THE UNIVERSITY
of EDINBURGH

Composing diverse policies for long-horizon tasks

Daniel Angelov Angelov

Doctor of Philosophy

Institute of Perception, Action and Behaviour

School of Informatics

University of Edinburgh

2020

Lay Summary

Humans interact and solve different manipulation and planning tasks using a variety of strategies and naturally adapt to the situation at hand. We alter the manner of execution as we transition from one part of the task to another — e.g. from reaching and pulling a book, to sliding it out and carefully regrasping it in place.

In contrast, when solving such long horizon tasks in the robotics domain, it is common to represent this multistage process using identical building blocks, albeit in a hierarchical fashion.

We argue that homogeneity is a limiting factor when solving diverse tasks. In the first part of the thesis we show how we can embrace representational diversity of the type of controllers — learned by trial and error, using observations or even hand-crafted strategies — to solve long horizon tasks. We rely on demonstrators to show the desired goal, which allows us to learn a way to compare how well the current state matches the desired one. This gives us the ability to sequence the diverse controllers.

In the second part we are interested in understanding what does a learned policy pay attention to. We abstract the human demonstration into this model, and use cause and effect strategy to manipulate what the model sees and track the changes in the output.

Finally, we conclude by looking into how having a 'physical dialogue' when demonstrating a task can improve the robustness of the controller. When the robot's internal strategy coincides with the demonstration, it can in real time nudge into a region where it's more uncertain. This also provides a way to highlight regions of the task that are crucial for completing it successfully.

Abstract

Humans utilise a large diversity of control and reasoning methods to solve different robot manipulation and motion planning tasks. This diversity should be reflected in the strategies used by robots in the same domains. In current practice involving sequential decision making over long horizons, even when the formulation is a hierarchical one, it is common for all elements of this hierarchy to adopt the same representation. For instance, the overall policy might be a switching model over Markov Decision Processes (MDPs) or local feedback control laws. This may not be well suited to a variety of naturally observed behaviours. For instance, when picking up a book from a crowded shelf, we naturally switch between goal-directed reaching, tactile regrasping, sliding the book until it is comfortably off an edge and then once again goal-directed pick and place. It is rare that a single representational form adequately captures this diversity, even in such a seemingly simple task.

When the robot must learn or adapt policies from experience, this poses significant challenges. The mis-match between the representational choices and the diversity of task types can result in a significant (sometimes exponential) increase in complexity with respect to time, observation and state-space dimensionality and other attributes. These and other factors can make the learning of such tasks in a "tabula rasa" setting extremely difficult. However, if we were willing to adopt a multi-representational framing of the problem, and allow for some of these constituent modules to be learned in different ways (some from expert demonstration, some by trial and error, and perhaps some being controllers designed from first principles in model-based formulations) then the problem becomes much more tractable. The core hypothesis we explore is that it is possible to devise such learning methods, and that they significantly outperform conventional alternatives on robotic manipulation tasks of interest.

In the first part of this thesis, we present a framework for sequentially composing diverse policies facilitating the solution of long-horizon tasks. We rely on demonstrations to provide a quick, not necessarily expert and optimal, way to convey the desired outcome. We model the similarity to demonstrated states in a *Goal Scoring Estimator model*. We show in a real robot experiment the benefits of diverse policies relying on their own strong inductive biases to efficiently solve different aspects of the task, through sequencing by the Goal Scoring Estimator model.

Next, we demonstrate how we can elicit *policy structure* through causal analysis and *task structure* through more efficient demonstrations involving interventions. This allows us to alter the manner of execution of a particular policy to match a desired learned user specification. Building a surrogate model of the demonstrator gives us the ability to causally reason about different aspects of the policy and which parts of that policy are salient. We can observe how intervening in the world by placing additional symbols impacts the validity of the original plan.

Finally, observing that 'static' imitation learning datasets can be limiting if we are aiming to create more robust policies, we present the *Learning from Inverse Intervention framework*. This allows the robot to simultaneously learn a policy while interacting with the demonstrator. In this interaction, the robot intervenes when there is little information gain and pushes the demonstrator to explore more informative areas even as the demonstration is being performed in real-time. This interaction brings the added benefit of drawing out information about the importance of different regions of the task. We verify the salience by visually inspecting samples from a generative model and by crafting plans that test these hypothetical areas.

These methods give us the ability to use demonstrations of a task, to build policies for salient targets, to alter their manner of execution and inspect to understand the causal structure, and to sequence them to solve novel tasks.

"The purpose of abstraction is not to be vague,
but to create a new semantic level in which one can be absolutely precise."
— Edsger Dijkstra

Acknowledgments

It is commonly said that the PhD period is some of the most creative, stressful and rewarding. The people around me, physically and spiritually, have had a big impact on my ability to enjoy and grow during this time.

First and foremost, I want to thank my supervisor Subramanian Ramamoorthy. He not only provided the ability to explore the academic world, but also truly grow as a researcher.

I thank my family for their unwavering support and in their constant belief that being a good person is of utmost importance.

I want to thank all the people in the lab, with whom I have spend countless days and nights — the RAD group has been a truly supportive space — as they have had a dramatic multiplier effect on the creativity and intellectual stimulation. I am grateful for the lunch time conversations, which have always raised more questions than have answered.

Specifically, I want to thank Svet, Yordan, Alex, Manny, who during the InSpace days made sure I was not lost in the sea of knowledge and helped me steer the ship of research towards new land.

This work was supported by the Engineering and Physical Sciences Research Council, as part of the CDT in Robotics and Autonomous Systems at Heriot-Watt University and The University of Edinburgh under Grant reference EP/L016834/1.

Declaration

I declare that this thesis was composed by myself, that the work contained herein is my own except where explicitly stated otherwise in the text, and that this work has not been submitted for any other degree or professional qualification except as specified.

Daniel Angelov Angelov

Publications

The following publications have been composed during the course of this doctorate:

D. Angelov, Y. Hristov, S. Ramamoorthy. From demonstrations to task-space specifications. Using causal analysis to extract rule parameterization from demonstrations. Journal of *Autonomous Agents and Multi-Agent Systems* (JAAMAS), Vol. 34(45), 2020. [7]

D. Angelov, Y. Hristov, M. Burke, S. Ramamoorthy. Composing Diverse Policies for Temporally Extended Tasks. *IEEE Robotics and Automation Letters* (RA-L), Vol. 5(2), 2020. [4]

D. Angelov, Y. Hristov, S. Ramamoorthy. DynoPlan: Combining Motion Planning and Deep Neural Network based Controllers for Safe HRL. In Proc. *Multidisciplinary Conference on Reinforcement Learning and Decision Making* (RLDM), 2019. [5]

D. Angelov, Y. Hristov, S. Ramamoorthy. Using Causal Analysis to Learn Specifications from Task Demonstrations. In Proc. *International Conference on Autonomous Agents and Multiagent Systems* (AAMAS), 2019. [6]

D. Angelov, S. Ramamoorthy. Learning from Demonstration of Trajectory Preferences through Causal Modeling and Inference. *Robotics: Science and Systems Workshop on Causal Imitation in Robotics* (R:SS CIR), 2018. [8]

D. Angelov, S. Ramamoorthy. LfII: Learning from Inverse Intervention during Demonstrations, 2021.

To my family.

Contents

List of Figures

List of Tables

Acronyms

AI	Artificial Intelligence
BC	Behavioral Cloning
BO	Bayesian Optimization
CNN	Convolutional Neural Network
DMP	Dynamic Movement Primitive
DRL	Deep Reinforcement Learning
GAN	Generative Adversarial Network
GMM	Gaussian Mixture Model
HRL	Hierarchical Reinforcement Learning
IRL	Inverse Reinforcement Learning
KL	Kullback—Leibler
LfD	Learning from Demonstration
LfII	Learning from Inverse Intervention
LQR	Linear Quadratic Controller
MDN	Mixture Density Network
MDP	Markov Decision Process
MPC	Model Predictive Control
NN	Neural Network
RL	Reinforcement Learning
SCM	Structural Causal Model
VAE	Variational Autoencoder

1

Introduction

1.1 Preface

Robotics has been strongly marching into several of the Industry 4.0 technologies [128]. The advancement of sensory capabilities as well as the need for more flexible robotics drives the introduction of learning processes within the industry. As the business needs have expanded beyond the structured and engineered 'pick and place' tasks, the techniques to support a changing supply of parts with a greater degree of variability have been rapidly developing.

As industrial based tasks such as assemblies or chemical processes move towards wider customization and 'lean' manufacturing with tight delivery deadlines (the prime effect [99]), this naturally increases the diversity of processes involved. These processes can be hand-engineered, removing slack from the critical path, or some parts slowly shifting to learning based methods — learn exhaustively once, quickly fine-tune per task. In this transition period, we are left with an open question on *"How does one combine the diversity of control strategies?"*

Sequencing policies requires innovation, as relying on the current strategies of Reinforcement Learning (RL) or Model Predictive Control (MPC) framework enforces unneeded homogeneity in the controller strategies, which is exactly what is desired to be avoided. A controller is a policy, a hand-specified or learned strategy that takes in observations of the world and produces a control signal that when executed by the agent, changes the world in a desired fashion. These methods usually attempt to capture the complexity or the dynamics of the task into a specific model, for all parts of the task. This approach, however, does not provide the desired application

diversity. We instead show the ability to quickly generate novel hierarchical controllers, relying on multiple heterogeneous systems, for new tasks learning from just a few demonstrations. Further, we examine the underlying learned policy to gain more fine grained properties for both the hierarchy and the structure of the policy itself.

In current engineered processes, the main methods for automation vary according to the human-robot ratio of in the workforce. At one end of the spectrum, we have Henry Ford's vision of the assembly line, where rope-and-pulleys are used to move the car along different worker stations. This simple solution allowed for the time needed to produce a car to be cut in half compared to bringing parts by a horse carriage. At the other end, completely automated pipelines — "lights out" manufacturing — complete their product with zero human intervention. Bemusingly, an example of such manufacturing is the production of integrated circuits and robots themselves [94, 110].

However, the above examples are specialized in domains that require little work on diverse or challenging engineering tasks. A parallel development in automation is targeting the collaborative aspect of work — with human and robot workers jointly performing some operation. This relatively recent advancement works by positioning the two close by and delegating the responsibility for doing the "tricky" for robots tasks to the human worker and saving time on the assembly line by having the robot prioritize easily automatable tasks.

More challenging tasks, such as manipulating wire and flexible materials, are still considered outside the scope of the capabilities of even modern robotics targeting industrial applications.

As an example, let us take one of the latest Tesla patent applications — Figure 1. Being a company without the burden and constraint of an established manufacturing process, they are perceived to be taking an exploratory approach that aims to change both the standard manufacturing processes and the product itself to ease production. The figure shows a wiring harness of a car, which is designed to be rigidified to make it manipulable by robots. Even with such changes, the process of connecting the power bars requires significant sensing and manipulation capabilities (1b) — the bars need to be aligned both in position and orientation, delicate and small spacers inserted in-between, and bolts slotted in and tightened to a particular torque level. Some of

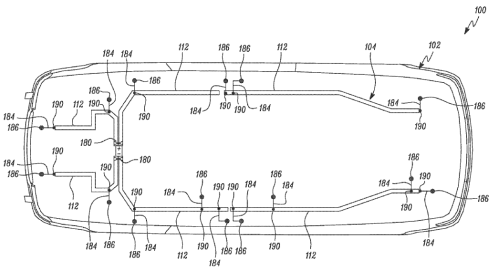


FIG. 1b

(a) Full body semi-flexible wiring.

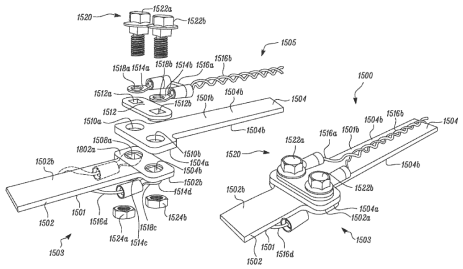


FIG. 15a FIG. 15b

(b) Bolt fastened connectors.

Figure 1: Example tasks of the Tesla wiring harness. Images adapted from [29].

these sub-tasks have natural solutions in the current process automation pipelines through standard motion planning or engineering processes; others that deal with flexible materials or unconstrained insertion require more innovative methods. The flexibility allows for the shape to alter in a continuous fashion within boundaries defined by the material, forcing the robot to sense the particular shape and react accordingly. Each of these contingencies needs to be evaluated and programmed individually, forcing the robot to reason online.

1.1.1 *Skills Composition*

These types of tasks highlight the diversity of skills, and by proxy controllers, that needs to be executed in sequence to complete these relatively straightforward for a human operator to demonstrate tasks. From a research perspective, each of those previously highlighted skills has been viewed in isolation, yet solving the task holistically has been reliant on a single process structure. This rigidity sidesteps some of the advantages where a solution is developed, e.g. in a simulation, using a (learned) physics model or completely data-driven. As a result, introducing a new model of a product by creating a modification of either a piece or step of the process may make all the skill acquired for the current task obsolete.

