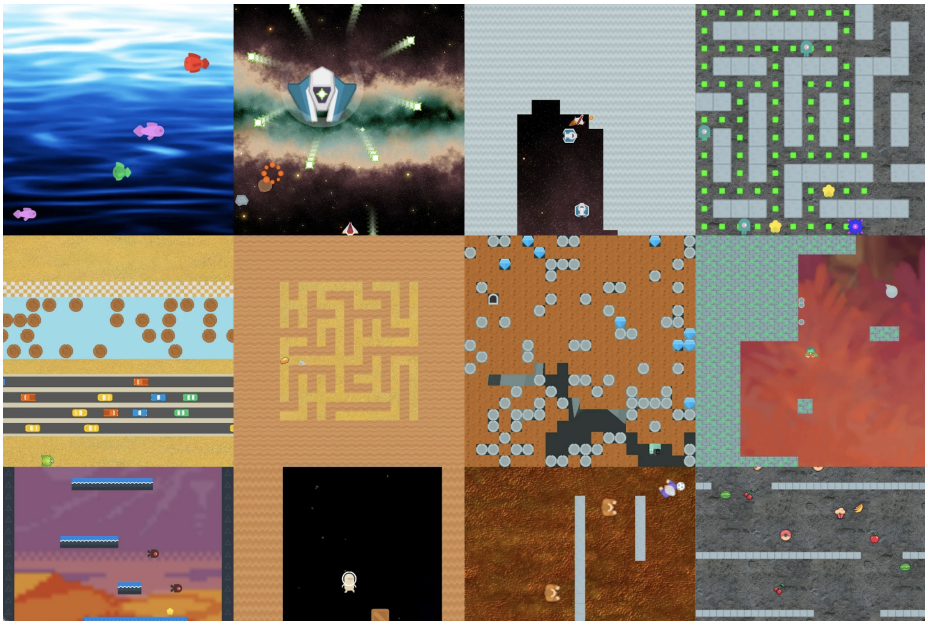


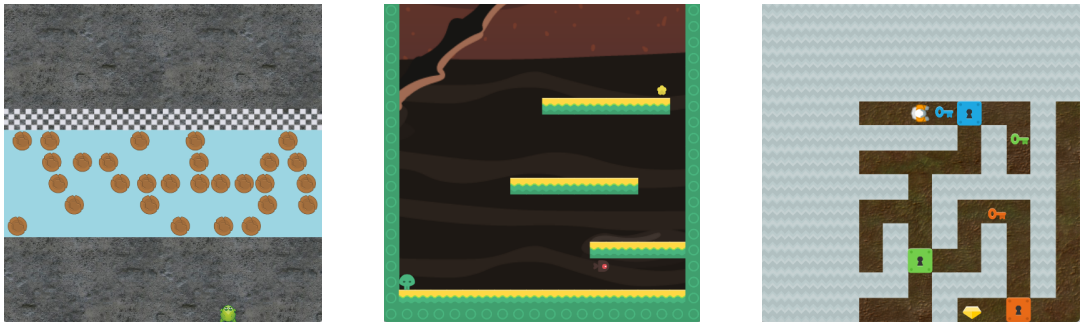
Figure E.1: An illustration of all Procgen environments

the PAIRED algorithm that produced an implicit curriculum. Prioritized Level Replay [307] also implicitly learns a curriculum for training an RL agent not through environment adaptation, but through judicious selection of past levels to replay based on learning potential. Later, [307] unified these two concepts under the Dual Curriculum Design framework with Robust PLR as a representative algorithm, and [306] introduced ACCEL, an evolutionary-based approach for editing environments to form a curriculum. Instead of proposing a new automatic curriculum learning framework, we draw inspiration from child development studies. We use level competence as an auxiliary reward within a curriculum inspired by [270]. Our results show that employing auxiliary rewards mitigates RL agent catastrophic forgetting, leading to faster and improved convergence.

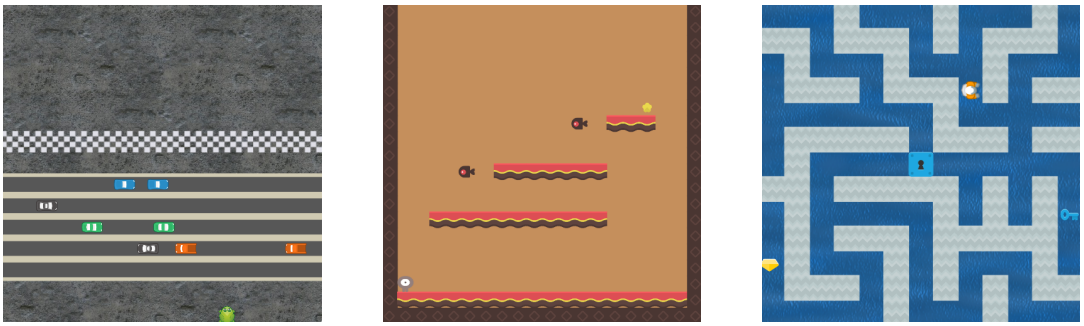
**Active Causal Learning** In RL curriculum development, we perform environmental interventions to enable swift achievement of learning objectives, resembling active causal learning [346–356]. However, curriculum learning in RL differs from active causal learning in that it dispenses with explicit causal structure learning and prioritizes targeted causal interventions, disregarding non-essential variables or edges.

## E.2 Procgen Environments

There are 12 different Procgen environments. Figure E.1 taken from 243 showing all available Procgen environments.



(a) Leaper Log (b) Climber Platform (c) Heist Key



(d) Leaper Car (e) Climber Enemy (f) Heist Size

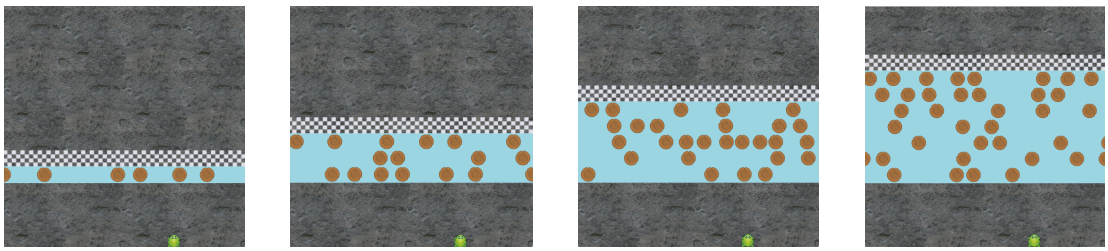
Figure E.2: We selected 3 Procgen environments with 2 different axes each and varied the level of difficulty across the axes within a game. (a)-(f) show the goal levels for the selected Procgen games.

### E.2.1 Our Adapted Procgen Environments

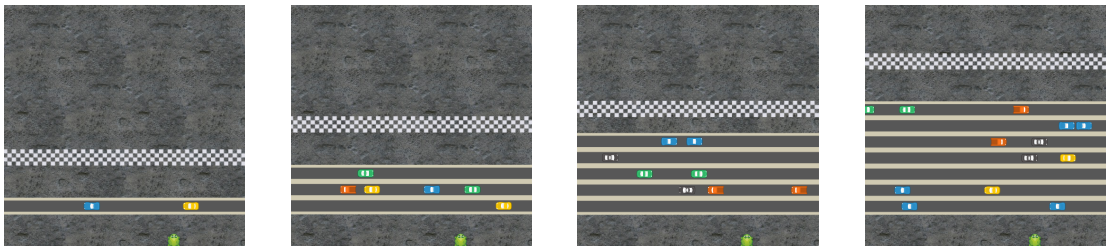
The goal of each game is provided in Table E.2. Each game includes three levels: Level 1 is the easiest level, whereas Level 3 is the most difficult level. Participants were given the goal to win Level 3. In addition, a level more difficult than Level 3, Level 4, was presented to a random subset of the participants. Participants did not need to learn Level 4 to achieve their goal of winning Level 3. Table E.1 describes the levels of the different games. We also include visual examples of the levels in Figure E.3 for the game Leaper across its two different axes, log lanes and car lanes.

Table E.1: Description of levels used in selected Procgen games of Leaper, Climber, and Heist. These games were chosen to vary certain aspects of each game based on a particular axis. The levels increase in difficulty from Level 1 through Level 4. The goal level to complete is Level 3; Level 4 is the most challenging level, which is not necessarily needed to be solved to complete Level 3.

| Game | Axis | Level 1 | Level 2 | Level 3 (Goal level) | Level 4 |
|---|---|---|---|---|---|
| Leaper | Log | 1 log lane | 3 log lanes | 5 log lanes | 7 log lanes |
| Leaper | Car | 1 car lane | 3 car lanes | 5 car lanes | 7 car lanes |
| Climber | Enemy | 0 enemies | 1 enemy | 2 enemies | 3 enemies |
| Climber | Platform | 1 platform | 2 platforms | 3 platforms | 4 platforms |
| Heist | Size | small | medium | large | extra large |
| Heist | Key | 1 key/lock | 2 keys/locks | 3 keys/locks | 4 keys/locks |



(a) Level 1 (b) Level 2 (c) Level 3 (Goal level) (d) Level 4



(e) Level 1 (f) Level 2 (g) Level 3 (Goal level) (h) Level 4

Figure E.3: (a)-(d) Levels of difficulty for the game Leaper with the number of log lanes as the difficulty axis. (e)-(f) Levels of difficulty for the Procgen game Leaper with the number of car lanes as the difficulty axis.

| Game | Goal |
|---|---|
| Leaper Log | Cross the finish line and avoid going in the water as log lanes increase. |
| Leaper Car | Cross the finish line and avoid getting hit by a car as car lanes increase. |
| Climber Enemy | Reach the jewel and avoid enemies as number of enemies increase. |
| Climber Platform | Reach the jewel and avoid a single enemy as number of platforms increase. |
| Heist Size | Reach the final jewel by getting key and unlocking a lock as size of maze increases. |
| Heist Key | Reach the final jewel by getting key and unlocking locks as number of locks and keys increase. |

Table E.2: We provide the goals for each of the Procgen games. Participants were never told the rules of the game and had to learn how to win the game through their own learning.

## E.3 Automated Curriculum Learning in Children

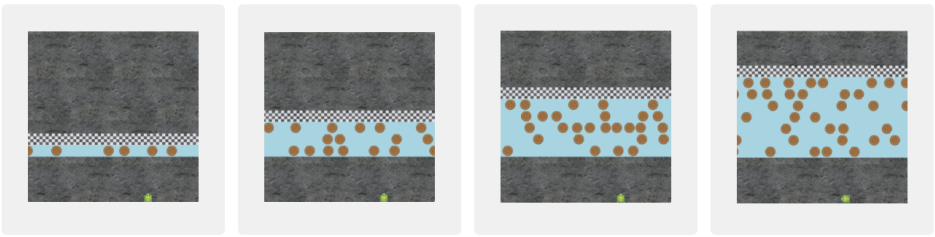
### E.3.1 Procgen Difficulty Levels

We provide the levels that were shown to participants in the different Procgen games (Fig. E.4). There were four levels in varying difficulty, with Level 3 being the goal level.

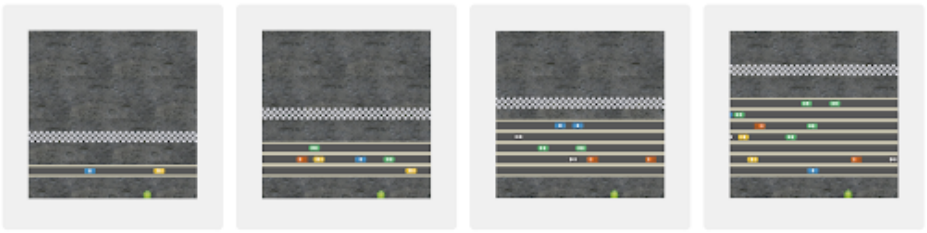
### E.3.2 Experimental Procedure

The study was performed on a computer. Participants were randomly assigned to play one of the four Procgen games. Participants first underwent a familiarization trial where they practiced exploring an empty environment of the game with a video-game controller (e.g., they explored Leaper without any lanes or obstacles between the starting point and the finishing line).

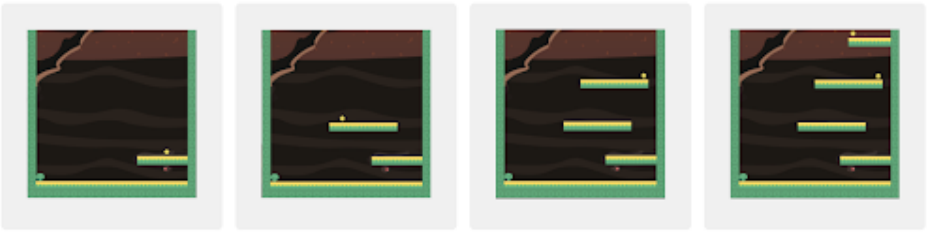
Next, participants were told to play the goal level of the game in which they would be rewarded a sticker if they won. The rules of the game were not revealed to the participants. Since the experiment aimed to measure curriculum learning in a case where the goal was too challenging to be attained outright, participants must fail the goal level to continue with the experiment. If a participant passed the goal level, experimenters reassigned them to play a different Procgen game. After the participant failed the goal level, the experimenter restated to them that the goal



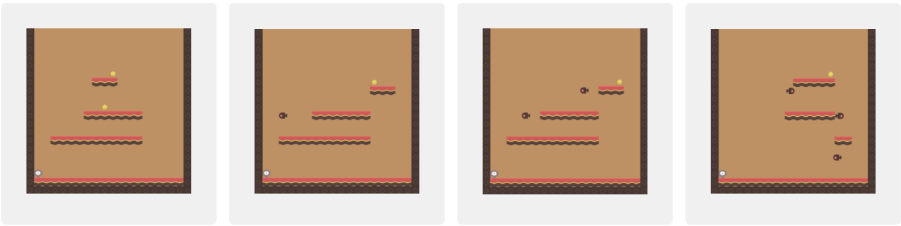
(a) Levels 1-4 for Leaper Log



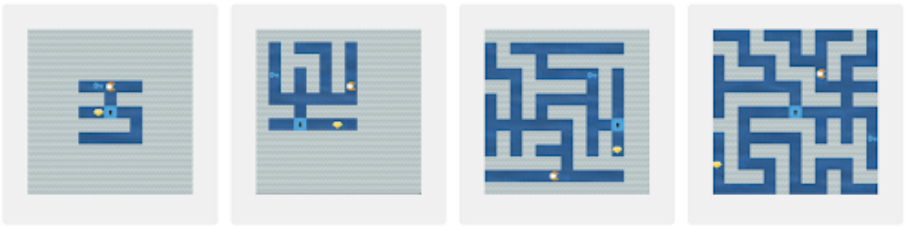
(b) Levels 1-4 for Leaper Car



(c) Levels 1-4 for Climber Platform



(d) Levels 1-4 for Climber Enemy



(e) Levels 1-4 for Heist Size

Figure E.4: Procgen levels that were shown to participants.

was to win that particular level, and once they did, they would get a sticker. The experimenter asked if the participant could tell them how the game worked and what the participant would have to do to get the sticker.