What is required is a flexible method that makes the process of sequencing skills from different domains easy, with just a few demonstrations. It should naturally be robust to disturbances and auto-correct what skill needs to be used from a library of available high-level actions.

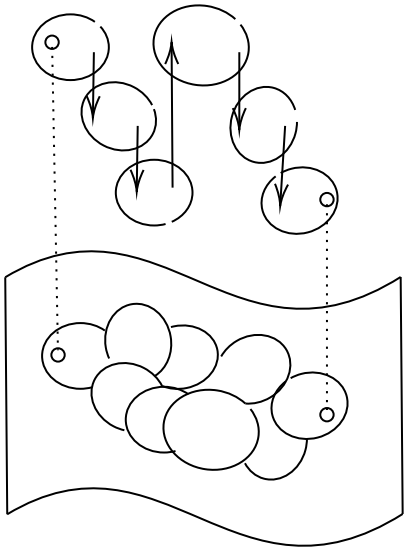


Figure 2: Composing controllers allows stable movement between any two points covered by such controllers.

Inspiration for the outlined solution comes from the concept of covering the state space of a task with controllers and navigating between different points within by sequencing them as in [26]. The controllers represent a particular small problem sub-task — a short-horizon skill and can be activated in a particular case — a subset of the task-space we call an initiation region. This initiation region governs where the current skill is contextually useful to be executed and is associated with the corresponding performance metrics. Additionally, a skill is associated with a flag that can terminate it — a useful signal to highlight that the skill has been completely executed and another decision about what to do next can be initiated. Composition is assumed to be the sequential application of one function and policy after another.

The diversity of skills allows for them to work on a variety of input sensor streams - from camera and depth information, state representations, to proprioception. These modalities are aligned with different output spaces - from poses to joint angles and work at different horizons and frame-rates. Not surprisingly, they also provide a different set of guarantees, which can be necessary for operation within certain domains [103].

As a competing alternative, the algorithm designer has to choose a particular representation, on top of which the skills can be developed for the particular problem.

For robotics it is common to rely on RGB-D (video camera and depth information) in cases that require precise and contextual information and control the robot end-effectors in Cartesian space. It is a resource intensive process that requires careful mapping. On the other end of the spectrum, when quick decisions are important, a high frame-rate camera can be used to torque control the joints of the robot. The current frameworks lack an intuitive way to combine the variability to styles different problems require.

1.1.2 *Choosing Sequences from Library*

In the proposed work, the compositionality comes from sequencing these skills through overlapping regions of initiation and area of operation. For instance, in the bolt inserting task, the controller can be activated when the context makes sense - the robot is holding a bolt and a connector is visible. The skill should not necessarily be conditioned on the number of connectors, their alignment, or whether they have spacers.

Finding the right sequence of controllers relies upon a task policy. It can be obtained by using expert demonstrations. This allows for a quick and intuitive method to provide instructions about the task and the goal itself. Rather than relying upon the recreation of reward associated with different steps along the trajectory, like Inverse Reinforcement Learning (IRL), or directly learning to replicate the behaviour as in Learning from Demonstration (LfD), we can learn a more straightforward representation. It provides an assessment of how far along the current state is in regards to the demonstrations of the task.

Combining this ability with a dynamics model of each controller, which can estimate what the sequence of future states for a given policy would be, we can evaluate the futures in a hypothetical selection of each possible controller. Intuitively, we can then select the skill that forces the world to be closest to solving the task. This provides a method that uses demonstrations to guide the skill selection process for solving long-horizon tasks.

1.1.3 *Skills Elicitation*

If we have demonstrations of the task, what else can we do? So far, we have relied on knowing what the substructure of the task is by the explicit annotations of the different skills used. However, whilst observing the full demonstration, if we perform varying interventions either in the resulting model input or during the demonstrations themselves, we can begin to understand the underlying *structure of the problem*.

One strategy would involve performing interventions in the environment of the task, and through causal analysis extracting the causal strength of the change. By using this type of augmentation, we can extract the items or symbols within the world that the expert is paying attention to during the demonstration. Symbols are assumed to be an instances or a visual instantiation of a particular object. For instance, the trajectory of how the bolts are inserted may be dependant on whether parallel tasks — such as connecting the battery — are performed, which may alter the degree of caution with which the task is done. Through such modifications of the environment, we can learn to differentiate between the states or preferences of the expert.

Moreover, we can begin to decompose the demonstrations to a set of rules that represent the internal state of the expert — e.g. "Stay away by at least X from the battery". By parameterizing these symbolic rules, we can use them as part of the latter task optimization procedure and as a result, create much more natural behaviour. This capability also increases the potential diversity of policies in the skills library.

1.1.4 *Policy decomposition*

Previously, we were assessing the policy in a static, off-policy fashion after completely fine-tuning the controllers and applying them to varying situations to extract the embedded information.

An alternative strategy to decompose the policy would be to look at interventions during training of the demonstration itself. This has the increased benefit of incrementally gaining insight into the policy alongside its increased performance. Current methods for blending expert and robot control do this sequentially — the robot executes a policy, with the human providing clarification actions when the robot

is uncertain, overlaying the correct action after the fact, or alternatively, the human expert having the capability to intervene within the robot policy (and provide better actions). But much more information can be obtained about the salience of different regions of the task space by both performing interactive interventions and allowing the *robot* to elicit information by augmenting the demonstration in real-time (rather than the human augmenting the robot policy).

The interactivity allows for the physical dialogue between the robot and the expert to extract information in real-time and to avoid the hard task of post-factum annotating trajectories. The augmentation of the trajectories allows for the "curious" robot to avoid repetitions in parts of the environment with enough data support, and move the demonstrator in a more robot-informative situation. This means that reaching a particular level of performance or robustness would need fewer demonstrations. This communication is performed by the robot nudging or moving to an internally desired area of the state space.

These methods for composition and skill elicitation would allow for machine learning policies to be learned in a data-efficient way, alter the policy to follow a different expert specification, and be part of a much longer task.

1.2 Problem Statement

In this thesis we consider the problem of sequencing diverse policies to solve long-horizon tasks and how can we decompose the tasks into sub-tasks. To perform this, we look at embedding different priors in the control model: (1) expert demonstrations - these provide a reasonable, close to an optimal demonstration of the task and convey the expected outcome, (2) causal analysis - building a Structural Causal Model (SCM) or intervening in the demonstration provide a method to better extract the underlying structure of the task and what is required for its completion.

We are tackling the questions:

- Given a library of diverse controllers, their dynamic models and expert demonstrations, how can we sequence them into a plan to solve temporally extended tasks?

- Assuming demonstrations of the low-level policies, can we create a surrogate model to evaluate the causal link between different symbols or objects in the environment, thus decomposing the policies, and the demonstrators' used specifications for the task? Can we interpolate between different specifications or map them to a rule-based system?
- Can we partition and extract more information about the structure of the task from interactions during demonstrations? Can this be a method to also obtain more diverse, data efficient demonstrations given robots and humans have comfort spaces with subconscious task and specification related preferences?

1.3 Thesis Overview

The thesis is split into distinct parts studying the composition of diverse policies and the different methods of finding task structure. The first represents the need for finding methods to combine policies of different nature - deliberate motion planning or reactive neural network policies into a single hierarchical controller. In the second part, we will expand on some of the common issues with hierarchical control encountered previously around decomposing the structure of a task and subsequent policy refinement.

1.3.1 *Composing Diverse Policies*

Tasks that have changing dynamics can be represented as hybrid systems and usually implemented as a hierarchical controller with the critical points initiating a change of controllers. A model-free approach would, in turn, be responsible for learning either the underlying policies, how to sequence them or both. This would mean that the resulting system is represented with the same paradigm, as the underlying controllers - e.g. a neural network, a decision tree, a program.

This restriction limits the type of problems that can simultaneously be tackled due to the constraints each structure exhibits on the system - in terms of control, input state space, control loop frequency, how the boundaries of the controllers are estimated.

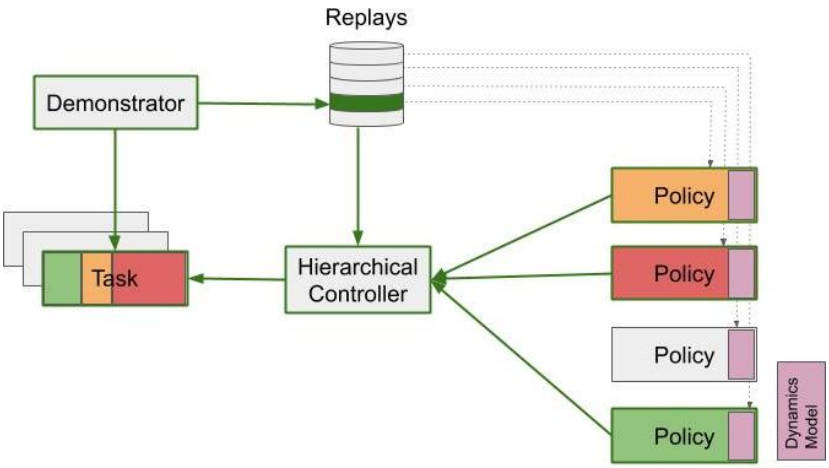


Figure 3: The hierarchical controller uses demonstrations as well as the estimated future states under the different policies to choose the best next controller.

Using demonstrations, we can bias a hierarchical controller (Figure 3) about the progress represented with the current state of the task. As a result, if each of the sub-controllers can predict the future state of the system conditioned on its execution, we can transform the sequence learning problem into a planning one, whilst being agnostic to the underlying controllers.

In particular, we learn a Goal Score Estimator model that approximates the progress of an observational state towards the desired configuration. With a low number of varying length demonstrations, it maps the state space to a scalar value in the range $[0..1]$, where 0 is a state close the beginning of demonstrations, 1 - towards the desired/end world state. We rely on the versatility of direct demonstrations to provide the information to learn this estimator. Even though the demonstrations are not temporally aligned, the flexibility of a Mixture Density Network (MDN) can learn the variability in the demonstrations.

This allows using a diverse set of controllers that work within the same system by incorporating a dynamics model. Those sub-policies can be tuned and work on a variety of state spaces and can form a library of controllers to be used across different long-horizon tasks.

We evaluate this architecture on a base Markov Decision Process (MDP) problem to show optimality in the compositionality of our system for tasks exhibiting a variety of time-scale dependant and skill diverse problems. We test its robustness to noise in the dynamics and Goal Score Estimator models and show the low sensitivity to varying

conditions. Finally, we show that it performs better than any individual method in the gear assembly task performed on the PR2 robot.

1.3.2 *Structure Elicitation*

1.3.2.1 *Policy sub-structure*

Common everyday tasks exhibit natural variations of their execution, which is often due to a small change in the specifications under which users perform them. We can learn a generative model of the different user behaviours, where the latent space samples represent a world configuration with the desired solution. This gives the capability to find solutions to unseen problems by dynamically reevaluating a proposed initial guess and backpropagating the desired configuration through the differentiable specifications model. With each iteration or step in the latent space, the solution converges closer to a possible configuration under the specifications for the generative model.

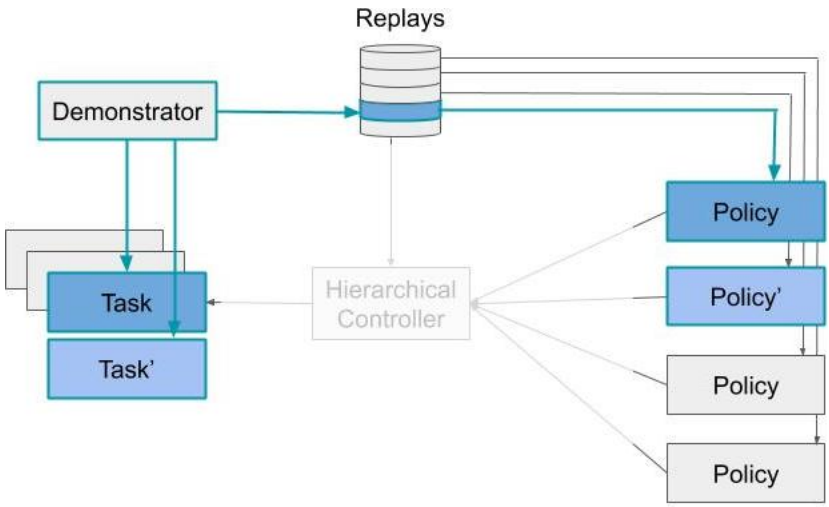


Figure 4: Using demonstrators that generate solutions to similar tasks, we can create a specification that differentiates between them and allows to interpolate between the solution of the resulting policies.

This allows an initial set of policies to be expanded to include a wider variance in the styles with which a task is solved. Further, we can interpolate between those policies and allow for the hierarchical controller to augment the final solution.

Additionally, through the generative model we can find which symbols have an impact on the user's specifications. We perform causal analysis by intervening in the

surrogate model of the user, allowing us to carry out an exhaustive search. By relaxing our assumptions and gaining knowledge about the symbols and their location in the environment we can learn to differentiate between specifications and what symbols are impacting each one of them.