Then, participants were asked which level of the game they wanted to play next and why. They were shown images of varying difficulty levels that quantitatively varied along a single game axis on a tablet, similarly to the ones in Figure E.3. Specifically, they were told the increasing variable of the axis along the levels but were not explicitly told the relative difficulty of each level. Participants were presented with a total of three levels of difficulty: the goal level and two levels that were incrementally easier than the goal level. A subset of 7 participants were further presented with an additional level that was unnecessarily more difficult than the goal level.

Participants received verbal and visual feedback about their performance after playing their selected levels each time. After every other trial, they were reminded to focus on the goal level in order to win a sticker.

This procedure continued until the participant passed the goal sticker level or up to a total of 10 trials, whichever was earlier. Participants who did not pass the goal level by the tenth trial were invited to play the goal level again and then the experiment concluded.

### E.3.3 Results

Figures E.5 and E.6 show the level adjustments children made when selections of a level that is unnecessarily challenging beyond the goal level are not or are included. In the latter case, since there is a total of 4 difficulty levels in this case, the maximum possible absolute level change is 3. Whereas one participant made a level adjustment from Level 4 to Level 1, no participant made a level adjustment from Level 1 to Level 4. Figure E.7 shows the level adjustments made within each level based on children's current level competence.

## E.4 Hand-Designed Curriculum Learning in Reinforcement Learning Agents

### E.4.1 Additional Method Details

Motivated by recent work that suggests tasks should be solved in an easiest-to-hardest fashion for better sample difficulty [272], our hand-designed curriculum function $\phi$ starts at 1 water lane (easiest) and trains the agent until a mean episode reward of 9 (out of possible 10) is achieved. Leaper is a sparse rewards problem,

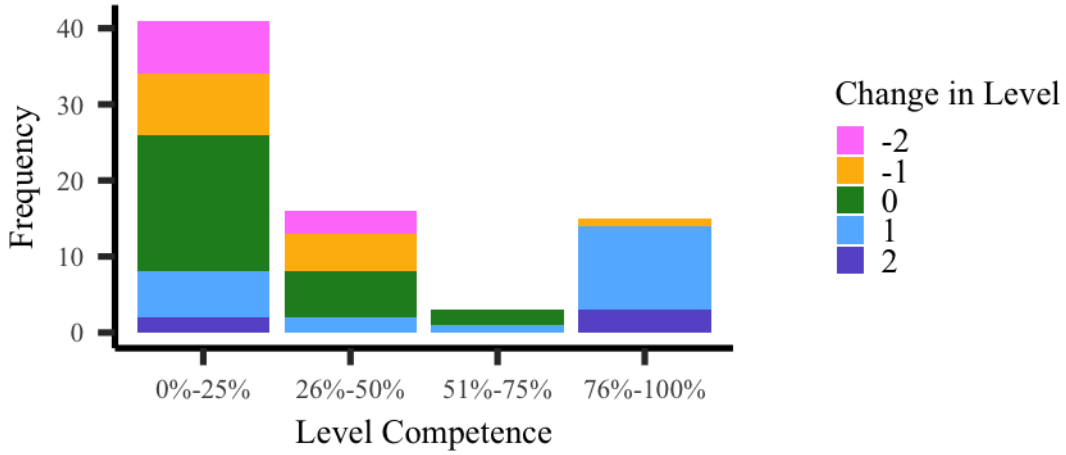



Figure E.5: Level adjustments based on children's level competence on the current level. The x-axis measures current level competence as a percentage; the y-axis shows subsequent level adjustment frequency. A level change of 1 implies choosing a game one level harder, while -1 means opting for one level easier. This figure includes participants' selections of levels easier than or equivalent to the goal level. Overall, children tend to remain on the current level when their level competence is less than 75%. However, upon reaching a 76% completion rate, children often transition to more challenging levels. Conversely, when children demonstrate less than 50% level competence, they are more inclined to return to easier levels. Thus, children adapt their learning trajectory based on their performance.

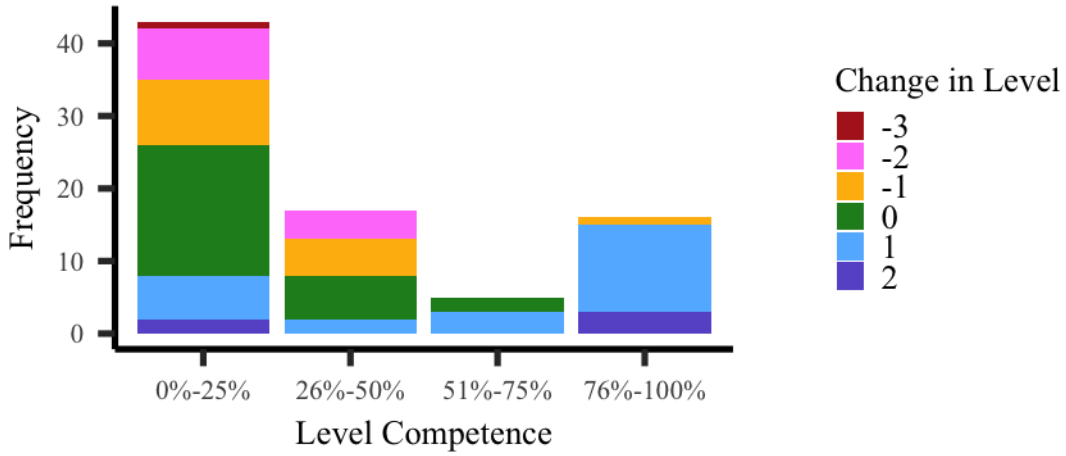
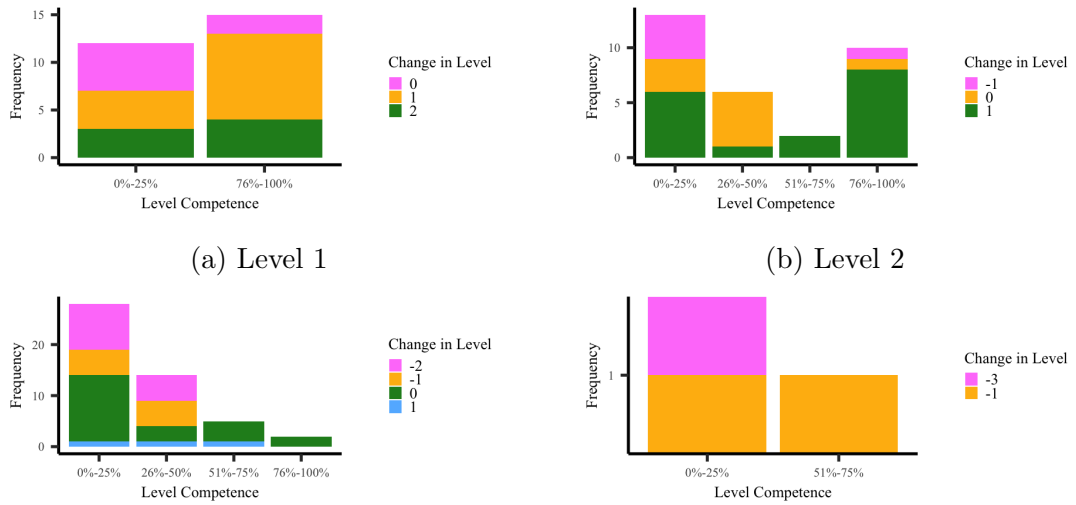



Figure E.6: Level adjustments based on children's percentage of competence on the current level with the inclusion of the extra challenging Level 4.




(a) Level 1 (b) Level 2

(c) Level 3 (Target) (d) Level 4 (Extra Challenging)

Figure E.7: Children made level adjustments based on competence within each level. Note that only 2 participants selected Level 4 at any point of the curriculum.

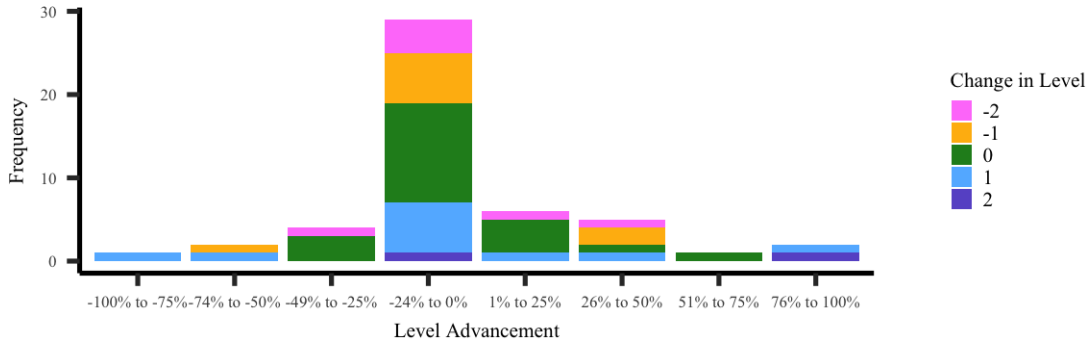



Figure E.8: There was no significant correlation between children's percentage of advancement within their current level and children's level adjustment. Most of the time children were making -24% to 0% level advancement, and yet many of them still opted to remain on the same level or select a more challenging level

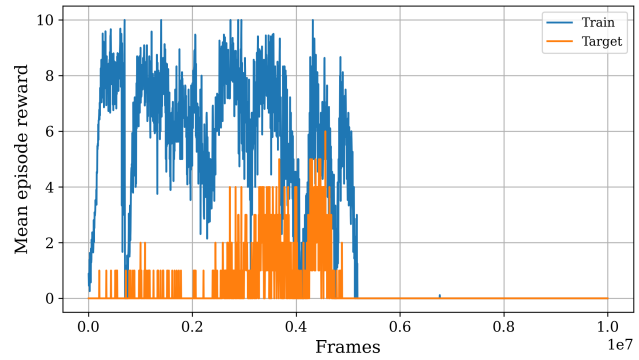
where a reward of 10 is only provided when an episode is solved (and 0 otherwise). Specifically, we run $w = 16$ parallel tasks, initially starting all 16 at 1 water lane. Then, the curriculum function $\phi$ increments $\Theta$ by $1/w$ (e.g., 1/16) to advance to a harder distribution of tasks. After each PPO update, the agent is evaluated against the target task $M_t$. If the agent successfully solves $M_t$, training concludes. Otherwise, training continues until a maximum number of frames $f_m = 10 \times 10^6$.

### E.4.2 Results

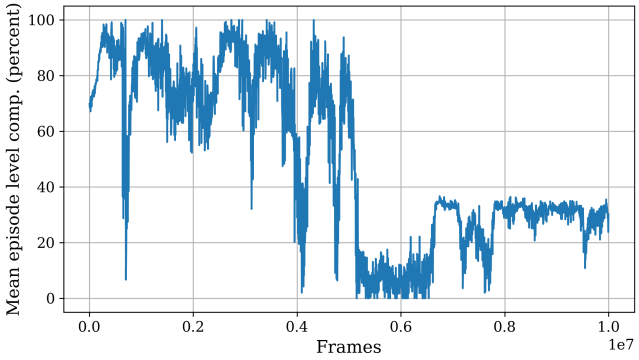
This section contains more analysis of the experiments conducted in Sec. 8.4.3 and Sec. 8.4.4.

### E.4.3 Random Level Baseline

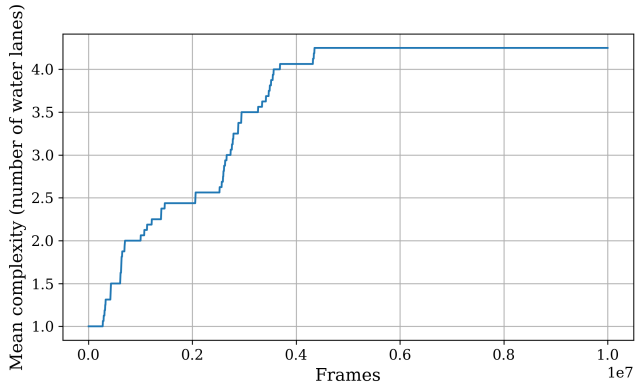
In this experiment, we evaluate a baseline using a random curriculum: where the level is randomly selected from the possible distribution of levels. We conduct this experiment six times. We consistently saw poor learning performance, and the reward obtained on the goal level was generally zero. This baseline is quite challenging as it is difficult to obtain a consistent learning signal from the extrinsic reward alone.




(a)

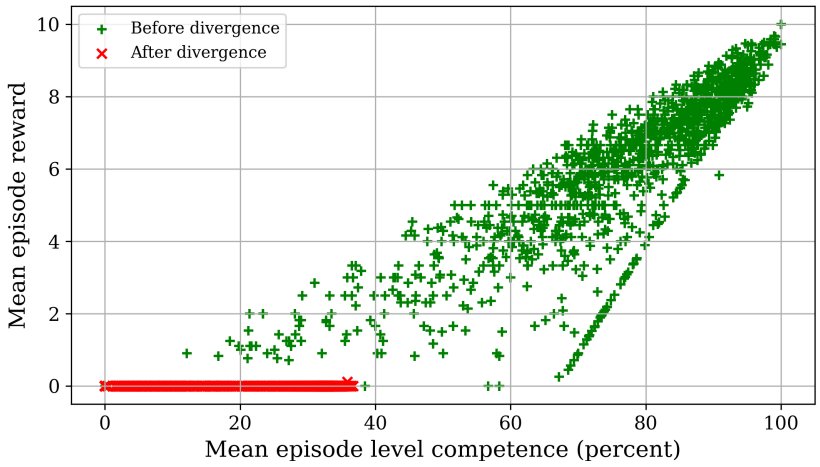



(b)

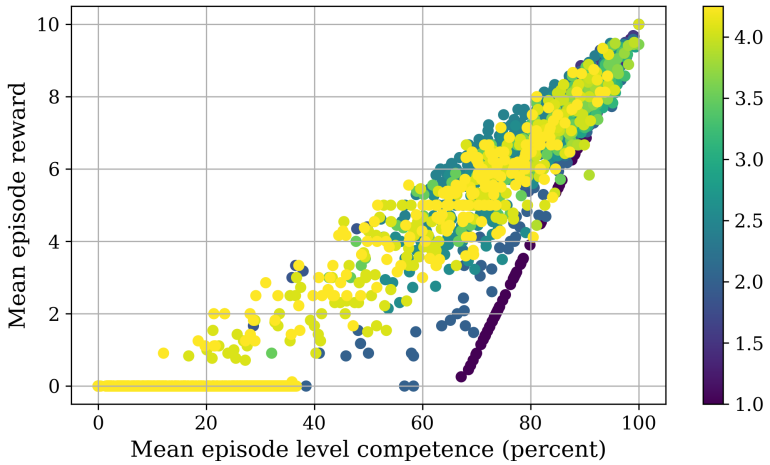



(c)

Figure E.9: Representative results for baseline curriculum learning with an RL agent. (a) Time history of mean episode reward obtained by the agent in both the training levels and the goal level. Training divergence from catastrophic forgetting results in a regression of reward to zero, which occurs around 5.18 million frames. (b) Time history of mean level competence in the training tasks. (c) Time history of the training task difficulty, as measured by number of water lanes.