Further, we can use constraint optimization to parameterise the interaction between the symbols and the proposed solution trajectory and as a result learn basic safety envelopes under which motion planning can be performed.

1.3.2.2 *Task sub-structure*

Sharing different skills between humans, especially focused on tactile control, are generally transferred through demonstrations. It is not well known which aspect of the structure of the demonstration is crucial and it is common to describe the action or require more information in a Q&A session, or alternatively to participate and nudge during a possible demonstration.

We want to emulate this type of inquisitive behaviour in a robot agent, in which it is an active participant in the demonstration. This differs from passively observing or optimizing a policy post-hoc using additional labels. By being an active participant, the learner can perturb and see if the motion is being counteracted or not by the expert. This type of information gathering can explain if the resulting trajectory is a variant — an additional degree of freedom — of a known sub-policy, or is actively deteriorating the expert performance — it informs about the salience of the current region.

We bring active learning to LfD with Learning from Inverse Intervention (LfII), where during a demonstration, the learner compares historically the alignment between the internal agent policy and the current demonstration to trigger an intervention that moves the agent in a different part of the state space that can be more novel. This iterative process allows to augment the demonstrations and make them more informative.

This type of intervention can also be used to elicit the structure of the task. By perturbing the policy to avoid a sub-region of the task space, we can evaluate the salience by determining if the resulting trajectory has solved the task. If the task is successfully completed, the avoided region is not important and vice-versa.

Alternatively, we can query a generative model and ask the demonstrator to evaluate the salience of the highlighted state. By iterating over a set of trajectory states, we can use this Visual Q&A strategy to decompose it into important sub-regions.

1.4 Major Contributions

- A strategy for composing diverse policies from a set in a pre-existing library to solve long-horizon tasks. Converting a learning problem into a planning/sequencing one by using expert demonstrations to learn a Goal Scoring Estimator. This improves tractability and robustness by fusing well established methods with learning based ones shown in a PR2 gear assembly problem. [Chapter 3]
- Using causal analysis to gain insight into the specifications users follow when completing different tasks in relation to known symbols in the world. Further, the ability to find clusters of possible solutions and to interpolate trajectories across different specifications in a tabletop trajectory generation task. [Chapter 4]
- LfII: an imitation learning strategy that augments the demonstrations in real-time and forces a higher diversity in the obtained data. It adds the ability to decompose the policy structure by evaluating the salience of different parts of the task space by either intervening in the policy to avoid a region of the space or using Visual Q&A to query the expert directly. [Chapter 5]

2

Background

The big shift in applying universal approximator functions to learning based problems, especially after 2013, builds on top of a period defined by handcrafting features to perform subsequent tasks. Even though these features could be learned from data, e.g. Histogram of Gradients (HoG) [38], Scale-invariant Feature Transform (SIFT) [97] they were a product of careful selection.

The machine learning boom that followed is mainly defined by the abundance of well-labeled data [20, 42, 93, 96], increasing computational power [68] and ever-increasing and diverse methods of applying these universal approximator functions [85, 91, 116, 135, 140, 141].

But, as suggested by the *"no free lunch"* theorem, we have been approaching the limits of these general approaches. The new wave of improvements will come from looking back into embedding some form of stronger bias to escape this local minimum.

The improvements within this thesis are relying on two fundamental inductive priors:

1. **Expert Demonstrations** - Learning from Demonstration provides an unmatched method that shows the goal, method, and manner of execution to complete particular tasks. It is also extremely well suited for robotics and tasks of varying complexity.
2. **Causal Modelling** - Fundamentally, the world we live in exhibits a causal structure, which combined with the robot's ability to interact within the same world, gives a strong prior about the relationship between observations, actions, and future states.

We will be discussing those on a fundamental level here and will be providing a cohesive targeted summary as part of each chapter.

Alongside demonstrations and causal analysis, we believe that having an understanding of current methods for composition and data assessment to be necessary, which can also be found at the end of the chapter.

2.1 Imitation Learning for Robotics

2.1.1 *Learning from Demonstration*

LfD is a paradigm that gives the ability for robots/agents to learn a particular skill. This is based on the assumption that the observation of an expert performing the task gives the necessary information to extract a corresponding robot controller. Contrasting the previous methods that rely on decomposing the problem and writing explicitly a controller or program to do the task; or providing a reward structure and iterating a policy until it reaches the desired performance. The big advantage of LfD is that the demonstrator does not need to be an expert in robot programming, rather, they need to be sufficiently familiar with the task.

Let us look back into the Tesla wiring harness example. A factory floor worker needs only to provide examples of the sequence and manner in which the harness needs to be connected, without explicitly programming. This is preferred, as it is hard in most situations for the domain experts to completely explain and articulate thoroughly most of the variability and reasoning for actions (especially ones made by "their gut feeling") [28]. In cases where the performance is sub-par, in a learning situation, more demonstrations can be provided to allow the algorithm to generalize better (more on generalization and how much data is needed in Section 2.4). This is different than *copy and replay* of the original demonstrations, rather it is creating a policy that replicated the observed behavior.

This is in stark contrast to the current process where software experts are on the factory floor, directly optimizing the robotic processes to achieve better amortization of the robot. In this setting, the engineer needs to reason about all of the variations

and sub-tasks that the robot needs to perform and how they are linked to the general structure.

In its foundation, LfD is the problem of learning a mapping between the states and desired demonstrated actions. However, there exists a difference between how this mapping is used when there is a mismatch between the recorded mapping or the embodiment mapping — Figure 5. When performing demonstrations we come across exactly the *correspondence problem* [9, 105].

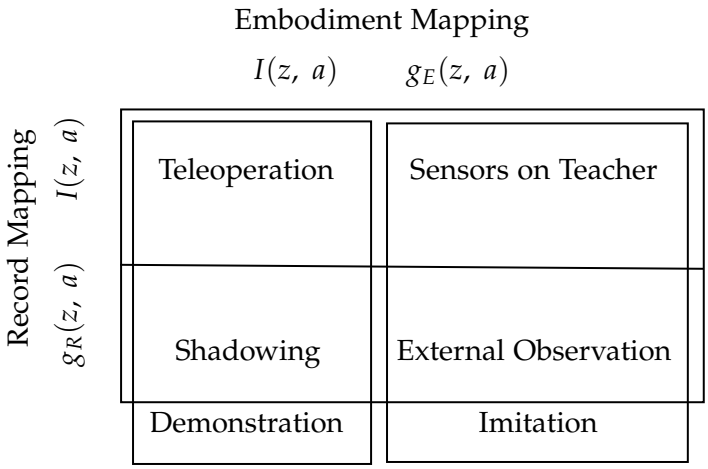


Figure 5: Transformations between the recorded medium and the embodiment of the demonstration or imitation. Inspired from [9].

Even though robots and humans perform work in a correspondingly similar environment and aligned physical capabilities, there still exists a non-identity mapping in some cases. However, there are two broad cases where this equivalence can hold — perceptual and physical equivalence.

2.1.1.1 *Physical equivalence*

When performing a task, humans, and robots may share the underlying physical platform. For instance, in the case of teleoperation or shadowing, the demonstrator can use the same hardware to complete the task but rely on identical or different perception streams. This physical equivalence makes the task of action generation much easier, as the direct actions are provided during the demonstration. Additionally, the robot agent may be equipped with additional sensors, such as depth or thermal cameras, potentially easing the perception task.

We rely on this equivalence for Chapter 3, where we use HTC Vive controllers to remotely teleoperate the PR2 robot, demonstrating a task, but relying on our vision to obtain task context. During the demonstration, proprioceptive and visual information from the robot was temporally aligned.

Similarly, Chapter 4 relies on kinesthetic teaching [123], where the expert is physically moving the robot joints as part of the demonstration. One significant difference is that the resulting visual data is different from the expected run-time distribution (see section. 2.4).

2.1.1.2 *Perceptual equivalence*

Similarly, in cases where the expert receives information in the same manner as the robot platform, through teleoperation or by instrumenting sensors on the teacher, there exists the ability to build up the right latent representation of the task. The closer these mappings are to an identity transformation, the easier it is to learn a representative policy. However, in the case of perceptual equivalence, it may not necessarily be straightforward to replicate the action results. In the case of the wire harness assembly, the human demonstrator motion around obstacles is constrained by the redundancy of the human body. As the kinematic chain and operational space may differ, the obtained demonstrations can have replication issues.

2.1.2 *Applications*

The ease with which demonstrations can provide an example of the task needed to be completed, without having to explicitly specify a goal, makes them an ideal candidate for specifying new tasks. The concept of non-experts modifying the behavior of an agent is shown in [60], where a robot policy is learned to play robot football (soccer), [18] — drive a car. This Behavioral Cloning (BC) has been extended to even more dynamic contexts, with a much narrowly distributed expert skill set, such as acrobatic drone flying [1], or in case of [89] — learning the representative hierarchical policy.

However, demonstrations can also be used to learn safe policies under the supervision [74], or as part of an auxiliary task of learning mappings for the perception and kinematic differences [9].

On a fundamental level, the demonstrations are providing an implicit representation of the goal of the task. Looking through the lens of IRL, we can invert the problem of learning a policy to learning a reward function, which can be later on used in an RL optimization setting [10, 109]. This two-step process allows us to learn more general policies, even beyond the expert performance [22].

2.2 Causal Modelling

One of the methods for embedding inductive bias in machine learning is to rely on causal analysis to extract a cause-effect relationship between the different parts of the demonstration. Here will discuss how interventions and counterfactuals work, as well as examples of how they have been used in the machine learning and robotics domains.

2.2.1 *Causal Analysis*

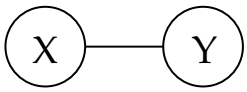


Figure 6: A small graphical model containing two variables - X and Y.

Let us being by considering a small Graphical Model as shown in Figure 6. We can model this relationship as follows

$$X := N_X \tag{1}$$

$$Y := f(X, N_Y) \tag{2}$$

where X and Y are nodes in the graph, N_X and N_Y are independent and identically distributed (iid) noise variables. f denotes a connecting function that represents how two variables are related. It can range from a simple linear function - $Y = \theta X + N_Y$ to

complex non-linear ones that increase the dimensionality of the resulting variable. We can then apply simple mathematical rearrangement and invert the dependency on a function g for X — $X := g(Y, N_X); Y := N_Y$.

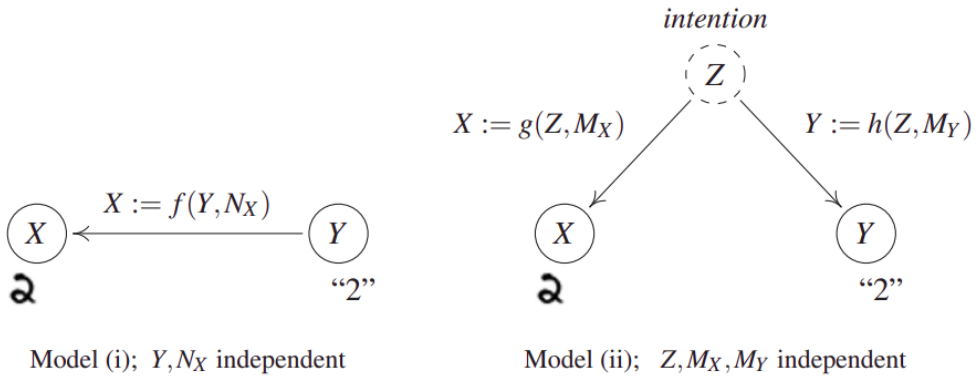


Figure 7: Two different Structural Causal Model (SCM). In the first one, the label is given to a parson, who creates the alternative representation. In the latter model, the intention of the human generates both the label and the image representation. For particular set of functions f, g, h and noise variables N_X, M_X, M_Y the two models would produce equivalent distributions of samples, but are naturally different from an intervention perspective. Adapted from [115].

Figure 7 are examples of SCM for a case where samples from the MNIST dataset is generated [113, 115]. It indicates both the relationship between the variables and also the direction of influence. In model (i) we have that both the class label Y and the noise sample N_X are independent ($Y \perp\!\!\!\perp N_X$) — the exact style of the letter written is not related to the class of the letter. In this setting, we assume we have a human which when given a class produces with best effort an image representing the class. We can compute the *observational distribution* $P_{X,Y}$ from P_Y, P_{N_X}, and f, that is we can learn the mapping from x to y without intervening in the system, better than chance.

In this SCM there are two possible interventions we can perform and produce the corresponding *interventional distribution*. We can intervene on X by changing the image, but it does not affect the class label, the human writer, or the data in the dataset. Mathematically, any changes on X will have no impact on Y, because of $Y := N_Y$.

However, intervening on Y has the impact of changing what the writer observes as a task, and as a result — the image produced. This is because $X := f(Y, N_X)$ and X is dependent on the variable Y. This is the reason why in the graph, there exists directionality in the connection between X and Y, i.e. there is an arrow linking the nodes.