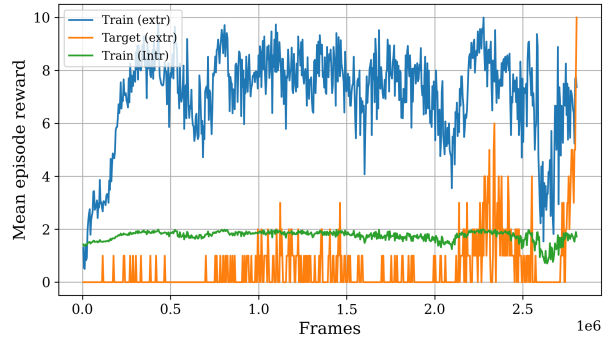



(a) Reward by level competence, colored by divergence.

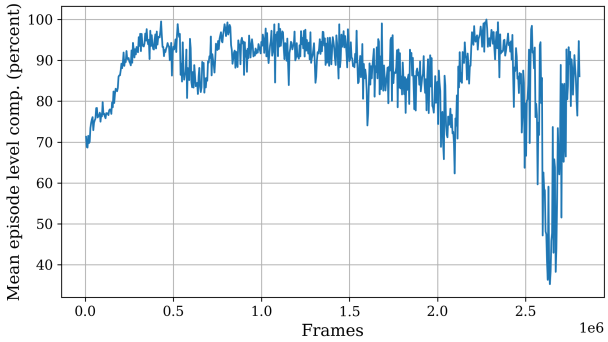



(b) Reward by level competence, colored by difficulty.

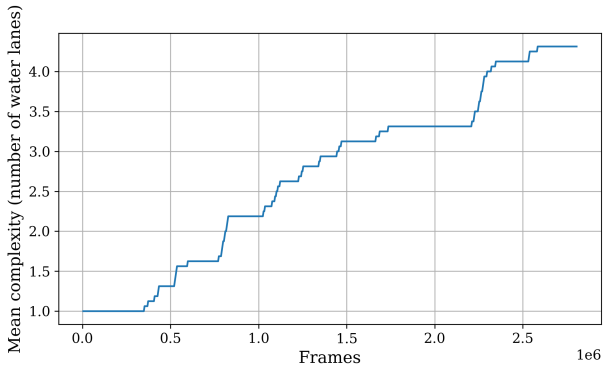
Figure E.10: Level competence is a proxy for reward. (a) Prior to training divergence, mean episode training reward is proportional to mean episode level competence. After training divergence, this relationship no longer holds: the reward remains at zero regardless of level competence. (b) Before training divergence, the exact relationship of reward and level competence depends on the task difficulty. The easiest task (1 water lane, dark blue) has the greatest slope, since changes in level competence yield relatively greater mean training reward. The slope decreases as difficulty increases because tasks have increasingly more vertical lanes before the goal.




(a)




(b)




(c)

Figure E.11: Representative results for curriculum learning with an RL agent while training on level competence as an auxiliary reward. (a) Time history of mean episode reward obtained by the agent in both the training tasks and the target task. The intrisic reward used for training is also shown, which is derived from the agent's level competence. The agent begins to generalize to the target task around 2.8 million frames, eventually leading to solving the target task in 2.806 million frames. (b) Time history of mean level competence in the training tasks. (c) Time history of the training task difficulty, as measured by number of water lanes.

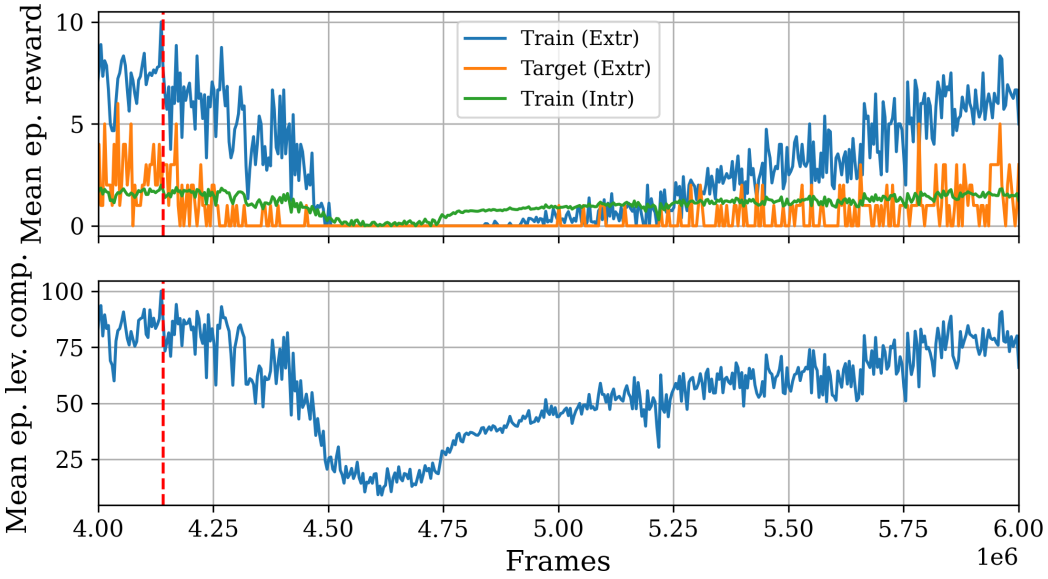



Figure E.12: Training using level competence as an intrinsic reward can recover from catastrophic forgetting that would have otherwise led to training divergence. The vertical red line marks the increase in task difficulty from 4 to 4.0625 water lanes, precipitating (recoverable) catastrophic forgetting.

# Bibliography


[1] R. Brooks, "A Human in the Loop: AI Won't Surpass Human Intelligence Anytime Soon," *IEEE Spectrum*, vol. 58, no. 10, pp. 48–49, 2021.

[2] D. Silver, T. Hubert, J. Schrittwieser, I. Antonoglou, M. Lai, A. Guez, M. Lanctot, L. Sifre, D. Kumaran, T. Graepel, *et al.*, "A General Reinforcement Learning Algorithm that Masters Chess, Shogi, and Go through Self-Play," *Science*, vol. 362, no. 6419, pp. 1140–1144, 2018.

[3] A. W. Senior, R. Evans, J. Jumper, J. Kirkpatrick, L. Sifre, T. Green, C. Qin, A. Žídek, A. W. Nelson, A. Bridgland, *et al.*, "Improved Protein Structure Prediction using Potentials from Deep Learning," *Nature*, vol. 577, no. 7792, pp. 706–710, 2020.

[4] OpenAI, "GPT-4 Technical Report," *arXiv preprint arXiv:2303.08774*, 2023.

[5] W. X. Zhao, K. Zhou, J. Li, T. Tang, X. Wang, Y. Hou, Y. Min, B. Zhang, J. Zhang, Z. Dong, *et al.*, "A Survey of Large Language Models," *arXiv preprint arXiv:2303.18223*, 2023.

[6] M. Anderson, "The Road Ahead for Self-Driving Cars," *IEEE Spectrum*, vol. 57, no. 5, pp. 8–9, 2020.

[7] E. Ackerman, "With New Roomba j7, iRobot Wants to Understand Our Homes," *IEEE Spectrum*, 9 Sep 2021.

[8] L. Sanneman, C. Fourie, and J. A. Shah, "The State of Industrial Robotics: Emerging Technologies, Challenges, and Key Research Directions," *Foundations and Trends in Robotics*, vol. 8, no. 3, pp. 225–306, 2021.

[9] National Institute of Standards and Technology (NIST), "AI for Manufacturing Robotics Initiative." `https://sites.google.com/view/ai4manufacturingrobotics/`, 29 Oct 2021.

[10] I. Akkaya, M. Andrychowicz, M. Chociej, M. Litwin, B. McGrew, A. Petron, A. Paino, M. Plappert, G. Powell, R. Ribas, *et al.*, "Solving Rubik's Cube with a Robot Hand," *arXiv preprint arXiv:1910.07113*, 2019.

[11] J. Bohren, R. B. Rusu, E. G. Jones, E. Marder-Eppstein, C. Pantofaru, M. Wise, L. Mösenlechner, W. Meeussen, and S. Holzer, "Towards Autonomous Robotic Butlers: Lessons Learned with the PR2," *2011 IEEE International Conference on Robotics and Automation (ICRA)*, 2011.

[12] A. Bendale and T. Boult, "Towards Open World Recognition," *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1893–1902, 2015.

[13] S. Kong and D. Ramanan, "OpenGAN: Open-Set Recognition via Open Data Generation," *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 813–822, 2021.

[14] S. Kong, D. Ramanan, T. Boult, A. Owens, C. Vondrick, Y.-X. Wang, and A. Shrivastava, "Open World Vision," *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshop*, 2021.

[15] N. Roy, I. Posner, T. Barfoot, P. Beaudoin, Y. Bengio, J. Bohg, O. Brock, I. Depatie, D. Fox, D. Koditschek, *et al.*, "From Machine Learning to Robotics: Challenges and Opportunities for Embodied Intelligence," *arXiv preprint arXiv:2110.15245*, 2021.

[16] J. Pearl, *Causality: Models, Reasoning, and Inference*. New York, NY: Cambridge University Press, 2nd ed., 2009.

[17] J. Peters, D. Janzing, and B. Schölkopf, *Elements of Causal Inference: Foundations and Learning Algorithms*. Cambridge, MA: MIT Press, 2017.

[18] J. Pearl, "Causal Inference in Statistics: An Overview," *Statistics Surveys*, vol. 3, pp. 96–146, 2009.

[19] P. Spirtes, C. Glymour, and R. Scheines, *Causation, Prediction, and Search*. MIT Press, 2nd ed., 2001.

[20] G. W. Imbens and D. B. Rubin, *Causal Inference in Statistics, Social, and Biomedical Sciences: An Introduction*. Cambridge University Press, 2015.

[21] J. Pearl, M. Glymour, and N. P. Jewell, *Causal Inference in Statistics: A Primer*. John Wiley & Sons, 2016.

[22] J. Pearl and D. Mackenzie, *The Book of Why: The New Science of Cause and Effect*. Basic Books, 2018.

[23] J. Splawa-Neyman, D. M. Dabrowska, and T. P. Speed, "On the Application of Probability Theory to Agricultural Experiments. Essay on Principles. Section 9.," *Statistical Science*, pp. 465–472, 1990.

[24] D. B. Rubin, "Estimating Causal Effects of Treatments in Randomized and Nonrandomized Studies," *Journal of Educational Psychology*, vol. 66, no. 5, p. 688, 1974.

[25] B. Schölkopf, F. Locatello, S. Bauer, N. R. Ke, N. Kalchbrenner, A. Goyal, and Y. Bengio, "Toward Causal Representation Learning," *Proceedings of the IEEE*, vol. 109, no. 5, pp. 612–634, 2021.

[26] S. Bakas, M. Reyes, A. Jakab, S. Bauer, M. Rempfler, A. Crimi, R. T. Shinohara, C. Berger, S. M. Ha, M. Rozycki, *et al.*, "Identifying the Best Machine Learning Algorithms for Brain Tumor Segmentation, Progression Assessment, and Overall Survival Prediction in the BRATS Challenge," *arXiv preprint arXiv:1811.02629*, 2018.

[27] P. Spirtes and K. Zhang, "Causal Discovery and Inference: Concepts and Recent Methodological Advances," *Applied Informatics*, vol. 3, no. 1, pp. 1–28, 2016.

[28] C. Glymour, K. Zhang, and P. Spirtes, "Review of Causal Discovery Methods Based on Graphical Models," *Frontiers in Genetics*, vol. 10, p. 524, 2019.

[29] T. E. Lee, J. Tremblay, T. To, J. Cheng, T. Mosier, O. Kroemer, D. Fox, and S. Birchfield, "Camera-to-Robot Pose Estimation from a Single Image," *2020 IEEE International Conference on Robotics and Automation (ICRA)*, 2020.

[30] V. Zeng, T. E. Lee, J. Liang, and O. Kroemer, "Visual Identification of Articulated Object Parts," *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2021.

[31] T. E. Lee, J. A. Zhao, A. S. Sawhney, S. Girdhar, and O. Kroemer, "Causal Reasoning in Simulation for Structure and Transfer Learning of Robot Manipulation Policies," *2021 IEEE International Conference on Robotics and Automation (ICRA)*, 2021.

[32] T. E. Lee*, S. Vats*, S. Girdhar, and O. Kroemer, "SCALE: Causal Learning and Discovery of Robot Manipulation Skills using Simulation," *Proceedings of The 7th Conference on Robot Learning (CoRL)*, 2023. *Equal contribution.

[33] S. Weichwald, S. W. Mogensen, T. E. Lee, D. Baumann, O. Kroemer, I. Guyon, S. Trimpe, J. Peters, and N. Pfister, "Learning by Doing: Controlling a Dynamical System using Causality, Control, and Reinforcement Learning," *Proceedings of the NeurIPS 2021 Competitions and Demonstrations Track*, vol. 176, pp. 246–258, 06–14 Dec 2022.

[34] A. Dahmani*, E. Yiu*, N. R. Ke, T. E. Lee, O. Kroemer, and A. Gopnik, "Toward Understanding Automated Causal Curriculum Learning in Humans and

Reinforcement Learning Agents," *The 6th International Workshop on Intrinsically Motivated Open-ended Learning (IMOL)*, 2023. *Equal contribution.

[35] A. Dahmani*, E. Yiu*, N. R. Ke, T. E. Lee, O. Kroemer, and A. Gopnik, "Toward Understanding Automated Causal Curriculum Learning in Humans and Reinforcement Learning Agents," *2023 Interactive Causal Learning Conference (ICLC)*, 2023. *Equal contribution.

[36] A. Dahmani*, E. Yiu*, T. E. Lee, N. R. Ke, O. Kroemer, and A. Gopnik, "From Child's Play to AI: Insights into Automated Causal Curriculum Learning," *Intrinsically Motivated Open-ended Learning Workshop, Thirty-seventh Conference on Neural Information Processing Systems (IMOL@NeurIPS)*, 2023. *Equal contribution.

[37] J. Kaddour, A. Lynch, Q. Liu, M. J. Kusner, and R. Silva, "Causal Machine Learning: A Survey and Open Problems," *arXiv preprint arXiv:2206.15475*, 2022.

[38] Z. Wang, X. Xiao, Z. Xu, Y. Zhu, and P. Stone, "Causal Dynamics Learning for Task-Independent State Abstraction," *International Conference on Machine Learning (ICML)*, 2022.

[39] S. Garrido-Jurado, R. Muñoz-Salinas, F. J. Madrid-Cuevas, and M. J. Marín-Jiménez, "Automatic Generation and Detection of Highly Reliable Fiducial Markers under Occlusion," *Pattern Recognition*, vol. 47, no. 6, pp. 2280–2292, 2014.

[40] M. Fiala, "ARTag, a Fiducial Marker System using Digital Techniques," *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, vol. 2, pp. 590–596, 2005.

[41] E. Olson, "AprilTag: A Robust and Flexible Visual Fiducial System," *2011 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 3400–3407, 2011.

[42] I. Fassi and G. Legnani, "Hand to Sensor Calibration: A Geometrical Interpretation of the Matrix Equation $AX{=}XB$," *Journal of Robotic Systems*, vol. 22, no. 9, pp. 497–506, 2005.

[43] J. Tobin, R. Fong, A. Ray, J. Schneider, W. Zaremba, and P. Abbeel, "Domain Randomization for Transferring Deep Neural Networks from Simulation to the Real World," *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 23–30, 2017.

[44] S. Levine, P. Pastor, A. Krizhevsky, J. Ibarz, and D. Quillen, "Learning Hand-Eye Coordination for Robotic Grasping with Deep Learning and Large-Scale Data Collection," *The International Journal of Robotics Research*, vol. 37, no. 4-5, pp. 421–436, 2018.

[45] V. Lepetit, F. Moreno-Noguer, and P. Fua, "EP$n$P: An Accurate O($n$) Solution to the P$n$P Problem," *International Journal of Computer Vision*, vol. 81, no. 2, pp. 155–166, 2009.

[46] T. To, J. Tremblay, D. McKay, Y. Yamaguchi, K. Leung, A. Balanon, J. Cheng, and S. Birchfield, "NDDS: NVIDIA Deep Learning Dataset Synthesizer." `https://github.com/NVIDIA/Dataset_Synthesizer`, 2018.

[47] S. Zakharov, I. Shugurov, and S. Ilic, "DPOD: Dense 6D Pose Object Detector in RGB Images," *arXiv preprint arXiv:1902.11020*, vol. 1, no. 2, 2019.

[48] S. Peng, Y. Liu, Q. Huang, X. Zhou, and H. Bao, "PVNet: Pixel-wise Voting Network for 6DoF Pose Estimation," *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 4561–4570, 2019.

[49] Y. Hu, J. Hugonot, P. Fua, and M. Salzmann, "Segmentation-driven 6D Object Pose Estimation," *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 3385–3394, 2019.

[50] K. Simonyan and A. Zisserman, "Very Deep Convolutional Networks for Large-Scale Image Recognition," *International Conference on Learning Representations (ICLR)*, 2015.

[51] B. Xiao, H. Wu, and Y. Wei, "Simple Baselines for Human Pose Estimation and Tracking," *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 466–481, 2018.

[52] J. Tremblay, T. To, B. Sundaralingam, Y. Xiang, D. Fox, and S. Birchfield, "Deep Object Pose Estimation for Semantic Robotic Grasping of Household Objects," *Proceedings of The 2nd Conference on Robot Learning (CoRL)*, 2018.

[53] B. Tekin, S. N. Sinha, and P. Fua, "Real-Time Seamless Single Shot 6D Object Pose Prediction," *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 292–301, 2018.

[54] A. Buslaev, A. Parinov, E. Khvedchenya, V. I. Iglovikov, and A. A. Kalinin, "Albumentations: Fast and Flexible Image Augmentations," *arXiv preprint arXiv:1809.06839*, 2018.

[55] T.-Y. Lin, M. Maire, S. J. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft COCO: Common Objects in Context," *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 740–755, 2014.

[56] B. Calli, A. Singh, A. Walsman, S. Srinivasa, P. Abbeel, and A. M. Dollar, "The YCB Object and Model Set: Towards Common Benchmarks for Manipulation Research," *2015 International Conference on Advanced Robotics (ICAR)*, pp. 510–517, 2015.

[57] J. Tremblay, A. Prakash, D. Acuna, M. Brophy, V. Jampani, C. Anil, T. To, E. Cameracci, S. Boochoon, and S. Birchfield, "Training Deep Networks with Synthetic Data: Bridging the Reality Gap by Domain Randomization," *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pp. 969–977, 2018.

[58] T. Schmidt, R. A. Newcombe, and D. Fox, "DART: Dense Articulated Real-Time Tracking," *Robotics: Science and Systems (RSS)*, 2014.

[59] N. D. Ratliff, J. Issac, D. Kappler, S. Birchfield, and D. Fox, "Riemannian Motion Policies," *arXiv preprint arXiv:1801.02854*, 2018.

[60] C.-A. Cheng, M. Mukadam, J. Issac, S. Birchfield, D. Fox, B. Boots, and N. Ratliff, "RMPflow: A Computational Graph for Automatic Motion Policy Generation," *International Workshop on the Algorithmic Foundations of Robotics (WAFR)*, pp. 441–457, 2018.

[61] J. Tremblay, T. To, A. Molchanov, S. Tyree, J. Kautz, and S. Birchfield, "Synthetically Trained Neural Networks for Learning Human-Readable Plans from Real-World Demonstrations," *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 5659–5666, 2018.

[62] S. Hinterstoisser, V. Lepetit, S. Ilic, S. Holzer, G. Bradski, K. Konolige, and N. Navab, "Model Based Training, Detection and Pose Estimation of Texture-Less 3D Objects in Heavily Cluttered Scenes," *Asian Conference on Computer Vision (ACCV)*, pp. 548–562, 2012.

[63] Y. Xiang, T. Schmidt, V. Narayanan, and D. Fox, "PoseCNN: A Convolutional Neural Network for 6D Object Pose Estimation in Cluttered Scenes," *Robotics: Science and Systems (RSS)*, 2018.

[64] D. P. Kingma and J. Ba, "Adam: A Method for Stochastic Optimization," *International Conference on Learning Representations (ICLR)*, 2015.

[65] C. Liu and M. Tomizuka, "Robot Safe Interaction System for Intelligent Industrial Co-Robots," *arXiv preprint arXiv:1808.03983*, 2018.

[66] C. H. Kim and J. Seo, "Shallow-Depth Insertion: Peg in Shallow Hole Through Robotic In-Hand Manipulation," *2019 International Conference on Robotics and Automation (ICRA)*, vol. 4, no. 2, pp. 383–390, 2019.

[67] N. Tian, A. K. Tanwani, J. Chen, M. Ma, R. Zhang, B. Huang, K. Goldberg, and S. Sojoudi, "A Fog Robotic System for Dynamic Visual Servoing," *2019 International Conference on Robotics and Automation (ICRA)*, pp. 1982–1988, 2019.

[68] T. Hodan, P. Haluza, Š. Obdržálek, J. Matas, M. Lourakis, and X. Zabulis, "T-LESS: An RGB-D Dataset for 6D Pose Estimation of Texture-less Objects," *2017 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pp. 880–888, 2017.

[69] M. Sundermeyer, Z.-C. Marton, M. Durner, M. Brucker, and R. Triebel, "Implicit 3D Orientation Learning for 6D Object Detection from RGB Images," *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 699–715, 2018.

[70] S.-E. Wei, V. Ramakrishna, T. Kanade, and Y. Sheikh, "Convolutional Pose Machines," *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 4724–4732, 2016.

[71] Z. Cao, T. Simon, S.-E. Wei, and Y. Sheikh, "Realtime Multi-Person 2D Pose Estimation using Part Affinity Fields," *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 7291–7299, 2017.

[72] W. Li, Z. Wang, B. Yin, Q. Peng, Y. Du, T. Xiao, G. Yu, H. Lu, Y. Wei, and J. Sun, "Rethinking on Multi-Stage Networks for Human Pose Estimation," *arXiv preprint arXiv:1901.00148*, 2019.

[73] K. Sun, B. Xiao, D. Liu, and J. Wang, "Deep High-Resolution Representation Learning for Human Pose Estimation," *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 5693–5703, 2019.

[74] D. J. Tan, N. Navab, and F. Tombari, "6D Object Pose Estimation with Depth Images: A Seamless Approach for Robotic Interaction and Augmented Reality," *arXiv preprint arXiv 1709.01459*, 2017.

[75] A. Dhall, D. Dai, and L. Van Gool, "Real-time 3D Traffic Cone Detection for Autonomous Driving," *2019 IEEE Intelligent Vehicles Symposium (IV)*, pp. 494–501, 2019.

[76] G. Pavlakos, X. Zhou, A. Chan, K. G. Derpanis, and K. Daniilidis, "6-DoF Object Pose from Semantic Keypoints," *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 2011–2018, 2017.

[77] F. C. Park and B. J. Martin, "Robot Sensor Calibration: Solving $AX{=}XB$ on the Euclidean Group," *IEEE Transactions on Robotics and Automation*, vol. 10, no. 5, pp. 717–721, 1994.

[78] J. Ilonen and V. Kyrki, "Robust Robot-Camera Calibration," *2011 15th International Conference on Advanced Robotics (ICAR)*, pp. 67–74, 2011.

[79] D. Yang and J. Illingworth, "Calibrating a Robot Camera," *BMVC*, pp. 1–10, 1994.

[80] K. Pauwels and D. Kragic, "Integrated On-line Robot-camera Calibration and Object Pose Estimation," *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 2332–2339, 2016.

[81] D. Park, Y. Seo, and S. Y. Chun, "Real-Time, Highly Accurate Robotic Grasp Detection using Fully Convolutional Neural Networks with High-Resolution Images," *arXiv preprint arXiv:1809.05828*, 2018.

[82] A. Aalerud, J. Dybedal, and G. Hovland, "Automatic Calibration of an Industrial RGB-D Camera Network Using Retroreflective Fiducial Markers," *Sensors*, vol. 19, no. 7, p. 1561, 2019.

[83] D. Morrison, P. Corke, and J. Leitner, "Closing the Loop for Robotic Grasping: A Real-time, Generative Grasp Synthesis Approach," *Robotics: Science and Systems (RSS)*, 2018.

[84] A. Feniello, H. Dang, and S. Birchfield, "Program Synthesis by Examples for Object Repositioning Tasks," *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 4428–4435, 2014.

[85] J. Mahler, J. Liang, S. Niyaz, M. Laskey, R. Doan, X. Liu, J. A. Ojea, and K. Goldberg, "Dex-Net 2.0: Deep Learning to Plan Robust Grasps with Synthetic Point Clouds and Analytic Grasp Metrics," *Robotics: Science and Systems (RSS)*, 2017.

[86] J. Bohg, J. Romero, A. Herzog, and S. Schaal, "Robot Arm Pose Estimation through Pixel-Wise Part Classification," *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 3143–3150, 2014.

[87] F. Widmaier, D. Kappler, S. Schaal, and J. Bohg, "Robot Arm Pose Estimation by Pixel-Wise Regression of Joint Angles," *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 616–623, 2016.

[88] J. Lambrecht and L. Kästner, "Towards the Usage of Synthetic Data for Marker-Less Pose Estimation of Articulated Robots in RGB Images," *2019 19th International Conference on Advanced Robotics (ICAR)*, pp. 240–247, 2019.

[89] J. Lambrecht, "Robust Few-Shot Pose Estimation of Articulated Robots using Monocular Cameras and Deep-Learning-based Keypoint Detection," *2019 7th International Conference on Robot Intelligence Technology and Applications (RiTA)*, pp. 136–141, 2019.

[90] Y. Zuo, W. Qiu, L. Xie, F. Zhong, Y. Wang, and A. L. Yuille, "CRAVES: Controlling Robotic Arm With a Vision-Based Economic System," *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 4214–4223, 2019.

[91] G. Bekey and J. Yuh, "The Status of Robotics," *IEEE Robotics & Automation Magazine*, vol. 15, no. 1, pp. 80–86, 2008.

[92] K. Doelling, J. Shin, and D. O. Popa, "Service Robotics for the Home: A State of the Art Review," in *Proceedings of the 7th International Conference on PErvasive Technologies Related to Assistive Environments (PETRA)*, pp. 1–8, 2014.

[93] P. Dario, E. Guglielmelli, B. Allotta, and M. C. Carrozza, "Robotics for Medical Applications," *IEEE Robotics & Automation Magazine*, vol. 3, no. 3, pp. 44–56, 1996.

[94] L. D. Riek, "Healthcare Robotics," *Communications of the ACM*, vol. 60, no. 11, pp. 68–78, 2017.

[95] G.-Z. Yang, B. J. Nelson, R. R. Murphy, H. Choset, H. Christensen, S. H. Collins, P. Dario, K. Goldberg, K. Ikuta, N. Jacobstein, *et al.*, "Combating COVID-19 – The Role of Robotics in Managing Public Health and Infectious Diseases," *Science Robotics*, vol. 5, no. 40, p. eabb5589, 2020.

[96] J. Bohg, K. Hausman, B. Sankaran, O. Brock, D. Kragic, S. Schaal, and G. S. Sukhatme, "Interactive Perception: Leveraging Action in Perception and Perception in Action," *IEEE Transactions on Robotics*, vol. 33, no. 6, pp. 1273–1291, 2017.