In model (ii) we also model the writer's intention for a class and the resulting image. We can apply *Reichenbach's common cause principle*, which states that in the case where we have two statistically dependent variables X, Y ($X \not\perp\!\!\!\perp Y$), there exists a third variable that is a confounder - Z, which causally influences both. Or in some special cases, it can coincide with either one of the other variables. What is even more important is that in the resulting graph, the two variables X and Y are shielded, such that if we condition on Z, they become independent ($X \perp\!\!\!\perp Y|Z$).

If we choose suitable functions and noise variables, the resulting observational distribution $P_{X,Y}$ can match the distribution in model (i). If we intervene on X, we have the same result as in model (i) — no change is observed. However, if we intervene on Y, we would not affect the image! This is different than in the first case for model (i), as the image generation function is independent of Y in model (ii).

In SCM, dependencies are generated by functions that compute variables from other variables. Rather than thinking of them as mathematical equations, it is better to assume they are computational programs, indicating that mathematical rearrangements may not hold true due to the causal structure of the graph.

More generally, when we think of causal modeling, we understand the process of drawing conclusions based on a causal model.

Formally, for the above example we can say that there exists an SCM $\mathfrak{C}$ defined as $\mathfrak{C} := (S, P_N)$. S is a collection of d structural assignments in the form of:

$$X_j := f_j(PA_j, N_j), \qquad j = 1, ..., d \tag{3}$$

where PA_j is the parent of (nodes that influence/directly cause) X_j, $PA_j = \{X_1, ..., X_d\} \backslash \{X_j\}$ and P_N is a joint distribution of the independent noise variable for each node.

We can calculate a distribution over the variables X, which we refer to as the *entailed distribution* or $P_X^{\mathfrak{C}}$.

2.2.1.1 *Interventions*

Now that we have the basis for working with a SCM, let's assume we intervene and change the probability of a particular variable X_k. We would then expect the distribution of the SCM to also change. It is equivalent to changing the graph, such that now X_k is dependent on the new intervention distribution. So, even if it was previously influenced by other variables, now it would be separated by modifying $\mathfrak{C}$ and would change the entailed distribution.

We can write it as modifying the assignment for X_k as:

$$X_k = \tilde{f}(\tilde{PA}_k, \tilde{N}_k) \tag{4}$$

Alternatively, we can also set $\tilde{f}$ to be a point mass of a particular value a. Thus we can call the entailed distribution of the new SCM and *intervention distribution* and annotate it as:

$$P_X^{\tilde{\mathfrak{C}}} := P_X^{\mathfrak{C};do(X_k =: \tilde{f}(\tilde{PA}_k, \tilde{N}_k))} \tag{5}$$

The intervention distribution in the modified graph is updated by setting the distribution of X_k to the new modified assignment. If we choose $\tilde{f}(\tilde{PA}_k, \tilde{N}_k) \neq f(PA_k, N_k)$ we would expect that there will be a shift in distributions $P_{X_j}^{\tilde{\mathfrak{C}}} \neq P_{X_j}^{\mathfrak{C}}$ if X_j is causally influenced by X_k.

2.2.1.2 *Counterfactuals*

Using counterfactuals for reasoning[1] can be considered as the process of conditioning the noise variables on particular observations and performing an intervention to measure the resulting distribution. In essence, it allows us to answer questions in the form *"In this situation x, if I had done Y, what would the outcome of Z be?"*.

We can condition on some value of the nodes $X = x$ as follows:

$$\mathfrak{C}_{X=x} := (S, P_N^{\mathfrak{C}|X=x}) \tag{6}$$

1 It is commonplace to use counterfactual reasoning in causal inference, but is debated around the ability to articulate the full graph or to limit the effects of the conditioning on the rest of the graph [106].

where the joint probability distribution of the original gets projected on the distribution when $X = X$, $P_N^{\mathfrak{C}|X=x} := P_{N|X=x}$, so the new noise variables no longer need to be i.i.d. and can be estimated from $X = x$ and the definitions of X. And in this new graph (with conditioned noise variables), we can further perform the intervention action to estimate the probability of Z and note it as:

$$P_Z^{\mathfrak{C}|X=x;\ do(Y=y)} \tag{7}$$

2.2.2 *Causality in robotics*

Causal analysis is an especially important tool in decision making for financial and medical regulations and institutions. It is particularly hard to evaluate in those domains, as performing interventions need to have an observation of both the entitled and interventional distributions, meaning part of the population is influenced by some aspect, and the rest - not. This raises a lot of ethical questions, as a critical part of the decisions can be done only from observational data (for more see Chapter 2.4). The gold standard in those domains, when possible, is to perform double-blind placebo-controlled randomized trials. This is to avoid spurious causal relationships that are an artefact of information leak regarding the arm of the trial or statistical correlations in the data that are not part of the same causal graph (or have a common base parent, beyond the scope of the experiment). The latter is a significant issue if the reason for the investigation is the formation of the graph itself, where the lack of scientific and common knowledge is unable to articulate fulling the domain of the problem.

Having an active participant in the world (i.e. the robot platform or surrogate model) allows us to simplify the hard task of performing these interventions and measuring the resulting distributions.

The ability to decompose the world representations into the different axis of variations [30, 69] allows us to build a latent space[2] that can be mapped into nodes from an SCM. As a result, the different hypotheses can be used to evaluate the machine learning model and the relationship of the task to the representations within the environment. We explore this further in Chapter 4.

2 non-linear neural network dimensionality reduction

Using observational causal analysis can still be useful to provide proxy variables that can be used to incorporate static knowledge within the domain of the problem [44, 120]. This provides a unique ability to shape the inference of the system by both relationships extracted from data and those provided by an expert.

But the interactions with the expert should not be unidirectional, and the ability to create a surrogate model of the system, which can be linked to symbolic representations of the world is useful to provide a causal meta-analysis and explanation of the network's actions [65] and further in Section 4.5.

As part of an RL framework, counterfactuals can be used to alleviate one of the big issues surrounding sparse reward, which is credit assignment [51]. This ability of the system to compare the reward distributions when the agent has the hindsight of a different action distribution can be beneficial to narrow the reward states. Similarly, these counterfactuals can be used to reason about intent in dynamical environments [19].

We further look into this ability to both query the system using interventions and obtain a better understanding of its working as part of Chapter 4. But also in a counterfactual setting to obtain salience of different regions of the space by having a robot manipulate a demonstration as part of Chapter 5.

2.3 Policy Composition

In Section 2.1 we have seen one way to generate appropriate policies that solve a particular task. However, it is of interest to solve problems beyond short horizon skill-based ones and shift to more multi-stage, longer assignments. That in itself requires the ability to compose, or chain, multiple skill-based policies.

2.3.1 *Flat Policy Composition*

Before we continue to a hierarchical composition, we need to highlight that the idea of splitting the state space and having regional controllers was first introduced as part of the optimal control framework [26]. Sequentially composing policies was used as a tool to create simple regional controllers that are tuned and maintain stability and

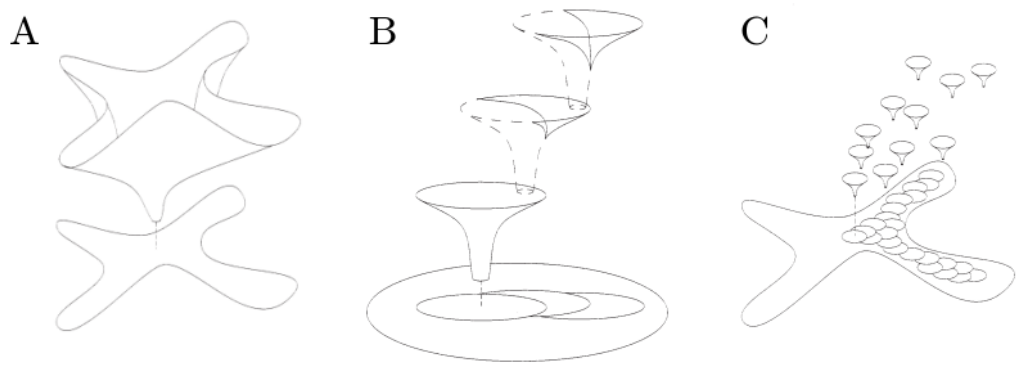


Figure 8: A - an ideal controller in the obstacle-free space. B - the sequential composition of the controllers, where the goal point of each controller is within the attraction space of the following one. C - Sequencing multiple controllers within the free-space of the problem in cells, where each cell has a single active controller. Adapted from [26].

controllability within the domain of operation. By having such overlapping regional controllers, the state of the system can be moved within the operational domains — Figure 8.

This is also used by Tedrake et al. [142] as LQR-Trees to allow stable locomotion by splitting the domain of operation into the above-highlighted tree structure using Linear Quadratic Controller (LQR) controllers.

The advantage of these splits is that the linearization that occurs as part of the controller is an adequate approximation of the local domain and can be independently optimized.

2.3.2 *Hierarchical Control and Long Horizon Tasks*

Temporal abstraction, or the notion that an agent can use not only low-level actions but also hierarchically abstracted actions, has been a long part of the additional inductive bias given to a learning agent [45, 48, 75, 90]. More on how inductive bias is integrated as part of the data used for learning in Section 2.4.

These abstractions have been balanced between using a set of known control strategies or learning the underlying hierarchy. The latter case can be best described as the options [117, 139] in the RL framework.

Similarly, learning Hierarchical Reinforcement Learning (HRL) policies are a method of balancing the exponentially growing state representations, such that de-

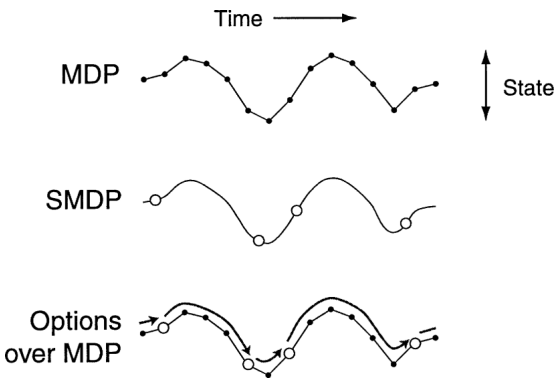


Figure 9: The state trajectory in the MDP is repented as a sequence of states visited through time. By increasing the abstractions of the action — compressing to larger continuous transitions, less decisions need to be made by the agent to reach a desired state. Adapted from [139]

cisions are not required at every step in a semi-Markov decision process [13]. This is done by invoking their own temporally extended activities and relying on the termination criteria to bring back control. We describe our method of sequencing the policies in Chapter 3, where rather than relying on the Bellman update equation to learn this sequencing, we are using expert demonstrations to provide a natural timing to activate the policies.

This balance of using experts as a warm start of policies is especially relevant to long-horizon tasks [61]. The demonstrations can be used to produce an initial distribution that can be learned in a supervised fashion, and subsequently fine-tuned for performance in an RL optimization loop.

We can also look at the work of Andrychowicz et al. [3] as an intervention strategy to the goal of the problem, such that rather than relying on an external demonstration, the mistakes that the policy makes can be used as hindsight into what the target value should have been.

On a macro level, long-horizon tasks have been well understood in the planning domain. Similar attempts for learning actions within a policy have also shown improvements in performance where replays of the system are possible [63]. In our work, described in Chapter 3 we use MPC alongside a heuristic to plan in a multi-policy setting.

Of course, to provide a mixture of policies in a problem context, we need to be able to verify the domain of stable operations of each one of them. This is necessary as

a precaution to ensure the trustworthiness of the overall system performance, where unlike the reward structure in RL, we want to define hard constraints that should not be violated. This problem can be naturally looked as a Bayesian Optimization (BO) of providing adversarial counterexamples [56] or as out-of-distribution detection [151].

As improvements in HRL have allowed for options to be learned in an end-to-end fashion, there has been a rising question of *what* would be considered a good option. In [64] it has been viewed as an additional cost to an option to increases interpretability. Or in the case around Chapter 5, we believe that having a salient target for each option is a method to split the trajectory into meaningful temporal abstractions.

2.3.3 *Policy Diversity*

Both flat and hierarchical policies can be represented in diverse ways. Especially for controllers that require rich manipulation, there have been a variety of strategies dedicated to pushing and pulling [53, 102, 103], dynamic nonprehensile controllability and planning [101], automatic synthesis [98], caging [119].

For policies requiring the robot to exert force or follow a longer trajectory, plain Dynamic Movement Primitive (DMP) [76, 127], or deep DMP [54] have been used.

When we have a dataset obtained from LfD that uses discrete actions, we can partition the space and convert the problem to a classification one with Neural Network (NN) [79], Bayesian Networks [77], Gaussian Mixture Model (GMM) [32, 145]. This is applicable to high dimentional video inputs as demonstrated by [49, 95, 154] by using Convolutional Neural Network (CNN) and Variational Autoencoder (VAE).