[97] J. Sturm, A. Jain, C. Stachniss, C. C. Kemp, and W. Burgard, "Operating Articulated Objects based on Experience," in *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 2739–2744, IEEE, 2010.

[98] J. Sturm, C. Stachniss, and W. Burgard, "A Probabilistic Framework for Learning Kinematic Models of Articulated Objects," *Journal of Artificial Intelligence Research*, vol. 41, pp. 477–526, 2011.

[99] S. Höfer, T. Lang, and O. Brock, "Extracting Kinematic Background Knowledge from Interactions using Task-Sensitive Relational Learning," *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 4342–4347, 2014.

[100] P. R. Barragän, L. P. Kaelbling, and T. Lozano-Pérez, "Interactive Bayesian Identification of Kinematic Mechanisms," *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 2013–2020, 2014.

[101] C. Moses, M. Noseworthy, L. P. Kaelbling, T. Lozano-Pérez, and N. Roy, "Visual Prediction of Priors for Articulated Object Interaction," *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 10480–10486, 2020.

[102] R. M. Martin and O. Brock, "Online Interactive Perception of Articulated Objects with Multi-Level Recursive Estimation Based on Task-Specific Priors," *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 2494–2501, 2014.

[103] D. Katz, A. Orthey, and O. Brock, "Interactive Perception of Articulated Objects," *Experimental Robotics*, pp. 301–315, 2014.

[104] K. Hausman, S. Niekum, S. Osentoski, and G. S. Sukhatme, "Active Articulation Model Estimation through Interactive Perception," *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 3305–3312, 2015.

[105] R. Martín-Martín, S. Höfer, and O. Brock, "An Integrated Approach to Visual Perception of Articulated Objects," *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 5091–5097, 2016.

[106] R. Martín-Martín and O. Brock, "Building Kinematic and Dynamic Models of Articulated Objects with Multi-Modal Interactive Perception," *AAAI Symposium on Interactive Multi-Sensory Object Perception for Embodied Agents*, 2017.

[107] M. Baum, M. Bernstein, R. Martin-Martin, S. Höfer, J. Kulick, M. Toussaint, A. Kacelnik, and O. Brock, "Opening a Lockbox through Physical Exploration," *2017 IEEE-RAS 17th International Conference on Humanoid Robotics (Humanoids)*, pp. 461–467, 2017.

[108] C. Eppner, R. Martín-Martín, and O. Brock, "Physics-Based Selection of Informative Actions for Interactive Perception," *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 7427–7432, 2018.

[109] F. Xiang, Y. Qin, K. Mo, Y. Xia, H. Zhu, F. Liu, M. Liu, H. Jiang, Y. Yuan, H. Wang, *et al.*, "SAPIEN: A SimulAted Part-based Interactive ENvironment," *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 11097–11107, 2020.

[110] X. Wang, B. Zhou, Y. Shi, X. Chen, Q. Zhao, and K. Xu, "Shape2Motion: Joint Analysis of Motion Parts and Attributes from 3D Shapes," *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 8876–8884, 2019.

[111] B. Abbatematteo, S. Tellex, and G. Konidaris, "Learning to Generalize Kinematic Models to Novel Objects," *Proceedings of the Conference on Robot Learning (CoRL)*, 2019.

[112] X. Li, H. Wang, L. Yi, L. J. Guibas, A. L. Abbott, and S. Song, "Category-Level Articulated Object Pose Estimation," *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 3706–3715, 2020.

[113] A. Jain, R. Lioutikov, C. Chuck, and S. Niekum, "ScrewNet: Category-Independent Articulation Model Estimation From Depth Images Using Screw Theory," *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 13670–13677, 2021.

[114] X. Huang, I. Walker, and S. Birchfield, "Occlusion-Aware Reconstruction and Manipulation of 3D Articulated Objects," *2012 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1365–1371, 2012.

[115] X. Huang, I. Walker, and S. Birchfield, "Occlusion-Aware Multi-View Reconstruction of Articulated Objects for Manipulation," *Robotics and Autonomous Systems*, vol. 62, no. 4, pp. 497–505, 2014.

[116] P. R. Florence, L. Manuelli, and R. Tedrake, "Dense Object Nets: Learning Dense Visual Object Descriptors By and For Robotic Manipulation," *Proceedings of The 2nd Conference on Robot Learning (CoRL)*, 2018.

[117] J. Lu, F. Richter, and M. Yip, "Robust Keypoint Detection and Pose Estimation of Robot Manipulators with Self-Occlusions via Sim-to-Real Transfer," *arXiv preprint arXiv:2010.08054*, 2020.

[118] J. Tremblay, S. Tyree, T. Mosier, and S. Birchfield, "Indirect Object-to-Robot Pose Estimation from an External Monocular RGB Camera," *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 4227–4234, 2020.

[119] Z. Wang and F. Lu, "VoxSegNet: Volumetric CNNs for Semantic Part Segmentation of 3D Shapes," *IEEE Transactions on Visualization and Computer Graphics*, vol. 26, no. 9, pp. 2919–2930, 2019.

[120] R. Martín-Martín, C. Eppner, and O. Brock, "The RBO Dataset of Articulated Objects and Interactions," *The International Journal of Robotics Research*, vol. 38, no. 9, pp. 1013–1019, 2019.

[121] A. X. Chang, T. Funkhouser, L. Guibas, P. Hanrahan, Q. Huang, Z. Li, S. Savarese, M. Savva, S. Song, H. Su, *et al.*, "ShapeNet: An Information-Rich 3D Model Repository," *arXiv preprint arXiv:1512.03012*, 2015.

[122] K. Mo, S. Zhu, A. X. Chang, L. Yi, S. Tripathi, L. J. Guibas, and H. Su, "PartNet: A Large-scale Benchmark for Fine-grained and Hierarchical Part-level 3D Object Understanding," *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 909–918, 2019.

[123] K. Simonyan and A. Zisserman, "Very Deep Convolutional Networks for Large-Scale Image Recognition," *International Conference on Learning Representations (ICLR)*, 2014.

[124] M. A. Fischler and R. C. Bolles, "Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography," *Communications of the ACM*, vol. 24, no. 6, pp. 381–395, 1981.

[125] K.-K. Maninis, S. Caelles, J. Pont-Tuset, and L. Van Gool, "Deep Extreme Cut: From Extreme Points to Object Segmentation," *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 616–625, 2018.

[126] X. B. Peng, M. Andrychowicz, W. Zaremba, and P. Abbeel, "Sim-to-Real Transfer of Robotic Control with Dynamics Randomization," *2018 IEEE International Conference on Robotics and Automation (ICRA)*, 2018.

[127] K. Bousmalis, A. Irpan, P. Wohlhart, Y. Bai, M. Kelcey, M. Kalakrishnan, L. Downs, J. Ibarz, P. Pastor, K. Konolige, *et al.*, "Using Simulation and Domain Adaptation to Improve Efficiency of Deep Robotic Grasping," *2018 IEEE International Conference on Robotics and Automation (ICRA)*, 2018.

[128] O. Kroemer, S. Niekum, and G. Konidaris, "A Review of Robot Learning for Manipulation: Challenges, Representations, and Algorithms," *arXiv preprint arXiv:1907.03146*, 2019.

[129] W. Zhao, J. P. Queralta, and T. Westerlund, "Sim-to-Real Transfer in Deep Reinforcement Learning for Robotics: a Survey," *IEEE Symposium Series on Computational Intelligence (SSCI)*, 2020.

[130] J. Liang, V. Makoviychuk, A. Handa, N. Chentanez, M. Macklin, and D. Fox, "GPU-Accelerated Robotic Simulation for Distributed Reinforcement Learning," *Proceedings of The 2nd Conference on Robot Learning (CoRL)*, 2018.

[131] K. Zhang, B. Schölkopf, P. Spirtes, and C. Glymour, "Learning Causality and Causality-Related Learning: Some Recent Progress," *National Science Review*, vol. 5, no. 1, pp. 26–29, 2018.

[132] B. Schölkopf, "Causality for Machine Learning," *arXiv preprint arXiv:1911.10500*, 2019.

[133] J. Zhang and E. Bareinboim, "Transfer Learning in Multi-Armed Bandits: A Causal Approach," *International Joint Conference on Artificial Intelligence (IJCAI)*, 2017.

[134] P. de Haan, D. Jayaraman, and S. Levine, "Causal Confusion in Imitation Learning," *Advances in Neural Information Processing Systems 32 (NeurIPS)*, vol. 32, 2019.

[135] Y. Li, A. Torralba, A. Anandkumar, D. Fox, and A. Garg, "Causal Discovery in Physical Systems from Videos," *Advances in Neural Information Processing Systems 33 (NeurIPS)*, vol. 33, pp. 9180–9192, 2020.

[136] S. A. Sontakke, A. Mehrjou, L. Itti, and B. Schölkopf, "Causal Curiosity: RL Agents Discovering Self-supervised Experiments for Causal Representation Learning," *arXiv preprint arXiv:2010.03110*, 2020.

[137] O. Ahmed, F. Träuble, A. Goyal, A. Neitz, M. Wüthrich, Y. Bengio, B. Schölkopf, and S. Bauer, "CausalWorld: A Robotic Manipulation Benchmark for Causal Structure and Transfer Learning," *arXiv preprint arXiv:2010.04296*, 2020.

[138] C. Devin, P. Abbeel, T. Darrell, and S. Levine, "Deep Object-Centric Representations for Generalizable Robot Learning," *2018 IEEE International Conference on Robotics and Automation (ICRA)*, 2018.

[139] W. Yu, J. Tan, C. K. Liu, and G. Turk, "Preparing for the Unknown: Learning a Universal Policy with Online System Identification," *Robotics: Science and Systems (RSS)*, 2017.

[140] A. Nouri and M. L. Littman, "Dimension Reduction and its Application to Model-Based Exploration in Continuous Spaces," *Machine Learning*, vol. 81, no. 1, pp. 85–98, 2010.

[141] J. Z. Kolter and A. Y. Ng, "Regularization and Feature Selection in Least-Squares Temporal Difference Learning," *International Conference on Machine Learning (ICML)*, 2009.

[142] R. Parr, L. Li, G. Taylor, C. Painter-Wakefield, and M. L. Littman, "An Analysis of Linear Models, Linear Value-Function Approximation, and Feature Selection for Reinforcement Learning," *International Conference on Machine Learning (ICML)*, 2008.

[143] J. Tan, T. Zhang, E. Coumans, A. Iscen, Y. Bai, D. Hafner, S. Bohez, and V. Vanhoucke, "Sim-to-Real: Learning Agile Locomotion for Quadruped Robots," *Robotics: Science and Systems (RSS)*, 2018.

[144] A. Molchanov, T. Chen, W. Hönig, J. A. Preiss, N. Ayanian, and G. S. Sukhatme, "Sim-to-(Multi)-Real: Transfer of Low-Level Robust Control Policies to Multiple Quadrotors," *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2019.

[145] Y. Chebotar, A. Handa, V. Makoviychuk, M. Macklin, J. Issac, N. Ratliff, and D. Fox, "Closing the Sim-to-Real Loop: Adapting Simulation Randomization with Real World Experience," *2019 International Conference on Robotics and Automation (ICRA)*, 2019.

[146] W. Masson, P. Ranchod, and G. Konidaris, "Reinforcement Learning with Parameterized Actions," *Proceedings of the AAAI Conference on Artificial Intelligence*, 2016.

[147] M. Hausknecht and P. Stone, "Deep Reinforcement Learning in Parameterized Action Space," *International Conference on Learning Representations (ICLR)*, 2016.

[148] Z. Fan, R. Su, W. Zhang, and Y. Yu, "Hybrid Actor-Critic Reinforcement Learning in Parameterized Action Space," *International Joint Conference on Artificial Intelligence (IJCAI)*, 2019.

[149] M. P. Deisenroth, G. Neumann, J. Peters, *et al.*, "A Survey on Policy Search for Robotics," *Foundations and Trends in Robotics*, vol. 2, no. 1–2, pp. 1–142, 2013.

[150] P. W. Battaglia, J. B. Hamrick, and J. B. Tenenbaum, "Simulation as an Engine of Physical Scene Understanding," *Proceedings of the National Academy of Sciences*, vol. 110, no. 45, pp. 18327–18332, 2013.

[151] D. Ha and J. Schmidhuber, "Recurrent World Models Facilitate Policy Evolution," *Advances in Neural Information Processing Systems 31 (NeurIPS)*, vol. 31, 2018.

[152] J. Peters, K. Mülling, and Y. Altun, "Relative Entropy Policy Search," *Proceedings of the AAAI Conference on Artificial Intelligence*, 2010.

[153] V. R. Konda and J. N. Tsitsiklis, "Actor-Critic Algorithms," *Advances in Neural Information Processing Systems 12 (NIPS)*, vol. 12, 2000.

[154] Z. Lu, H. Pu, F. Wang, Z. Hu, and L. Wang, "The Expressive Power of Neural Networks: A View from the Width," *Advances in Neural Information Processing Systems 30 (NIPS)*, vol. 30, 2017.

[155] A. M. Saxe, J. L. Mcclelland, and S. Ganguli, "Exact Solutions to the Nonlinear Dynamics of Learning in Deep Linear Neural Networks," *International Conference on Learning Representations (ICLR)*, 2014.

[156] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal Policy Optimization Algorithms," *arXiv preprint arXiv:1707.06347*, 2017.

[157] A. Hill, A. Raffin, M. Ernestus, A. Gleave, A. Kanervisto, R. Traore, P. Dhariwal, C. Hesse, O. Klimov, A. Nichol, M. Plappert, A. Radford, J. Schulman, S. Sidor, and Y. Wu, "Stable Baselines." `https://github.com/hill-a/stable-baselines`, 2018.

[158] A. A. Rusu, N. C. Rabinowitz, G. Desjardins, H. Soyer, J. Kirkpatrick, K. Kavukcuoglu, R. Pascanu, and R. Hadsell, "Progressive Neural Networks," *arXiv preprint arXiv:1606.04671*, 2016.

[159] R. Desimone and J. Duncan, "Neural Mechanisms of Selective Visual Attention," *Annual Review of Neuroscience*, vol. 18, no. 1, pp. 193–222, 1995.

[160] G. Konidaris, S. Kuindersma, R. Grupen, and A. Barto, "Robot Learning from Demonstration by Constructing Skill Trees," *The International Journal of Robotics Research*, vol. 31, no. 3, pp. 360–375, 2012.