2.4 Data Assessment

At its foundation, machine learning has decreased the cost and time of making an inference, or a decision. Previously, the situation would be passed to a committee or an expert, who would in turn use the accumulated experience to provide an adequate interpretation of the case and as a result - a decision. The learning of the process and experience, that the human has, are trying to model the general variability of the representation of the data. Let us take, for example, the highlighted case in Figure 1b

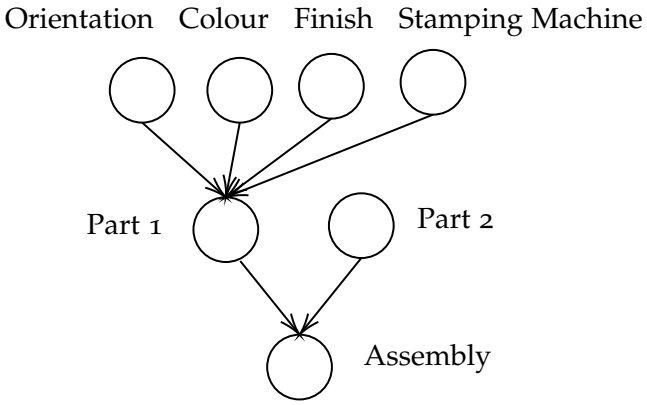


Figure 10: Assembly characterization by a graph.

and the task of visually inspecting the correctness of the assembly. The observations within the images can come from a graphical model that captures the possible modeled aspects (nodes). The image of the assembly comes from parts (Figure 10), which in respect are defined by attributes such as orientation, color, finish, etc.

When we investigate each of these attributes in more detail, we begin to understand that each of them in respect comes from a distribution, defined by the manufacturing process. They can come in several colors, there is some assumed orientation, and additional parameters defining the finish.

As a result, we are left with the finding that through our observation of the phenomena, we capture a subset of that natural variation — Figure 12. Our aim is to record as best as we can this variation in our dataset. Through the process of removing duplicates and collating samples, we produce an ever decreased subset of the original

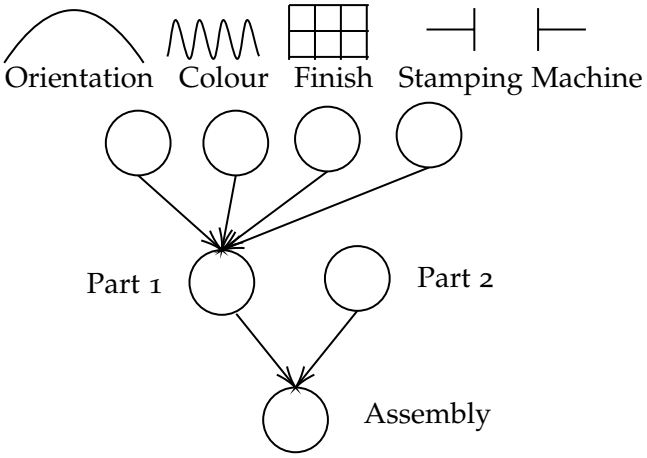


Figure 11: Propagation of the initial node distributions as part of the variability of the assembly.

data. Further, we would then split into training, testing, and evaluation parts - with the hope that we can independently model the natural variation within the data, as well as capture independent samples that can be used to evaluate the performance during and after training.

In the neural networks optimization process it is common to use loss functions such as ones based on Kullback—Leibler (KL) divergence, where there exists an optimization incentive to prefer unbiased, balanced data.

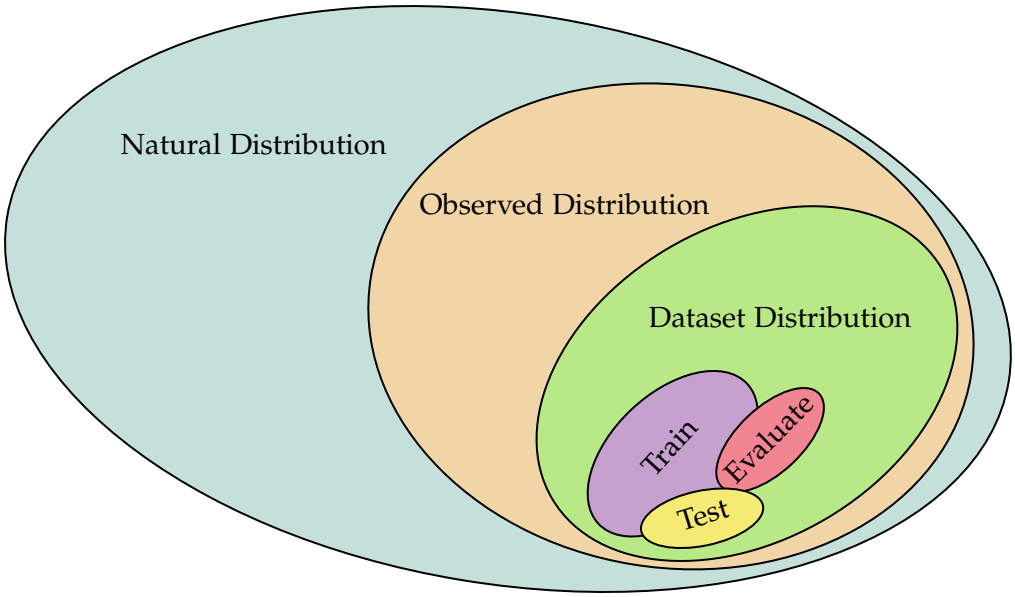


Figure 12: Data diversity between nature and the captured subset in our datasets.

2.4.1 *Dataset curation*

The process of creating a targeted dataset is described as part of one of the most popular datasets - ImageNet [42]. There are significant logistics and licensing issues around surfacing the needed images, as well as providing multi-layer verification of the created labels. Those parts have been since then the core IP of several data-acquisition companies — LabelBox, Hive, Cloudfactory, hCaptcha — highlighting the significant need for these services.

Regardless of all of the precautions around the original ImageNet labeling and data distributions, datasets should not be considered static pieces of information.

Rather, as an iterative process, arising to service a desired machine learning model needs. Beyond the original use as a benchmark, subsequent work on reassessing the labels [16, 122] has shown that there can be significant issues with the aggregated distributed labeling process. In the case of policy learning [35] — bias, lack of generalization ability, and out of distribution samples make static dataset collection an unreasonable assumption.

On further investigation, it has been shown the widely spread bias associated with the data by portraying a predominantly western-centric view of the world. This has led to expansions of OpenImages [14, 93] to version 6, as well as crowdsourced suggestions to classes and NeurIPS Inclusive Images Challenge that aims to stress test image classifiers across new geographic distributions.

From a robotics perspective, the demonstrations collected for a particular task can exhibit similar issues. However, there are further special types of considerations that have to be taken into account. The physical manifestation of the environment, plays only a part of the data collection pipeline, with additional axis of variation across the demonstrated trajectory itself, task sequence, the interaction of the trajectory with the environment, or the manner of execution of the trajectory. Those include implicit biases — e.g. is the demonstration quick or slow, smooth or jittery, and explicit ones — e.g. this is how to perform 'task X', and other task related specifications, that the machine learning system has to work with. From an optimization perspective, both types are created equal, but should be treated differently.

While in static datasets, one strategy of overcoming biases is on collecting new images to diversify the distribution, the demonstration related data can have additionally other methods. Adding new trajectories can be much more time consuming, than images, but the agent interacting with the demonstrator is a modality not available in the first case. A strategy, further expanded in Chapter 5, provides an unsupervised way to augment the data collection process through disturbing the teacher in parts of the demonstration, where the internal agent model is aligned well.

In essence, iterations on the data attempt to expand the dataset distribution to better match the global world observed distribution. However, if we have a large or dynamic dataset, how do we select data that is informative?

2.4.2 *Exploration, Active learning*

When we have a process from which we can sample data, and ask an oracle to evaluate this data against some criteria, what would be a good way to select where to sample next?

The learning algorithm has a number of possible strategies to use to optimize its performance. It can draw samples from the least performing class, cases where the margin between classes is minimal, or rely on entropy to obtain a diverse set [12, 131]. The different query strategies can be optimized to minimize multiple metrics, such as expected error reduction — attempting to reduce the generalization error, model change — maximizing the model parameter shift, uncertainty/variance reduction — sample that has the highest uncertainty or would decrease the variance of the model, or through selecting diverse sub-spaces [39, 40]. We will revisit this concept that not all data is created the same, in Chapter 5. There we illustrate that if the learner takes an active approach to the data collection regime, we can improve the overall performance of the system.

This process of active learning has also been applied for batched CNN networks, where the above heuristic does not perform optimally [129, 133], with the key insight being that a subset of points needs to be evaluated together, rather on a point per point basis. This idea of reformulating the heuristic on which is the next sample can also be viewed as an Deep Reinforcement Learning (DRL) - [47] or even as a meta-learning problem [36], where we would rely on a second level optimization process to learn this heuristic.

Another option in the generative domain is using a Generative Adversarial Network (GAN) network to synthesise training data for the active learning algorithm [158], which differs from the usual scenario of using the existing pool of unlabelled data.

2.4.3 *Data Augmentation and Synthetic Data*

All of these strategies so far have the aim of expanding the dataset such that it supports the biggest sub-domain from the observable distribution.

Another heuristic-driven approach would be data-augmentation. This is the process of augmenting the input data with a known function set to produce a sample that has reasonable expectations to come from the observed distribution. In the image case, this can be simple manipulations that vary the brightness, contrast, blurriness that have been shown to improve the performance and generalization of the learning algorithm [153]. These manipulations can be applied equally efficiently in the unsupervised case as regularizers [149].

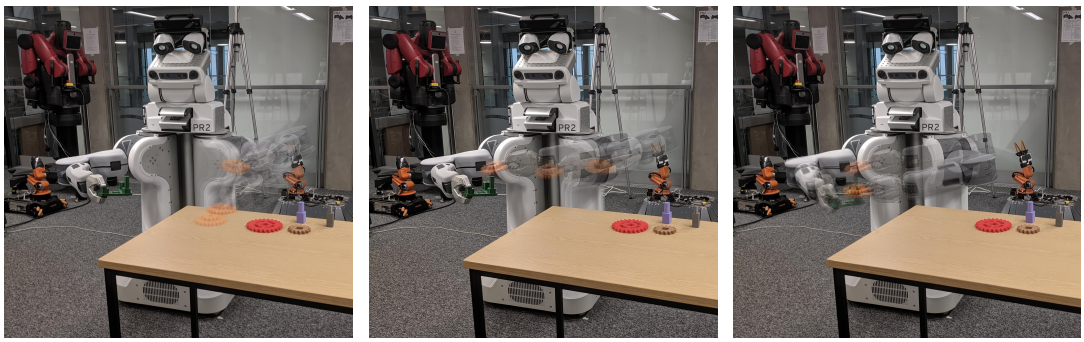
Finally, in cases where producing data is limiting, or the process can be virtually modeled, synthetic data can be produced that can exhaustively cover the space of variations. This can be seen in safety-critical applications, where failures have a high cost, or in situations with a long-tailed distribution, where having a passive observational approach has limited ability to generate the required diversity.

These types of data approaches are standard for classification or regression-based tasks. However, what would the alternative be for robot control tasks? How many demonstrations/samples of a task are enough? How do we diversify them such that the constructed task latent space matches the natural task distribution? These questions, alongside how to create informative demonstrations for the robot learning agent, will be discussed in Chapters 4 and 5.

3

Sequentially Composing Diverse Policies

In the introductory Tesla wiring harness example, we saw that solving long-horizon tasks requires having a set of tuned controllers for the different aspects of the task (e.g. aligning studs, placing spacers and bolts). Hierarchical motion planning can rely on the discontinuous switches between different local dynamics to build approximate models and to facilitate the design of local, region-specific controllers. However, it becomes combinatorially challenging to implement such a pipeline for complex temporally extended tasks, especially when the sub-controllers work on different information streams, time scales, and action spaces. In this chapter, we introduce a method that can automatically compose diverse policies comprising motion planning trajectories, dynamic motion primitives, and neural network controllers. We introduce a global *Goal Scoring Estimator* that uses local, per-motion primitive dynamics models and corresponding activation state-space sets to sequence diverse policies in a locally optimal fashion. We use expert demonstrations to convert what is typically viewed as a gradient-based learning process into a planning process without explicitly specifying pre- and post-conditions. We first illustrate the proposed framework using an MDP benchmark to showcase robustness to action and model dynamics mismatch, and then with a particularly complex physical gear assembly task, solved on a PR2 robot. We show that the proposed approach successfully discovers the optimal sequence of controllers and solves both tasks efficiently.



(a) Gear Pick Up (b) Move Gear (c) Gear Insertion

Figure 13: Robot setup for the gear assembly task. The robot needs to pick up a gear by leveraging the surface of the table, slide it up to an edge, grasp and move it in a collision free manner to the other hand, before inserting the gear onto the base plate.

3.1 Introduction

For robots to work in the wild, they need to be able to perform a variety of consecutive tasks that might require vastly different skills. Each individual skill could be partitioned and optimized outside of this complex system and is potentially constructed using several diverse methods, control strategies or sensor domains, such as motion planning approaches for reaching, contact-aware grasping, picking and placing, or through the use of end-to-end neural network-based controllers.