[161] Z. Su, O. Kroemer, G. E. Loeb, G. S. Sukhatme, and S. Schaal, "Learning Manipulation Graphs from Demonstrations Using Multimodal Sensory Signals," *2018 IEEE International Conference on Robotics and Automation (ICRA)*, 2018.

[162] O. Kroemer, C. Daniel, G. Neumann, H. Van Hoof, and J. Peters, "Towards Learning Hierarchical Skills for Multi-Phase Manipulation Tasks," *2015 IEEE International Conference on Robotics and Automation (ICRA)*, 2015.

[163] A. Chenu, N. Perrin-Gilbert, and O. Sigaud, "Divide & Conquer Imitation Learning," *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2022.

[164] J. Achterhold, M. Krimmel, and J. Stueckler, "Learning Temporally Extended Skills in Continuous Domains as Symbolic Actions for Planning," *arXiv preprint arXiv:2207.05018*, 2022.

[165] N. Jiang, A. Kulesza, and S. Singh, "Abstraction Selection in Model-Based Reinforcement Learning," *International Conference on Machine Learning (ICML)*, 2015.

[166] G. Konidaris and A. Barto, "Efficient Skill Learning using Abstraction Selection," *International Joint Conference on Artificial Intelligence (IJCAI)*, 2009.

[167] B. C. Da Silva, G. Konidaris, and A. G. Barto, "Learning Parameterized Skills," *International Conference on Machine Learning (ICML)*, p. 1443–1450, 2012.

[168] B. C. Da Silva, G. Baldassarre, G. Konidaris, and A. Barto, "Learning Parameterized Motor Skills on a Humanoid Robot," *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 5239–5244, 2014.

[169] J. B. Tenenbaum, V. De Silva, and J. C. Langford, "A Global Geometric Framework for Nonlinear Dimensionality Reduction," *Science*, vol. 290, no. 5500, pp. 2319–2323, 2000.

[170] L. P. Kaelbling and T. Lozano-Pérez, "Learning Composable Models of Parameterized Skills," *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 886–893, 2017.

[171] Z. Wang, C. R. Garrett, L. P. Kaelbling, and T. Lozano-Pérez, "Learning Compositional Models of Robot Skills for Task and Motion Planning," *The International Journal of Robotics Research*, vol. 40, no. 6-7, pp. 866–894, 2021.

[172] J. Peters, J. Kober, K. Mülling, O. Krämer, and G. Neumann, "Towards Robot Skill Learning: From Simple Skills to Table Tennis," *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pp. 627–631, 2013.

[173] R. Pahič, Z. Lončarević, A. Gams, and A. Ude, "Robot Skill Learning in Latent Space of a Deep Autoencoder Neural Network," *Robotics and Autonomous Systems*, vol. 135, p. 103690, 2021.

[174] K. Khetarpal, M. Klissarov, M. Chevalier-Boisvert, P.-L. Bacon, and D. Precup, "Options of Interest: Temporal Abstraction with Interest Functions," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, no. 4, pp. 4444–4451, 2020.

[175] M. Abdulhai, D.-K. Kim, M. Riemer, M. Liu, G. Tesauro, and J. P. How, "Context-Specific Representation Abstraction for Deep Option Learning," *arXiv preprint arXiv:2109.09876*, 2021.

[176] A. Bagaria and G. Konidaris, "Option Discovery using Deep Skill Chaining," *International Conference on Learning Representations (ICLR)*, 2020.

[177] A. Bagaria, J. K. Senthil, and G. Konidaris, "Skill Discovery for Exploration and Planning using Deep Skill Graphs," *International Conference on Machine Learning (ICML)*, 2021.

[178] B. Eysenbach, A. Gupta, J. Ibarz, and S. Levine, "Diversity is All You Need: Learning Skills without a Reward Function," *International Conference on Learning Representations (ICLR)*, 2018.

[179] A. Sharma, S. S. Gu, S. Levine, V. Kumar, and K. Hausman, "Dynamics-Aware Unsupervised Discovery of Skills," *International Conference on Learning Representations (ICLR)*, 2020.

[180] K. C. Stocking, A. Gopnik, and C. Tomlin, "From Robot Learning to Robot Understanding: Leveraging Causal Graphical Models for Robotics," *Proceedings of the 5th Conference on Robot Learning (CoRL)*, 2022.

[181] M. Diehl and K. Ramirez-Amaro, "Why Did I Fail? A Causal-Based Method to Find Explanations for Robot Failures," *IEEE Robotics and Automation Letters*, vol. 7, no. 4, pp. 8925–8932, 2022.

[182] N. R. Ke, A. Didolkar, S. Mittal, A. Goyal, G. Lajoie, S. Bauer, D. Rezende, Y. Bengio, M. Mozer, and C. Pal, "Systematic Evaluation of Causal Discovery in Visual Model Based Reinforcement Learning," *Proceedings of the NeurIPS 2021 Datasets and Benchmarks Track*, 2021. arXiv preprint arXiv:2107.00848.

[183] S. A. Sontakke, A. Mehrjou, L. Itti, and B. Schölkopf, "Causal Curiosity: RL Agents Discovering Self-supervised Experiments for Causal Representation Learning," *International Conference on Machine Learning (ICML)*, 2021.

[184] A. Sonar, V. Pacelli, and A. Majumdar, "Invariant Policy Optimization: Towards Stronger Generalization in Reinforcement Learning," *Learning for Dynamics and Control*, 2021.

[185] O. Kroemer, S. Niekum, and G. D. Konidaris, "A Review of Robot Learning for Manipulation: Challenges, Representations, and Algorithms," *Journal of Machine Learning Research*, vol. 22, no. 30, 2021.

[186] R. S. Sutton, D. Precup, and S. Singh, "Between MDPs and Semi-MDPs: A Framework for Temporal Abstraction in Reinforcement Learning," *Artificial Intelligence*, vol. 112, no. 1-2, pp. 181–211, 1999.

[187] G. Konidaris, L. Kaelbling, and T. Lozano-Perez, "Symbol Acquisition for Probabilistic High-Level Planning," *International Joint Conference on Artificial Intelligence (IJCAI)*, 2015.

[188] J. Luo, E. Solowjow, C. Wen, J. A. Ojea, A. M. Agogino, A. Tamar, and P. Abbeel, "Reinforcement Learning on Variable Impedance Controller for High-Precision Robotic Assembly," *2019 International Conference on Robotics and Automation (ICRA)*, pp. 3080–3087, 2019.

[189] V. Makoviychuk, L. Wawrzyniak, Y. Guo, M. Lu, K. Storey, M. Macklin, D. Hoeller, N. Rudin, A. Allshire, A. Handa, *et al.*, "Isaac Gym: High Performance GPU-Based Physics Simulation For Robot Learning," *arXiv preprint arXiv:2108.10470*, 2021.

[190] R. Brost, "Automatic Grasp Planning in the Presence of Uncertainty," *1986 IEEE International Conference on Robotics and Automation (ICRA)*, vol. 3, pp. 1575–1581, 1986.

[191] T. Haavelmo, "The Probability Approach in Econometrics," *Econometrica*, vol. 12, pp. S1–S115 (supplement), 1944.

[192] J. Aldrich, "Autonomy," *Oxford Economic Papers*, vol. 41, pp. 15–34, 1989.

[193] J. Peters, S. Bauer, and N. Pfister, "Causal Models for Dynamical Systems," in *Probabilistic and Causal Inference: The Works of Judea Pearl (to appear); ArXiv e-prints (2001.06208)*, ACM, 2022.

[194] L. Ljung, *System Identification: Theory for the User*. Pearson Education, 1998.

[195] K. J. Åström and R. M. Murray, *Feedback Systems: An Introduction for Scientists and Engineers*. Princeton, NJ: Princeton University Press, 2nd ed., 2008.

[196] F. Allgöwer and A. Zheng, *Nonlinear Model Predictive Control*. Basel, Switzerland: Birkhäuser, 2012.

[197] F. Borrelli, A. Bemporad, and M. Morari, *Predictive Control for Linear and Hybrid Systems*. New York, NY: Cambridge University Press, 2017.

[198] K. J. Åström and B. Wittenmark, *Adaptive Control*. Courier Corporation, 2013.

[199] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. Cambridge, MA: MIT Press, 1998.

[200] D. Bell, J. Kay, and J. Malley, "A Non-Parametric Approach to Non-Linear Causality Testing," *Economics Letters*, vol. 51, no. 1, pp. 7–18, 1996.

[201] P. Waage and C. M. Guldberg, "Studier over Affiniteten (in Danish)," *Forhandlinger i Videnskabs-selskabet i Christiania*, pp. 35–45, 1864.

[202] D. P. Bertsekas, *Dynamic Programming and Optimal Control: Volume 1*. Belmont, MA: Athena Scientific, 2000.

[203] B. D. O. Anderson and J. B. Moore, *Optimal Control: Linear Quadratic Methods*. Courier Corporation, 2007.

[204] S. Lange, T. Gabel, and M. Riedmiller, "Batch Reinforcement Learning," in *Reinforcement Learning: State-of-the-Art* (M. Wiering and M. van Otterlo, eds.), (Berlin, Heidelberg), pp. 45–73, Springer Berlin Heidelberg, 2012.

[205] S. Levine, A. Kumar, G. Tucker, and J. Fu, "Offline Reinforcement Learning: Tutorial, Review, and Perspectives on Open Problems," *ArXiv e-prints (2005.01643)*, 2020.

[206] J. Bravo, "Learning By Doing NeurIPS 2021 Competition Solution Code." `https://doi.org/10.5281/zenodo.5895099`, Jan. 2022.

[207] B. Bussmann, "A Neural Network Approach to Controlling Chemical Reactions." `https://doi.org/10.5281/zenodo.6006496`, Feb. 2022.

[208] C. M. Patiño, E. Lopez, and J. Rodriguez, "factoredai/learn-by-doing-neurips-2021: v1.0.0." `https://doi.org/10.5281/zenodo.5888574`, Jan. 2022.

[209] A. Lei, B. Schölkopf, and I. Posner, "Variational Causal Dynamics: Discovering Modular World Models from Interventions," *arXiv preprint arXiv:2206.11131*, 2022.

[210] Y. Liu, B. Huang, Z. Zhu, H. Tian, M. Gong, Y. Yu, and K. Zhang, "Learning World Models with Identifiable Factorization," *Advances in Neural Information Processing Systems 36 (NeurIPS)*, vol. 36, 2024.

[211] W. Yao, Y. Sun, A. Ho, C. Sun, and K. Zhang, "Learning Temporally Causal Latent Processes from General Temporal Data," *International Conference on Learning Representations (ICLR)*, 2022.

[212] W. Yao, G. Chen, and K. Zhang, "Temporally Disentangled Representation Learning," *Advances in Neural Information Processing Systems 35 (NeurIPS)*, vol. 35, pp. 26492–26503, 2022.

[213] B. Huang, C. Lu, L. Leqi, J. M. Hernández-Lobato, C. Glymour, B. Schölkopf, and K. Zhang, "Action-Sufficient State Representation Learning for Control with Structural Constraints," *International Conference on Machine Learning (ICML)*, pp. 9260–9279, 2022.

[214] P. Lippe, S. Magliacane, S. Löwe, Y. M. Asano, T. Cohen, and S. Gavves, "CITRIS: Causal Identifiability from Temporal Intervened Sequences," *International Conference on Machine Learning (ICML)*, pp. 13557–13603, 2022.

[215] P. Lippe, S. Magliacane, S. Löwe, Y. M. Asano, T. Cohen, and E. Gavves, "Causal Representation Learning for Instantaneous and Temporal Effects in Interactive Systems," *International Conference on Learning Representations (ICLR)*, 2023.

[216] P. N. Johnson-Laird, "Mental Models and Human Reasoning," *Proceedings of the National Academy of Sciences*, vol. 107, no. 43, pp. 18243–18250, 2010.

[217] A. S. Polydoros and L. Nalpantidis, "Survey of Model-Based Reinforcement Learning: Applications on Robotics," *Journal of Intelligent & Robotic Systems*, vol. 86, no. 2, pp. 153–173, 2017.

[218] T. M. Moerland, J. Broekens, A. Plaat, and C. M. Jonker, "Model-based Reinforcement Learning: A Survey," *Foundations and Trends in Machine Learning*, vol. 16, no. 1, pp. 1–118, 2023.

[219] F.-M. Luo, T. Xu, H. Lai, X.-H. Chen, W. Zhang, and Y. Yu, "A Survey on Model-Based Reinforcement Learning," *Science China Information Sciences*, vol. 67, no. 2, p. 121101, 2024.

[220] D. Hafner, T. Lillicrap, I. Fischer, R. Villegas, D. Ha, H. Lee, and J. Davidson, "Learning Latent Dynamics for Planning from Pixels," *International Conference on Machine Learning (ICML)*, 2019.

[221] D. Hafner, T. Lillicrap, J. Ba, and M. Norouzi, "Dream to Control: Learning Behaviors by Latent Imagination," *International Conference on Learning Representations (ICLR)*, 2020.

[222] D. Hafner, T. Lillicrap, M. Norouzi, and J. Ba, "Mastering Atari with Discrete World Models," *International Conference on Learning Representations (ICLR)*, 2021.

[223] D. Hafner, J. Pasukonis, J. Ba, and T. Lillicrap, "Mastering Diverse Domains through World Models," *arXiv preprint arXiv:2301.04104*, 2023.

[224] M. Watter, J. Springenberg, J. Boedecker, and M. Riedmiller, "Embed to Control: A Locally Linear Latent Dynamics Model for Control from Raw Images," *Advances in Neural Information Processing Systems 28 (NIPS)*, vol. 28, 2015.

[225] E. Banijamali, R. Shu, M. Ghavamzadeh, H. Bui, and A. Ghodsi, "Robust Locally-Linear Controllable Embedding," *International Conference on Artificial Intelligence and Statistics*, pp. 1751–1759, 2018.

[226] M. Zhang, S. Vikram, L. Smith, P. Abbeel, M. Johnson, and S. Levine, "SOLAR: Deep Structured Representations for Model-Based Reinforcement Learning," *International Conference on Machine Learning (ICML)*, 2019.

[227] Z.-M. Zhu, X.-H. Chen, H.-L. Tian, K. Zhang, and Y. Yu, "Offline Reinforcement Learning with Causal Structured World Models," *arXiv preprint arXiv:2206.01474*, 2022.