In many practical applications, we wish to combine a diversity of such controllers to solve complex tasks. This typically requires that controllers share a common domain representation and a notion of progress to sequence these. For instance, the problem of assembly, as shown in Figure 13, can be partitioned by first picking up a mechanical part, then using motion planning and trajectory control to move this in close proximity to an assembly, before the subsequent use of a variety of wiggle policies to fit the parts together, as shown by [86]. Alternatively, the policy could be trained in an end-to-end fashion with a neural network, but one may find this difficult for extended tasks with sparse rewards, such as in Figure 13. In the interest of sample efficiency and tractability, such end-to-end learning could be warm-started by using samples from a motion planner, which provides information on how to bring the two pieces together and concentrates effort on learning an alignment policy, as in [144]. Additionally, the completion of these independent sub-tasks can be viewed as a global metric of progress.

We propose a hybrid hierarchical control strategy that allows for the use of diverse sets of sub-controllers, consisting of commonly used goal-directed motion planning techniques, other strategies such as wiggle, slide, and push-against [103] that are so elegantly used in human manipulation, as well as deep neural network-based policies that are represented very differently from their sampling-based motion planning counterparts.

Thus, we tackle a key challenge associated with existing motion primitive scheduling approaches, which typically assume that a common representation is used by all sub-controllers. We make use of the fact that controllers tend to have a dynamic model of the active part of their state space — either an analytical or a learned model, and further estimate how close each state is to complete the overall task using a novel Goal Scoring Estimator. This allows the hierarchical controller to model the outcome of using any of the available sub-controllers and then determine which of these would bring the world state closest to achieving the desired solution — in the spirit of model predictive control.

As in the work of [26] on sequencing funnels and [142] on LQR-Trees, the scheduled controllers for sub-regions of the state space can be optimized in our framework, allowing for compositional task completion, but importantly, also for additional diversity of the controller set.

Value function approximation techniques used in the reinforcement learning community [83] can be considered similar to the proposed progress estimator, but only model the expected reward and require the actions to be in the same state space. We attempt to remedy this oversight, by allowing for a diversity of action and state spaces, and by modeling global progress at a local controller level.

This chapter makes the following contributions:

- We use a **Goal Score Estimator** to sequence a set of policies to solve a task. This estimator is trained using expert demonstrations to evaluate the current and future state of the plan and helps to transform the hierarchical learning problem into a **planning** problem.

- We provide a method for composing **diverse** policies that work with different input information, or decompose the action in either joint or end-effector space and work at different operational frequencies to solve a high-level task.

We first evaluate the use of the controller dynamics and the goal metric to compose policies in a hybrid controller on an MDP benchmark problem to evaluate robustness to action and model dynamics noise. Next, we apply this approach to a physical gear assembly task performed by the PR2 robot, making use of both motion planning and visual neural network policies (Figure. 13).

3.2 Related Work

Robotics

Compositionality is a key paradigm for robot control, which methods of composing controllers of a single type like [25, 26, 142] aim to exploit. These techniques rely on partitioning a state space into smaller overlapping operating regions and tuning sub-controllers (feedback or LQR) for operation in these regions. Unfortunately, these methods often fail to consider the fact that different tasks may require different controller sequences, and the scheduling of control laws in work on compositionality is often underemphasized. Inspired by this capability and the *funnels* framework [102][1] this work provides a MPC [53] framework for compositional sequencing where controllers can be of different types and operate using different state spaces.

The ability to act on different state-spaces and action sets is particularly important, as the sub-policies required to complete a temporally extended task can be highly variable. For example, sub-problems such as grasping and pushing have been addressed and investigated at least since the 1980s, and these could be encapsulated into operation as motion primitives [103]. Using a diverse set of policies allows for the selection of controllers that best fit the working domain - for example [98] highlights that compliance may be needed when movement and sensing reach the perception noise boundary, [101] advocate using non-prehensile grasps for manipulation of objects and [119] explore manipulation strategies that allow for caging of objects, such that these can be re-grasped stably in a subsequent stage. Alternatively these motion

1 Regions of robustness arising from the dynamics and control applied in a sub-region of the control space.

planning strategies can be formulated using stable nonlinear attractor systems as in DMPs [76, 127] or as DeepDMPs [54]. We aim to create a hybrid control framework that allows the use of these diverse motion planning controllers, alongside neural network policies to solve long-sequence tasks.

Learning from Demonstration

To expedite the learning process, it is common to provide demonstrated example solution trajectories to a problem. Methods like Behaviour Cloning (BC) allow for simple visuomotor policies to be learned end-to-end [18], or to be extended to learn safe policies [74], extract preferences [6] or to learn mappings for the perception and kinematic differences [9]. Alternatively, they can be used to calculate the relative value of each state through inverse reinforcement learning and to create a hierarchical formulation for control [89]. As explained in [35], there are limitations to BC in terms of the number of demonstrations, generalization, and the challenge of modeling complex scenarios as discussed in Chapter 2. However, we use these full task demonstrations as a means for estimating the distance to the desired goal state, which is arguably a simpler task than learning an entire policy. Additionally, by allowing different controller representations, we do not need to re-represent one control law in alternative approximate forms.

Reinforcement Learning

In the RL literature, the concept of options has parallels to our work, as each policy can be viewed as a controller with the initiation set as its domain. Our method lies between learning policies over options as in [13], and computing solutions using learning from demonstration by inverse reinforcement learning [10].

The options framework [117, 139] provides a formal means to work with hierarchically structured sequences of decisions made by a set of RL controllers. Temporal abstractions have been extensively investigated [45, 48, 75, 90, 139], and it is clear that hierarchical structure helps to simplify control, allows an observer to disambiguate the different states of the agent, and encapsulates a control policy and termination of the policy within a subset of the state space of the problem. This split in the state space allows us to verify the individual controller within the domain of operation [2, 56], deliberate about the cost of an option and increases interpretability [64]. Our work

can be viewed as using a planner as a hierarchical policy in the options framework, which is made possible through the incorporation of a goal-scoring progress function learned from demonstration.

Similarly, [63] showed how planning can be incorporated into action selection when future states can be evaluated. Our method borrows this view of temporally abstracting trajectories and extends it by applying a dynamics model for each of the options, allowing an agent to assess its states and incorporate foresight [3] in its actions.

The work of [104] highlights that including a dense reward indeed increases the overall performance of the agent. Instead of using a predetermined dense function, we learn a Goal Scoring estimator from the demonstrations. As shown in [144] naively tuning and shaping a reward function may result in sub-optimal solutions using base actions. Furthermore, our planner selects an already learned controller and thus avoids converging to sub-optimal behaviors.

As highlighted in [138], there are limits to the use of RL in robotics. By leveraging strategies from both RL and control communities, this work aims to increase the scope of problems that can be tackled in robotics.

3.3 Method

Our framework defines a hierarchical controller over the set of pre-existing controllers. Each policy uses its dynamic model to propagate the current state to a future state conditioned on its control law. The Goal Scoring Estimator, learned over expert demonstrations, evaluates those future states and selects a controller that brings the system closest to the desired configuration.

Formally, assume the existence of a learned set of controllers $\mathcal{C} = \{c_1, c_2, ..., c_N\}$ including those learnt from experience in previously solved problems. Using notation similar to the RL options framework [139], each controller c_ω is independently defined by a control law $\pi_\omega(s) \rightarrow a$, $s \in \mathcal{S}_\omega$, action $a \in \mathcal{A}_\omega$, a working domain $\mathcal{I}_\omega, \mathcal{I}_\omega \subseteq \mathcal{S}_\omega$ where the controller can be started, and a termination criterion β_ω. We rely on a forward dynamics model $s_{t+1} \sim \mathcal{D}_\omega(s_t, a_t)$, which is a stochastic mapping, and a

learned per-task Goal Scoring metric $g \sim \mathcal{G}_{K_j}(s_t)$, $0 \leq g \leq 1$, that estimates the progress of the state s_t with respect to a desired world configuration. We assume $\mathcal{G}_{K_j}$ to change monotonically through the demonstrated trajectories. The different controllers can work on different state spaces $\mathcal{S} = \{\mathcal{S}_1, \mathcal{S}_2, .., \mathcal{S}_N\}$ as long as there exists a space $\mathcal{S}^*$, such that $\mathcal{S}_i \subseteq \mathcal{S}^*$. This means there exists a higher or equal order state space, which maps the controller space of operation to regions of $\mathcal{S}^*$.

This work constructs a hybrid hierarchical controller $\pi_\Omega(\omega_t|s_t)$ that can choose the next controller c_{ω_t} that needs to be executed to bring a learned latent state s_t to some desired s_{final}. It uses the forward dynamics model $\mathcal{D}_\omega$ in an n-step MPC look-ahead, using a Goal Scoring metric $\mathcal{G}_K$ that evaluates how close s_{t+n} is to s_{final}, without taking into account the cost of transitioning in-between.

As shown in Figure 14, in this work, we use a VAE to learn a latent state s_t from image observations. We assume that each controller in the library has an associated forward dynamics model, trained to predict the next latent state, s_{t+1}. This provides us with an implicit mapping between states and allows us to render an image of an expected scene for each controller that is applied. This scene prediction is then used by the goal score metric to evaluate the effect of choosing each controller and to select the most appropriate controller to be used at a given time step. In effect, this means that controllers act on the appropriate state components, but the underlying state representation used for controller selection is conditioned on image observations. Conditioning on images is feasible, as the robot head camera provides an overhead view of the entire workspace. While it may be possible to learn a shared state representation or mapping between states, this can be challenging (e.g. mapping from joint angles to images is extremely hard), while learning to predict the next latent state is a much easier task. Each of the framework components is described below.

3.3.1 *Goal Score Evaluation*

The key component of the proposed framework is the ability to evaluate how well a particular state s maps to parts of a demonstrated expert trajectory. This allows us to estimate the temporal distance of that state to the end of the demonstration (see Figure 14). In a similar manner to [130], who use adjacency of frames as positive

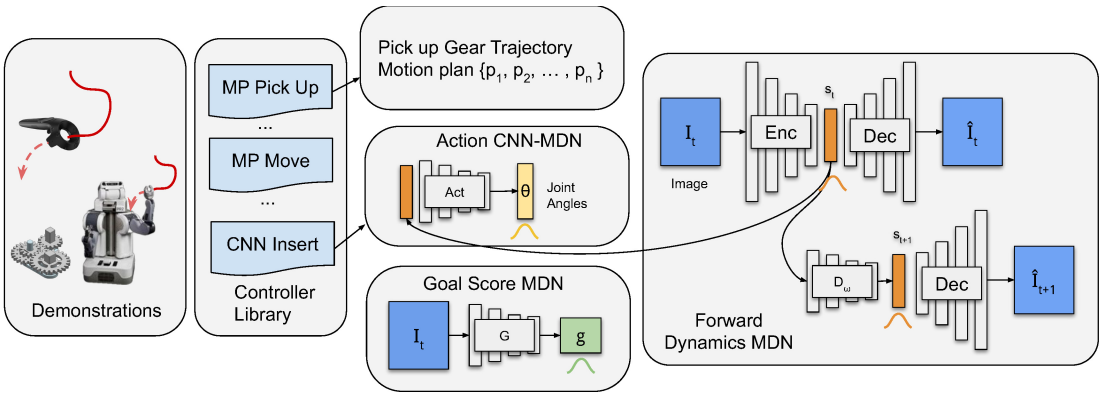


Figure 14: Demonstrations were performed by using an HTC Vive controller that directly teleoperates the end-effector of the PR2 robot at 20 Hz. In the Gear Assembly Task, our controller library includes motion planning (MP) primitives (operating on joint angles) for picking up or moving and a convolutional neural network (CNN) for inserting the gear (operating on images). The MP primitives produce a trajectory for executing a task. The CNN policy takes a latent representation of the image state and generates a distribution over the target joint angles of the robot. The Forward Dynamics models use a VAE representation alongside an s_{t+1} dynamics prediction network that uses the same decoder. The Goal Score Estimator network takes in an image and produces a distribution over how well this image maps to a particular point in the demonstrations.

and negative examples, we leverage the temporal sequence of the demonstration as a measure of task completeness.

We capture demonstrations of the global task (in its entirety) to use as weak supervision for learning a Goal Scoring Estimator network that allows us to map a state to a progress estimation value $g \sim \mathcal{G}(s_t)$ for a given task. To build the Goal Scoring models, we use a convolutional network head with a MDN tail to encode the different goal representations based on image observations. The network predicts a distribution over the proximity of the current state to the desired goal state.

The first observation of a demonstration can be viewed as score 0 – far away from the goal state, whereas the final observation as score 1.0 – a target representation of the world. Even though there may not be a one-to-one mapping between the values within several demonstrations, we rely on the variability in their lengths being encoded within the different modes of the MDN of the Goal Scoring Model.

3.3.2 *Controller Selection*

At a particular point at state $s_t, s_t \in S^*$ when c_ω is active, we can compute the goodness of following the current controller given these conditions up to a particular time horizon. The action given by the policy is $a_t = \pi_\omega(\hat{s_t}), \hat{s} \in S_\omega$, and following the dynamics model we can write that:

$$s_{t+1} = \mathcal{D}_\omega(s_t, a_t) = \mathcal{D}_\omega(s_t, \pi_\omega(\hat{s_t})). \tag{8}$$

As the dynamics model is conditioned on the controller c_ω, we can simplify to $s_{t+1} = \mathcal{D}_\omega(s_t)$. Chaining this for n steps into the future we obtain

$$s_{t+n} = \mathcal{D}_\omega \circ \mathcal{D}_\omega \circ \cdots \circ \mathcal{D}_\omega(s_t) = \mathcal{D}_\omega^n(s_t). \tag{9}$$

We can evaluate this future state as $g_{t+n} = \mathcal{G} \circ \mathcal{D}_\omega^n(s_t)$. as $\mathcal{G}$ provides an expected utility of the terminal sate of the current process. Thus, the hierarchical controller over controllers can be sequentially optimized by maximizing the expected utility by an adaptation of the Bellman equation,

$$\pi_\Omega(\omega_t|s_t) = \arg\max_\omega \left(\mathbb{E}\left[\mathbf{1}_{\mathcal{I}_\omega}(s_t) \cdot \mathcal{G} \circ \mathcal{D}_\omega^n(s_t)\right]\right) \tag{10}$$

This chooses the controller that is within the operation domain for the current state and delivers the largest goal score estimate after n steps. After choosing and evaluating the optimal π_Ω with respect to the above criterion, another controller can be selected at the next time step, with repetition until the goal is reached.

3.3.3 *Controller Dynamics Modelling*

The dynamics of each controller is modeled individually only within its operational domain. This simplifies the complexity the dynamics model has to learn and thus requires less data. Here, we learn a neural dynamics model for each controller that predicts the latent state configuration s_{t+1} from s_t, as in [62]. The architecture, shown in Figure 14, is based on a VAE encoding but includes an additional dynamics

network, which predicts the next latent state if a given controller were applied. The same decoder is used to force the two representations not to diverge.

A diverse dynamics network can be used as a prior for each controller [46] and the execution of the controllers themselves can be used to build an individual model using the image state space if it is not provided internally.

3.4 Experimental Setup

We perform two sets of experiments to investigate the efficacy of the structured hierarchical policy by performing MPC future predictions at each step on a simulated MDP problem and on a much more complex physical gear assembly task on the PR2 robot.

3.4.1 *Simulated MDP*

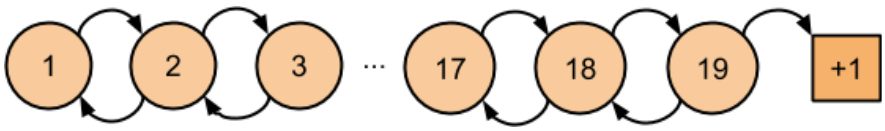


Figure 15: The 19-state MDP problem. The action space of the MDP is to move "left" or "right". The goal of the MDP problem is to reach past state 19 and obtain the +1 reward, which is equivalent to a termination state 20.

In the first experiment, we use the standard 19-state random walk task as defined in [66] and shown in Figure 15 to illustrate concepts in a simple sequential decision making task. The goal of the agent is to reach past the 19^{th} state and obtain the $+1$ reward. The action space of the agent is to go "left" or "right", moving the agent to an increasing or decreasing state. There also exist 5 controllers defined as in Section 3.3, with the following policies: (1-3) policies that go "right" with a different termination probabilities $\beta = \{0.9, 0.5, 0.2\}$; (4) random action; (5) policy with action to go "'left" with $\beta = 0.5$. We assume that there exists a noisy dynamics model $\mathcal{D}_\omega$ and the goal evaluation model $\mathcal{G}_{MDP}$, which has the probability of falsely predicting the current state or its value of 0.2.

Further, we expand the MDP to be of size 100 and evaluate how sensitive the performance of the model is in regards to noise in the Goal Scoring Evaluator and in each of the dynamics models.

3.4.2 *Gear Assembly*

In this task, the PR2 robot needs to assemble the first part of the Siemens Challenge[2], which involves grasping a compound gear from a table, and placing it on a peg module held in the other hand of the robot. We record expert demonstrations of the task being performed and assume access to a set of controllers that (1) picks up the gear from the table; (2) moves the left PR2 arm in proximity to the other arm; (3) inserts the gear on the peg module.

The agent receives the robot's joint state space, as well as images from the head camera and are used as needed by the corresponding controllers. In the whole sequence during demonstration, the data is collected with the aim of learning a Goal Scoring Metric from the images. It aims to use the GSM to solve the demonstrated task with the pre-existing set of skill library.

Policy (1, 2) rely exclusively on scripted path planning techniques and work using discrete time steps, while (3) is learned entirely with a neural network. Controllers (1, 2) share a common state space of the robot's joint angles, whereas (3) works directly on the visual pixel input from the robot's head camera.

The visual neural policy, shown in Figure 14, performs imitation learning by using behavior cloning of the 50 teleoperated demonstrations. This is trained until convergence or 100 epochs using different encoder heads - small convolutional network, ResNet-50, -101. The expert-illustrated trajectories were performed using an HTC Vive controller teleoperating the PR2 robot and the process took less than 1h wall time. We use a BC loss, augmented with the VAE loss to train our policy models, optimizer was Adam with $\alpha = 0.001$ and weight decay of $1e^{-6}$. The action generation part of the network is an MDN that predicts a distribution of the next time step joint angles θ, which are set as the internal PID targets for the robot 7-DOF arm.

2 The challenge is at https://new.siemens.com/us/en/company/fairs-event

The dynamics model for each controller is learned independently and is represented with a Forward Dynamics MDP, learned from forward rollouts of the policy network. The Goal Score estimator is learned on an additional 5 rollouts of the full gear assembly task and operates on the latent space of the particular policy. Throughout all of the experiments we use the Adam optimizer with a weight decay rate of $1e^{-}6$, batch size of 120, train for 200 epoch and the MDN uses 24 Gaussian mixtures. We show the performance of this model with several video streams from different cameras on the robot (head, left, and right forearm cameras).

Additionally, we compare the performance of the scripted Motion Planning method (using RRT Connect [92]), Dynamic Motion Primitives (learned from the MPs), and the Visual Neural Policy on each subtask, as well as using the full sequence under the different controllers as a baseline.

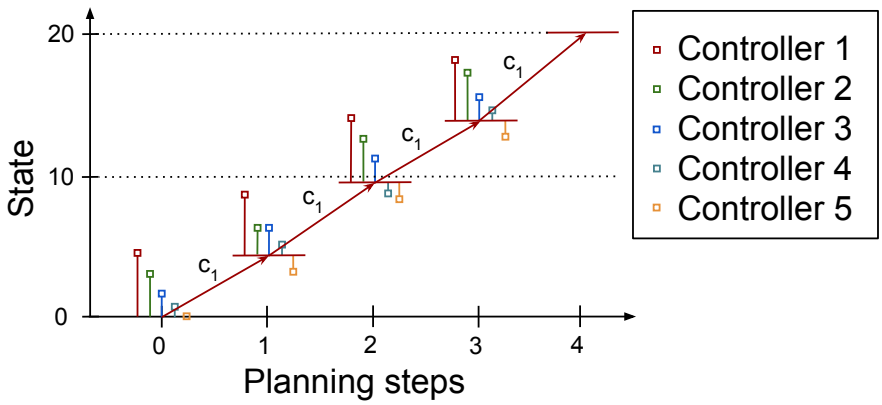


Figure 16: MDP solution. At timestep 0, a rollout of the 5 controllers is performed with the dynamics model. The expected resulting state is marked using vertical bars. The best performing controller is used within the environment to obtain the next state - the red line at state 5 and planning step 1. This process is iterated until a desired state is reached.

3.5 Experimental Results

We demonstrate the viability of composing diverse policies by using the controller dynamics as a method for choosing a satisfactory policy. The dynamics can be learned independently of the task and can be used to solve a downstream task.

3.5.1 *Simulated MDP*

This problem illustrates the feasibility of using our architecture as a planning method. Figure 16 shows that the agent reaches the optimal state in just 4 planning steps, where each planning step is a rollout of a controller. The predicted state under the specified time horizon is illustrated at each step for the different controller options. This naturally suggests the use of the policy π_1 that outperforms the alternatives (π_1 reaches state 6, π_2 - state 4, π_2 - state 3, π_3 - state 1, π_4 - state 1, π_5 - state 0). Even though the predicted state differs from the true rollout of the policy, it allows the hierarchical controller to use the controller that would progress the state the furthest. The execution of some controllers (i.e. c_5 in planning steps 1, 2, 3) reverts the state of the world to a less desirable one. By using the forward dynamics, we can avoid sampling these undesirable controllers.

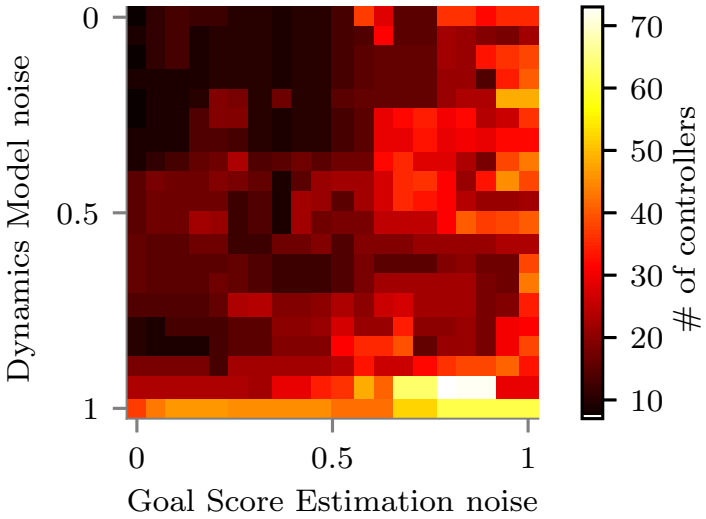
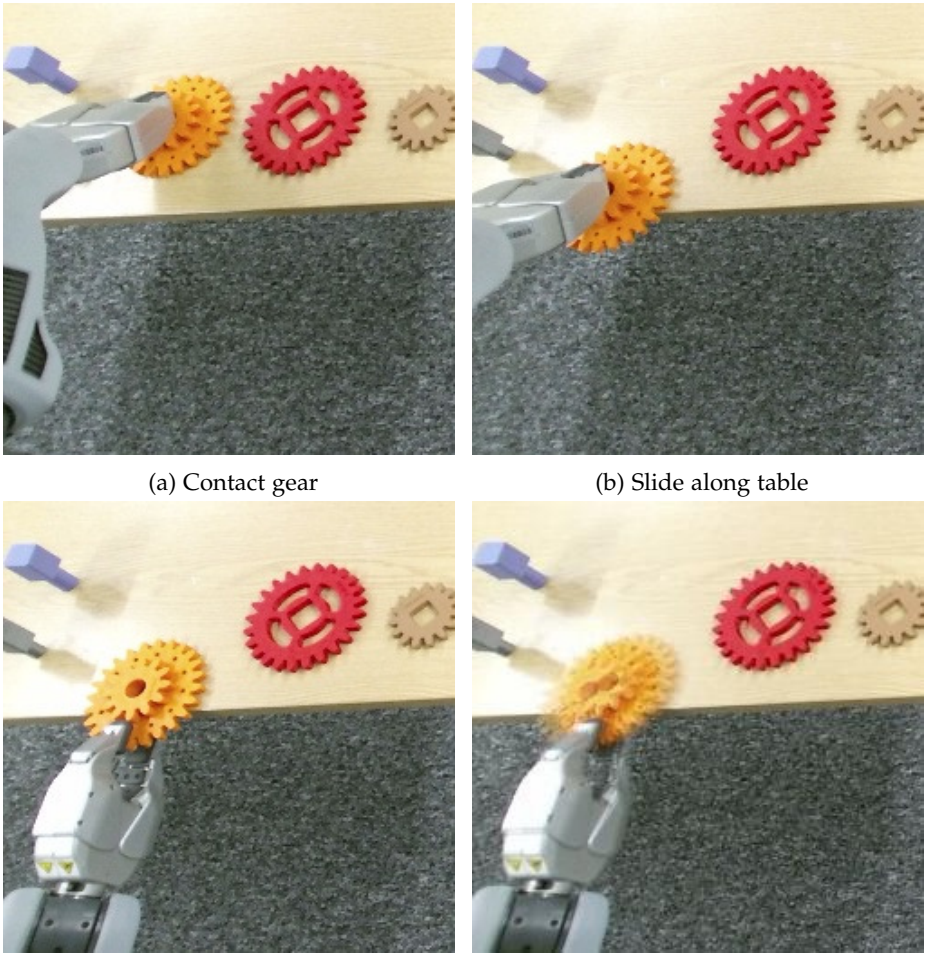


Figure 17: Sensitivity to noise in the dynamics model and the Goal Score Estimator for a world of size 100. The heatmap illustrates the number of controllers that were used in order to reach the target with a lower number - top left - being optimal. The number of controllers varies between the optimal 8 and 72.