[228] R. P. Poudel, H. Pandya, and R. Cipolla, "Contrastive Unsupervised Learning of World Model with Invariant Causal Features," *arXiv preprint arXiv:2209.14932*, 2022.

[229] R. Moraffah, M. Karami, R. Guo, A. Raglin, and H. Liu, "Causal Interpretability for Machine Learning - Problems, Methods and Evaluation," *ACM SIGKDD Explorations Newsletter*, vol. 22, no. 1, pp. 18–33, 2020.

[230] F. Doshi-Velez and B. Kim, "Towards a Rigorous Science of Interpretable Machine Learning," *arXiv preprint arXiv:1702.08608*, 2017.

[231] L. H. Gilpin, D. Bau, B. Z. Yuan, A. Bajwa, M. Specter, and L. Kagal, "Explaining Explanations: An Overview of Interpretability of Machine Learning," *2018 IEEE 5th International Conference on Data Science and Advanced Analytics (DSAA)*, 2018.

[232] D. V. Carvalho, E. M. Pereira, and J. S. Cardoso, "Machine Learning Interpretability: A Survey on Methods and Metrics," *Electronics*, vol. 8, no. 8, p. 832, 2019.

[233] P. Linardatos, V. Papastefanopoulos, and S. Kotsiantis, "Explainable AI: A Review of Machine Learning Interpretability Methods," *Entropy*, vol. 23, no. 1, p. 18, 2020.

[234] K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation," *arXiv preprint arXiv:1406.1078*, 2014.

[235] Y. Bengio, T. Deleu, N. Rahaman, R. Ke, S. Lachapelle, O. Bilaniuk, A. Goyal, and C. Pal, "A Meta-Transfer Objective for Learning to Disentangle Causal Mechanisms," *arXiv preprint arXiv:1901.10912*, 2019.

[236] P. Brouillard, S. Lachapelle, A. Lacoste, S. Lacoste-Julien, and A. Drouin, "Differentiable Causal Discovery from Interventional Data," *Advances in Neural Information Processing Systems 33 (NeurIPS)*, vol. 33, 2020.

[237] R. Y. Rohekar, S. Nisimov, Y. Gurwicz, and G. Novik, "From Temporal to Contemporaneous Iterative Causal Discovery in the Presence of Latent Confounders," *International Conference on Machine Learning (ICML)*, 2023.

[238] A. Gopnik, C. Glymour, D. M. Sobel, L. E. Schulz, T. Kushnir, and D. Danks, "A Theory of Causal Learning in Children: Causal Maps and Bayes Nets," *Psychological Review*, vol. 111, no. 1, 2004.

[239] T. L. Griffiths and J. B. Tenenbaum, "Structure and Strength in Causal Induction," *Cognitive Psychology*, vol. 51, no. 4, pp. 334–384, 2005.

[240] T. L. Griffiths and J. B. Tenenbaum, "Theory-Based Causal Induction," *Psychological Review*, vol. 116, no. 4, pp. 661–716, 2009.

[241] D. C. Penn and D. J. Povinelli, "Causal Cognition in Human and Nonhuman Animals: A Comparative, Critical Review," *Annual Review of Psychology*, vol. 58, pp. 97–118, 2007.

[242] D. A. Braun, C. Mehring, and D. M. Wolpert, "Structure Learning in Action," *Behavioural Brain Research*, vol. 206, no. 2, pp. 157–165, 2010.

[243] K. Cobbe, C. Hesse, J. Hilton, and J. Schulman, "Leveraging Procedural Generation to Benchmark Reinforcement Learning," *International Conference on Machine Learning (ICML)*, 2020.

[244] A. Gopnik, A. N. Meltzoff, and P. K. Kuhl, *The Scientist in the Crib: Minds, Brains, and How Children Learn*. William Morrow & Co, 1999.

[245] L. Schulz, "The Origins of Inquiry: Inductive Inference and Exploration in Early Childhood," *Trends in Cognitive Sciences*, vol. 16, no. 7, pp. 382–389, 2012.

[246] H. G. Schmidt, S. M. Loyens, T. Van Gog, and F. Paas, "Problem-Based Learning is Compatible with Human Cognitive Architecture: Commentary on Kirschner, Sweller, and Clark (2006)," *Educational Psychologist*, vol. 42, no. 2, pp. 91–97, 2007.

[247] S. M. Loyens, J. Magda, and R. M. Rikers, "Self-Directed Learning in Problem-Based Learning and its Relationships with Self-Regulated Learning," *Educational Psychology Review*, vol. 20, no. 4, pp. 411–427, 2008.

[248] L. Alfieri, P. J. Brooks, N. J. Aldrich, and H. R. Tenenbaum, "Does Discovery-Based Instruction Enhance Learning?," *Journal of Educational Psychology*, vol. 103, no. 1, p. 1, 2011.

[249] F. Khan, B. Mutlu, and J. Zhu, "How Do Humans Teach: On Curriculum Learning and Teaching Dimension," *Advances in Neural Information Processing Systems 24 (NIPS)*, vol. 24, 2011.

[250] F. Poli, G. Serino, R. Mars, and S. Hunnius, "Infants Tailor Their Attention to Maximize Learning," *Science Advances*, vol. 6, no. 39, p. eabb5053, 2020.

[251] A. Ten, P. Kaushik, P.-Y. Oudeyer, and J. Gottlieb, "Humans Monitor Learning Progress in Curiosity-Driven Exploration," *Nature Communications*, vol. 12, no. 1, p. 5972, 2021.

[252] A. F. Baranes, P.-Y. Oudeyer, and J. Gottlieb, "The Effects of Task Difficulty, Novelty and the Size of the Search Space on Intrinsically Motivated Exploration," *Frontiers in Neuroscience*, vol. 8, p. 317, 2014.

[253] Y. Bengio, J. Louradour, R. Collobert, and J. Weston, "Curriculum Learning," *International Conference on Machine Learning (ICML)*, pp. 41–48, 2009.

[254] S. Sukhbaatar, Z. Lin, I. Kostrikov, G. Synnaeve, A. Szlam, and R. Fergus, "Intrinsic Motivation and Automatic Curricula via Asymmetric Self-Play," *arXiv preprint arXiv:1703.05407*, 2017.

[255] C. Florensa, D. Held, X. Geng, and P. Abbeel, "Automatic Goal Generation for Reinforcement Learning Agents," *International Conference on Machine Learning (ICML)*, pp. 1515–1528, 2018.

[256] A. Jabri, K. Hsu, A. Gupta, B. Eysenbach, S. Levine, and C. Finn, "Unsupervised Curricula for Visual Meta-Reinforcement Learning," *Advances in Neural Information Processing Systems 32 (NeurIPS)*, vol. 32, 2019.

[257] S. Narvekar, B. Peng, M. Leonetti, J. Sinapov, M. E. Taylor, and P. Stone, "Curriculum Learning for Reinforcement Learning Domains: A Framework and Survey," *Journal of Machine Learning Research*, vol. 21, pp. 1–50, 2020.

[258] E. Kosoy, D. M. Chan, A. Liu, J. Collins, B. Kaufmann, S. H. Huang, J. B. Hamrick, J. Canny, N. R. Ke, and A. Gopnik, "Towards Understanding How Machines Can Learn Causal Overhypotheses," *arXiv preprint arXiv:2206.08353*, 2022.

[259] E. Sumner, A. X. Li, A. Perfors, B. Hayes, D. Navarro, and B. W. Sarnecka, "The Exploration Advantage: Children's Instinct to Explore Allows Them to Find Information that Adults Miss," *PsyArXiv preprint*, 2019.

[260] E. S. Sumner, M. Steyvers, and B. W. Sarnecka, "It's Not the Treasure, It's the Hunt: Children Are More Explorative on an Explore/Exploit Task than Adults," *CogSci*, pp. 2891–2897, 2019.

[261] E. G. Liquin and A. Gopnik, "Children are More Exploratory and Learn More than Adults in an Approach-Avoid Task," *Cognition*, vol. 218, p. 104940, 2022.

[262] J. X. Wang, M. King, N. Porcel, Z. Kurth-Nelson, T. Zhu, C. Deck, P. Choy, M. Cassin, M. Reynolds, F. Song, *et al.*, "Alchemy: A Benchmark and Analysis Toolkit for Meta-Reinforcement Learning Agents," *Proceedings of the NeurIPS 2021 Datasets and Benchmarks Track*, 2021. arXiv preprint arXiv:2102.02926.

[263] C. Jiang, N. R. Ke, and H. van Hasselt, "Learning How to Infer Partial MDPs for In-Context Adaptation and Exploration," *arXiv preprint arXiv:2302.04250*, 2023.

[264] J. Pearl, "Causal Inference without Counterfactuals: Comment," *Journal of the American Statistical Association*, vol. 95, no. 450, pp. 428–431, 2000.

[265] C. Glymour, P. Spirtes, and R. Scheines, "Causal Inference," *Erkenntnis*, vol. 35, no. 1-3, pp. 151–189, 1991.

[266] P. Soviany, R. T. Ionescu, P. Rota, and N. Sebe, "Curriculum Learning: A Survey," *International Journal of Computer Vision*, vol. 130, no. 6, pp. 1526–1565, 2022.

[267] A. Graves, M. G. Bellemare, J. Menick, R. Munos, and K. Kavukcuoglu, "Automated Curriculum Learning for Neural Networks," *International Conference on Machine Learning (ICML)*, pp. 1311–1320, 2017.

[268] M. Shi and V. Ferrari, "Weakly Supervised Object Localization using Size Estimates," *European Conference on Computer Vision (ECCV)*, pp. 105–121, 2016.

[269] P. Soviany, R. T. Ionescu, P. Rota, and N. Sebe, "Curriculum Learning: A Survey," *arXiv preprint arXiv:2101.10382*, 2021.

[270] C. Li, M. Zhang, and Y. He, "Curriculum Learning: A Regularization Method for Efficient and Stable Billion-Scale GPT Model Pre-Training," *OpenReview preprint*, 2021.

[271] S. Arora and A. Goyal, "A Theory for Emergence of Complex Skills in Language Models," *arXiv preprint arXiv:2307.15936*, 2023.

[272] Q. Li, Y. Zhai, Y. Ma, and S. Levine, "Understanding the Complexity Gains of Single-Task RL with a Curriculum," *International Conference on Machine Learning (ICML)*, 2023.

[273] J. L. Elman, "Learning and Development in Neural Networks: The Importance of Starting Small," *Cognition*, vol. 48, no. 1, pp. 71–99, 1993.

[274] P.-Y. Oudeyer, F. Kaplan, and V. V. Hafner, "Intrinsic Motivation Systems for Autonomous Mental Development," *IEEE Transactions on Evolutionary Computation*, vol. 11, no. 2, pp. 265–286, 2007.

[275] C. Romac, R. Portelas, K. Hofmann, and P.-Y. Oudeyer, "TeachMyAgent: a Benchmark for Automatic Curriculum Learning in Deep RL," *International Conference on Machine Learning (ICML)*, pp. 9052–9063, 2021.

[276] R. Bellman, "A Markovian Decision Process," *Journal of Mathematics and Mechanics*, pp. 679–684, 1957.

[277] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. Cambridge, MA: MIT Press, 2nd ed., 2018.

[278] M. Jiang, E. Grefenstette, and T. Rocktäschel, "Prioritized Level Replay," *International Conference on Machine Learning (ICML)*, 2021.

[279] M. McCloskey and N. J. Cohen, "Catastrophic Interference in Connectionist Networks: The Sequential Learning Problem," *Psychology of Learning and Motivation*, vol. 24, pp. 109–165, 1989.

[280] J. Kirkpatrick, R. Pascanu, N. Rabinowitz, J. Veness, G. Desjardins, A. A. Rusu, K. Milan, J. Quan, T. Ramalho, A. Grabska-Barwinska, *et al.*, "Overcoming Catastrophic Forgetting in Neural Networks," *Proceedings of the National Academy of Sciences*, vol. 114, no. 13, pp. 3521–3526, 2017.

[281] M. Toneva, A. Sordoni, R. T. d. Combes, A. Trischler, Y. Bengio, and G. J. Gordon, "An Empirical Study of Example Forgetting During Deep Neural Network Learning," *arXiv preprint arXiv:1812.05159*, 2018.

[282] C. V. Nguyen, A. Achille, M. Lam, T. Hassner, V. Mahadevan, and S. Soatto, "Toward Understanding Catastrophic Forgetting in Continual Learning," *arXiv preprint arXiv:1908.01091*, 2019.

[283] R. C. Atkinson, "Optimizing the Learning of a Second-Language Vocabulary," *Journal of Experimental Psychology*, vol. 96, no. 1, p. 124, 1972.

[284] N. Kornell and J. Metcalfe, "Study Efficacy and the Region of Proximal Learning Framework," *Journal of Experimental Psychology: Learning, Memory, and Cognition*, vol. 32, no. 3, p. 609, 2006.

[285] L. P. Kaelbling, M. L. Littman, and A. W. Moore, "Reinforcement Learning: A Survey," *Journal of Artificial Intelligence Research*, vol. 4, pp. 237–285, 1996.

[286] Y. Li, "Deep Reinforcement Learning," *arXiv preprint arXiv:1810.06339*, 2018.

[287] S. E. Li, "Deep Reinforcement Learning," *Reinforcement Learning for Sequential Decision and Optimal Control*, pp. 365–402, 2023.

[288] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, "Playing Atari with Deep Reinforcement Learning," *arXiv preprint arXiv:1312.5602*, 2013.

[289] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, *et al.*, "Human-Level Control through Deep Reinforcement Learning," *Nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[290] O. Vinyals, I. Babuschkin, W. M. Czarnecki, M. Mathieu, A. Dudzik, J. Chung, D. H. Choi, R. Powell, T. Ewalds, P. Georgiev, *et al.*, "Grandmaster Level in StarCraft II using Multi-Agent Reinforcement Learning," *Nature*, vol. 575, no. 7782, pp. 350–354, 2019.

[291] OpenAI, M. Andrychowicz, B. Baker, M. Chociej, R. Jozefowicz, B. McGrew, J. Pachocki, A. Petron, M. Plappert, G. Powell, A. Ray, *et al.*, "Learning Dexterous In-Hand Manipulation," *The International Journal of Robotics Research*, vol. 39, no. 1, pp. 3–20, 2020.

[292] D. L. Rohde and D. C. Plaut, "Language Acquisition in the Absence of Explicit Negative Evidence: How Important is Starting Small?," *Cognition*, vol. 72, no. 1, pp. 67–109, 1999.

[293] X. Wang, Y. Chen, and W. Zhu, "A Survey on Curriculum Learning," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2021.

[294] R. Portelas, C. Colas, L. Weng, K. Hofmann, and P.-Y. Oudeyer, "Automatic Curriculum Learning For Deep RL: A Short Survey," *International Joint Conference on Artificial Intelligence (IJCAI)*, 2021.

[295] R. Wang, J. Lehman, J. Clune, and K. O. Stanley, "Paired Open-Ended Trailblazer (POET): Endlessly Generating Increasingly Complex and Diverse Learning Environments and Their Solutions," *arXiv preprint arXiv:1901.01753*, 2019.

[296] R. Wang, J. Lehman, A. Rawal, J. Zhi, Y. Li, J. Clune, and K. Stanley, "Enhanced POET: Open-Ended Reinforcement Learning through Unbounded Invention of Learning Challenges and their Solutions," *International Conference on Machine Learning (ICML)*, pp. 9940–9951, 2020.

[297] J. Z. Leibo, E. Hughes, M. Lanctot, and T. Graepel, "Autocurricula and the Emergence of Innovation from Social Interaction: A Manifesto for Multi-Agent Intelligence Research," *arXiv preprint arXiv:1903.00742*, 2019.

[298] M. Chevalier-Boisvert, L. Willems, and S. Pal, "Minimalistic Gridworld Environment for Gymnasium." `https://github.com/Farama-Foundation/Minigrid`, 2018.

[299] B. Tang, M. A. Lin, I. Akinola, A. Handa, G. S. Sukhatme, F. Ramos, D. Fox, and Y. Narang, "IndustReal: Transferring Contact-Rich Assembly Tasks from Simulation to Reality," *Robotics: Science and Systems (RSS)*, 2023.

[300] M. Dennis, N. Jaques, E. Vinitsky, A. Bayen, S. Russell, A. Critch, and S. Levine, "Emergent Complexity and Zero-Shot Transfer via Unsupervised Environment Design," *Advances in Neural Information Processing Systems 33 (NeurIPS)*, vol. 33, pp. 13049–13061, 2020.

[301] K. J. Åström, "Optimal Control of Markov Processes with Incomplete State Information," *Journal of Mathematical Analysis and Applications*, vol. 10, no. 1, pp. 174–205, 1965.

[302] L. P. Kaelbling, M. L. Littman, and A. R. Cassandra, "Planning and Acting in Partially Observable Stochastic Domains," *Artificial Intelligence*, vol. 101, no. 1-2, pp. 99–134, 1998.

[303] Y. Abbasi-Yadkori and G. Neu, "Online Learning in MDPs with Side Information," *arXiv preprint arXiv:1406.6812*, 2014.

[304] A. Hallak, D. Di Castro, and S. Mannor, "Contextual Markov Decision Processes," *arXiv preprint arXiv:1502.02259*, 2015.

[305] A. Modi, N. Jiang, S. Singh, and A. Tewari, "Markov Decision Processes with Continuous Side Information," *Algorithmic Learning Theory*, vol. 83, pp. 597–618, 2018.

[306] J. Parker-Holder, M. Jiang, M. Dennis, M. Samvelyan, J. Foerster, E. Grefenstette, and T. Rocktäschel, "Evolving Curricula with Regret-Based Environment Design," *International Conference on Machine Learning (ICML)*, pp. 17473–17498, 2022.

[307] M. Jiang, M. Dennis, J. Parker-Holder, J. Foerster, E. Grefenstette, and T. Rocktäschel, "Replay-Guided Adversarial Environment Design," *Advances in Neural Information Processing Systems 34 (NeurIPS)*, vol. 34, pp. 1884–1897, 2021.

[308] R. Portelas, C. Colas, K. Hofmann, and P.-Y. Oudeyer, "Teacher Algorithms for Curriculum Learning of Deep RL in Continuously Parameterized Environments," *Proceedings of the Conference on Robot Learning (CoRL)*, 2020.

[309] M. Samvelyan, A. Khan, M. Dennis, M. Jiang, J. Parker-Holder, J. Foerster, R. Raileanu, and T. Rocktäschel, "MAESTRO: Open-Ended Environment Design for Multi-Agent Reinforcement Learning," *International Conference on Learning Representations (ICLR)*, 2023.

[310] I. Mediratta, M. Jiang, J. Parker-Holder, M. Dennis, E. Vinitsky, and T. Rocktäschel, "Stabilizing Unsupervised Environment Design with a Learned Adversary," *Conference on Lifelong Learning Agents (CoLLAs)*, pp. 270–291, 2023.

[311] M. Beukman, S. Coward, M. Matthews, M. Fellows, M. Jiang, M. Dennis, and J. Foerster, "Refining Minimax Regret for Unsupervised Environment Design," *arXiv preprint arXiv:2402.12284*, 2024.

[312] J. Bruce, A. Anand, B. Mazoure, and R. Fergus, "Learning about Progress from Experts," *International Conference on Learning Representations (ICLR)*, 2023.

[313] R. Bommasani, D. A. Hudson, E. Adeli, R. Altman, S. Arora, S. von Arx, M. S. Bernstein, J. Bohg, A. Bosselut, E. Brunskill, *et al.*, "On the Opportunities and Risks of Foundation Models," *arXiv preprint arXiv:2108.07258*, 2021.

[314] J. Hill, "Real Time Control of a Robot with a Mobile Camera," *Proceedings of the 9th International Symposium on Industrial Robots*, pp. 233–245, 1979.

[315] S. Hutchinson, G. D. Hager, and P. I. Corke, "A Tutorial on Visual Servo Control," *IEEE Transactions on Robotics and Automation*, vol. 12, no. 5, pp. 651–670, 1996.

[316] V. Veitch, A. D'Amour, S. Yadlowsky, and J. Eisenstein, "Counterfactual Invariance to Spurious Correlations: Why and How to Pass Stress Tests," *arXiv preprint arXiv:2106.00545*, 2021.

[317] M. Makar, B. Packer, D. Moldovan, D. Blalock, Y. Halpern, and A. D'Amour, "Causally Motivated Shortcut Removal Using Auxiliary Labels," *arXiv preprint arXiv:2105.06422*, 2021.

[318] M. A. Goodrich and A. C. Schultz, "Human-Robot Interaction: A Survey," *Foundations and Trends in Human–Computer Interaction*, vol. 1, no. 3, pp. 203–275, 2008.

[319] A. Hussein, M. M. Gaber, E. Elyan, and C. Jayne, "Imitation Learning: A Survey of Learning Methods," *ACM Computing Surveys (CSUR)*, vol. 50, no. 2, pp. 1–35, 2017.

[320] H. Zhao, H. Chen, F. Yang, N. Liu, H. Deng, H. Cai, S. Wang, D. Yin, and M. Du, "Explainability for Large Language Models: A Survey," *ACM Transactions on Intelligent Systems and Technology*, vol. 15, no. 2, pp. 1–38, 2024.

[321] R. Kashefi, L. Barekatain, M. Sabokrou, and F. Aghaeipoor, "Explainability of Vision Transformers: A Comprehensive Review and New Perspectives," *arXiv preprint arXiv:2311.06786*, 2023.

[322] E. Kıcıman, R. Ness, A. Sharma, and C. Tan, "Causal Reasoning and Large Language Models: Opening a New Frontier for Causality," *arXiv preprint arXiv:2305.00050*, 2023.

[323] M. Zečević, M. Willig, D. S. Dhami, and K. Kersting, "Causal Parrots: Large Language Models May Talk Causality But Are Not Causal," *Transactions on Machine Learning Research*, 2023.

[324] S. J. Gotts, H. J. Jo, G. L. Wallace, Z. S. Saad, R. W. Cox, and A. Martin, "Two Distinct Forms of Functional Lateralization in the Human Brain," *Proceedings of the National Academy of Sciences*, vol. 110, no. 36, pp. E3435–E3444, 2013.

[325] M. C. Corballis, "Left Brain, Right Brain: Facts and Fantasies," *PLoS Biology*, vol. 12, no. 1, p. e1001767, 2014.

[326] Q.-Y. Zhou, J. Park, and V. Koltun, "Open3D: A Modern Library for 3D Data Processing," *arXiv preprint arXiv:1801.09847*, 2018.

[327] S. Katz, A. Tal, and R. Basri, "Direct Visibility of Point Sets," in *ACM SIGGRAPH 2007 papers*, pp. 24–es, ACM, 2007.

[328] M. Ester, H.-P. Kriegel, J. Sander, X. Xu, *et al.*, "A Density-based Algorithm for Discovering Clusters in Large Spatial Databases with Noise," *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining (KDD)*, vol. 96, no. 34, pp. 226–231, 1996.

[329] K. Zhang, M. Sharma, J. Liang, and O. Kroemer, "A Modular Robotic Arm Control Stack for Research: Franka-Interface and FrankaPy," *arXiv preprint arXiv:2011.02398*, 2020.

[330] J. R. Kubricht, K. J. Holyoak, and H. Lu, "Intuitive Physics: Current Research and Controversies," *Trends in Cognitive Sciences*, vol. 21, pp. 749–759, 2017.

[331] D. Ha and J. Schmidhuber, "World Models," *CoRR*, vol. abs/1803.10122, 2018.

[332] D. Hafner, T. P. Lillicrap, M. Norouzi, and J. Ba, "Mastering Atari with Discrete World Models," *CoRR*, vol. abs/2010.02193, 2020.

[333] J. Wang, C. Hu, Y. Wang, and Y. Zhu, "Dynamics Learning With Object-Centric Interaction Networks for Robot Manipulation," *IEEE Access*, vol. 9, pp. 68277–68288, 2021.

[334] O. Kroemer and G. Sukhatme, "Meta-level Priors for Learning Manipulation Skills with Sparse Features," *2016 International Symposium on Experimental Robotics (ISER)*, 2017.

[335] D. J. Wilkinson, *Stochastic Modelling for Systems Biology*. Chapman and Hall/CRC Mathematical and Computational Biology Series, New York, NY: Chapman & Hall/CRC, 2006.

[336] A. J. Lotka, "Contribution to the Theory of Periodic Reactions," *The Journal of Physical Chemistry*, vol. 14, no. 3, pp. 271–274, 1909.

[337] X. Jian and L. Zushu, "Dynamic Model and Motion Control Analysis of Three-link Gymnastic Robot on Horizontal Bar," *International Conference on Robotics, Intelligent Systems and Signal Processing*, vol. 1, pp. 83–87, 2003.

[338] R. M. Murray, Z. Li, and S. S. Sastry, *A Mathematical Introduction to Robotic Manipulation*. CRC Press, 2017.

[339] L. S. Cubit, R. Canale, R. Handsman, C. Kidd, and L. Bennetto, "Visual Attention Preference for Intermediate Predictability in Young Children," *Child Development*, vol. 92, no. 2, pp. 691–703, 2021.

[340] C. Kidd, S. T. Piantadosi, and R. N. Aslin, "The Goldilocks Effect: Human Infants Allocate Attention to Visual Sequences that are Neither Too Simple nor Too Complex," *PloS One*, vol. 7, no. 5, p. e36399, 2012.

[341] C. Kidd, S. T. Piantadosi, and R. N. Aslin, "The Goldilocks Effect in Infant Auditory Attention," *Child Development*, vol. 85, no. 5, pp. 1795–1804, 2014.

[342] L. S. Vgotsky, *Mind in Society: The Development of Higher Psychological Processes*. Harvard University Press, 1978.

[343] J. A. Leonard, S. R. Cordrey, H. Z. Liu, and A. P. Mackey, "Young Children Calibrate Effort Based on the Trajectory of Their Performance," *Developmental Psychology*, vol. 59, no. 3, p. 609, 2023.

[344] A. Ruggeri and T. Lombrozo, "Children Adapt Their Questions to Achieve Efficient Search," *Cognition*, vol. 143, pp. 203–216, 2015.

[345] D. Weinshall, G. Cohen, and D. Amir, "Curriculum Learning by Transfer Learning: Theory and Experiments with Deep Networks," *International Conference on Machine Learning (ICML)*, pp. 5238–5246, 2018.

[346] K. P. Murphy, "Active Learning of Causal Bayes Net Structure," *University of California, Berkeley Technical Report*, 2001.

[347] S. Tong and D. Koller, "Active Learning for Structure in Bayesian Networks," *International Joint Conference on Artificial Intelligence (IJCAI)*, vol. 17, no. 1, pp. 863–869, 2001.

[348] Y.-B. He and Z. Geng, "Active Learning of Causal Networks with Intervention Experiments and Optimal Designs," *Journal of Machine Learning Research*, vol. 9, no. Nov, pp. 2523–2547, 2008.

[349] F. Eberhardt, "Almost Optimal Intervention Sets for Causal Discovery," *arXiv preprint arXiv:1206.3250*, 2012.

[350] C. Squires, S. Magliacane, K. Greenewald, D. Katz, M. Kocaoglu, and K. Shanmugam, "Active Structure Learning of Causal DAGs via Directed Clique Tree," *arXiv preprint arXiv:2011.00641*, 2020.

[351] A. Ghassami, S. Salehkaleybar, N. Kiyavash, and E. Bareinboim, "Budgeted Experiment Design for Causal Structure Learning," *International Conference on Machine Learning (ICML)*, pp. 1724–1733, 2018.

[352] A. Ghassami, S. Salehkaleybar, and N. Kiyavash, "Interventional Experiment Design for Causal Structure Learning," *arXiv preprint arXiv:1910.05651*, 2019.

[353] A. Hyttinen, F. Eberhardt, and P. O. Hoyer, "Experiment Selection for Causal Discovery," *Journal of Machine Learning Research*, vol. 14, pp. 3041–3071, 2013.

[354] M. Kocaoglu, A. Dimakis, and S. Vishwanath, "Cost-Optimal Learning of Causal Graphs," *International Conference on Machine Learning (ICML)*, pp. 1875–1884, 2017.

[355] E. M. Lindgren, M. Kocaoglu, A. G. Dimakis, and S. Vishwanath, "Experimental Design for Cost-Aware Learning of Causal Graphs," *arXiv preprint arXiv:1810.11867*, 2018.

[356] N. Scherrer, O. Bilaniuk, Y. Annadani, A. Goyal, P. Schwab, B. Schölkopf, M. C. Mozer, Y. Bengio, S. Bauer, and N. R. Ke, "Learning Neural Causal Models with Active Interventions," *arXiv preprint arXiv:2109.02429*, 2021.