To investigate the robustness and convergence properties of our method, we introduce noise within the system, while expanding the MDP to be of size 100 and maintaining the same 5 controllers as above. We can see in Figure. 17 how the number of controllers required to reach the target location varies at different noise levels. When we observe low amounts of noise, the performance remains stable and requires activating any of these controllers a total of fewer than 20 times (top-left part of the heatmap). The expected optimal number of controller activations based on policy 1



(a) Contact gear (b) Slide along table

(c) Re-grasp stably (d) Grasp variability

Figure 18: Images **(a-c)** illustrate key frames of the pick up policy that involves making physical contact with the gear, sliding it along the table surface to an edge and grasping it firmly in the new position. **(d)** A visual overlay of 3 random pickup attempts. The difference in grasp position relative to the gear is comparable to the inner diameter and is a byproduct of the stochasticity in the sliding and grasping action. This does not hinder the performance of the CNN in the full task.

is 12 (black region of the heatmap). As the noise in both the dynamics model and the Goal Score Estimation increases, we observe a degradation and the selection of more sub-optimal controllers. The model is more sensitive to noise in the Goal Score Estimator than when the dynamics of the controllers make errors in their predictions. Despite this, the method converges to the optimal state.

It is interesting to note that the method uses close to or the optimal number of controller activations in cases where multiple policies would drive the world in a progressive state, highlighting that the goal score metric is capable of choosing longer horizon controllers due to the MPC look ahead.

3.5.2 *Gear Assembly*

We build the library of controllers for the task - picking up a gear (Figure 18), moving it close to the base of the assembly, and inserting the gear on the base plate (Figure. 19). A motion planning control method was used to perform different tasks. Those demonstrations were used to build the DMP model, using the ROS-DMP module, which is based on [76]. The CNN policy was trained using 50 teleoperated demonstrations [3] covering a wide variety of initialization cases for each specific task. We did not observe any task performance changes between the small Convolutional or the ResNet-50,-101 head, and therefore relied on the simple architecture as seen in Table 1. Other tasks may benefit from deeper or more complex models (such as [147, 155, 156]), but integration within the method would remain the same.

Table 1: Performance of the control policy with different structure of the NN head. We evalute on the Gear Insert task and how it would impact the full task performance. We can select the small network, as the performance remains the same with lower inference and training cost.

	Gear Insert	Full Task
ResNet-50	10/10	10/10
ResNet-101	10/10	10/10
Small Conv	10/10	10/10

The input image has a size of 128x128 pixels. We observe that the NN policy does not require a sophisticated feature extractor to create necess50ary features for the task. The final network used 5 convolutional layers of 4x4 filters, stride 1, with batch normalization and leaky ReLU activations.

Table 2 shows the performance of the different controllers on different tasks. The MP and DMP models exhibit stable performance in contact-based tasks, but fail where the initial conditions differ — in Figure 18 we can see the variability that the pickup controller exhibits in terms of the location of the grasp on the gear, which leads to failures in attempting to insert this onto the base assembly. The issue comes from the tolerances of the fit as using an MP and a sequence of trajectory points does not compensate for any inaccuracies incurred during the previous stages of the process

3 Interestingly, additional "what-if" *(it's tilted, flipped, not visible, etc.)* singular training examples were detrimental to the performance of the model. In order to incorporate that part of the state space, a full set of overlapping and interpolating examples need to be provided. This is also supported by the data discussion in Chapter 2.

Table 2: Table of successful trials for different policies. MP - Motion Planning, DMP - Dynamic Motion Primitive, CNN - Convolutional Neural Network. The CNN policy has a maximum of 50 steps to reach the goal. The symbol '*' indicates policies terminated early due to safety concerns of the shielding policy.

Control Method	Pick Up	Gear Move	Gear Insert	Full Task
MP	10/10	10/10	1/10	1/10
DMP	10/10	10/10	1/10	1/10
CNN	*	10/10	10/10	*
MP & CNN (Our)	10/10	10/10	10/10	**10/10**

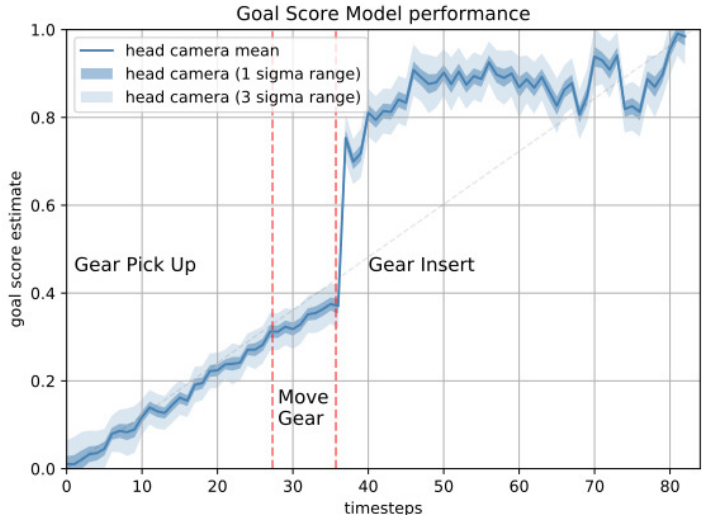
or manual positioning. Precise insertion is known to fail outside of a very small convergence basin when using MP controllers - we obtain similar (bad) performance similar to [100, 144].

As a baseline, we compare against optimally sequencing the MP and DMP control strategies, which can be seen under the "Full Task" performance. Due to the low performance on a part of the task, the overall success rate is limited.

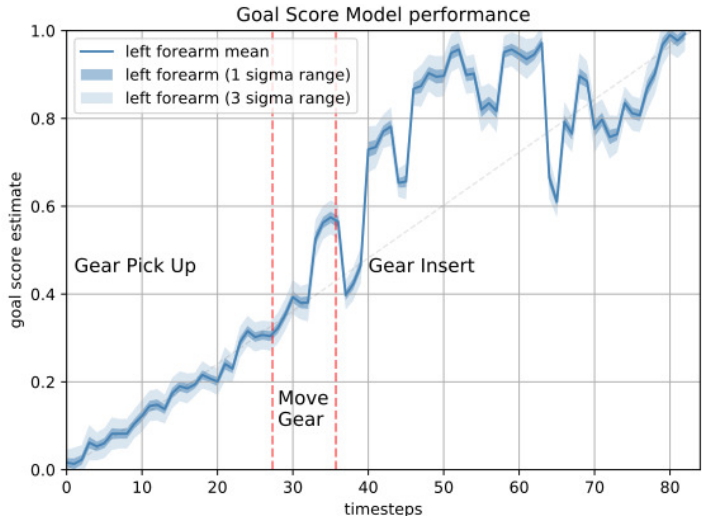
Figure 19: The execution of a neural network policy for inserting the gear on the peg.

In contrast, the natural variability of the grasp is part of the training set of the CNN model and successfully inserts the gear even with a high variance of initial locations (Figure. 19). As the visual CNN policy is not dependant on the absolute position of either the grasped location or the position of the base assembly, it performs corrective/feedback actions for the policy to succeed. However, the CNN performance on the pickup task could not be evaluated, as the prescribed controller actions were jerky and violated safety constraints (pre-defined velocity and position limits).

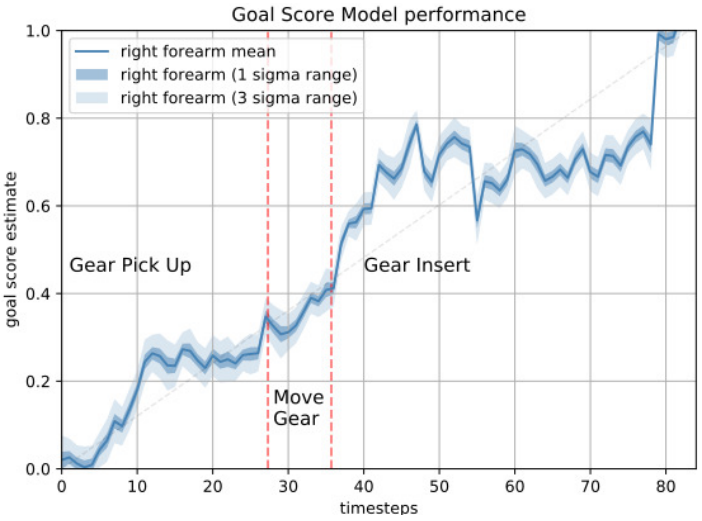
This illustrates that the combination of **MP** for picking up the gear and moving it closer to the assembly and **CNN** to insert the gear, selected using our method allows for the full task to be successfully solved optimally 10 out of the 10 attempts. This



(a) Head Camera



(b) Left Forearm



(c) Right Forearm

Figure 20: The goal score metric calculated during the execution of a random trial. During the first two motion planning controllers, the model is monotonically increasing the goal metric. The stochasticity of the neural network policy leads to oscillating scores. Using different input streams, the prediction accuracy could be altered — the scene head camera does not see the fine details of the movement which the forearm cameras do, leading to a closer to goal score. The peaks in the forearm cameras are associated with states where the peg is extremely close to the gear hole, highlighting that proximity. Example snapshots from different views can be seen in Figure 21.

(a) Head Camera (b) Left Forearm (c) Right Forearm

Figure 21: Snapshots of the input from different cameras on the PR2 robot demonstrate a moment, where the head camera cannot differentiate how well the task is performed, the left camera is optimistic from its perspective, while the right accurately evaluates the performance as sub-optimal, leading to the goal scoring network predicting a decreased value.

shows the advantage of using a diverse set of controllers, allowing each one to be tuned to the domain of operation.

The Goal Score Estimator model is trained on only 5 full task demonstrations. We empirically choose $n = 10$ for the n step MPC look ahead as our planning horizon. Figure 20 illustrates the Goal Score estimation for a previously unseen demonstration from camera streams with different viewpoints. The score for the different controllers can clearly be used to sequence the policies. This is shown by the fact that the score follows a monotonically increasing value with regards to the average score for the individual controller domain.

3.6 Limitations of Sequentially Composing Policies

There are a few assumptions in regards to the ability to sequentially compose the policies. On a fundamental level, there is a presumption about the existence of a policy library with attached dynamics models. The problem is framed as the ability to rapidly redeploy the system by demonstrating an alternative configuration that can be formulated through this set of known policies. Especially for learned dynamics, there is a known limitation where for the predictions to follow the expected latent trajectory, the seed location needs to be within the domain of operation of the learned policy.

In the definition of the hierarchical policy, a greedy n-step MDP is chosen to propagate each policy and select the best performing one. This is sufficient for the demonstrated temporally extended tasks, but is susceptible to the same issues as

potential fields by finding local minima. The current architecture as defined by Eq.10 doesn't have the capabilities of re-planning beyond the natural state localisation within the different policies. Additionally, the planning is not taking into account any adjacent constraints that can be imposed such as temporal or spatial constraints. For instance, if multiple policies can transition from a seed state to a target state, the one taking less time will be preferred, regardless of any additional constraints like distance to humans or fragile objects. This can be potentially alleviated by altering the selection function by prioritizing externally defined measures. More examples of altering a policy conditioned on specifications is examined in Chapter 4.

Additionally, the greedy selection process works on a static world representation. In essence, the internal MDP is stateless. This leads to the process lacking the ability for the top-level hierarchical function to execute repetitive tasks. A representative task that illustrates this failure is playing the piano — the visual state representation (e.g. which keys are pressed) is not informative about the current state of the MDP or the world (a melody has multiple repeating sequences). More advanced methods for joint policy optimization and state construction can be found in Chapter 5.

3.7 Conclusion

We introduce a method for composing diverse policies with varied representations, including Motion Planning, Dynamic Motion Primitives, and Convolutional Neural Networks. This allows for the solution of combinatorially complex and temporally extended tasks requiring multiple steps, without needing to pre-define controller sequences or design high-level state machines. We sequence tasks by using a Goal Scoring Model trained by expert demonstrations providing a weak supervisory signal. The Goal Scoring Estimator model provides a controller invariant prediction of progress towards a goal, which can be used with shared latent space across sub-controllers. This work has also introduced different methods that allow for a model-based or a model-free way to create a dynamics model, which can be used to analytically plan the next best option within a model predictive control framework.

But what is the best strategy to build up the controller library? One of the limitations of this work is that we assumed a parallel task to provide the needed

diversity, but we are also interested in building it. In Chapter 5 we take a step towards being able to intelligently partition tasks into sub-problems and in Chapter 4 to alter already already existing policies and decompose them into rules around known symbols.

4

Causal Analysis on Policy Structure

In this chapter we focus on eliciting structure within the policy — what does it (and the human demonstrator) pay attention to, can we alter this salience by focusing on alternative demonstrations, or parameterize this difference?

Learning models of user behavior is an important problem that is broadly applicable across many application domains requiring human-robot interaction. This chapter shows that it is possible to learn generative models for distinct user behavioral types, extracted from human demonstrations, by enforcing the clustering of preferred task solutions within the latent space of a high capacity neural model. We use these models to differentiate between user types and to find cases with overlapping solutions. Moreover, we can alter an initially guessed solution to satisfy the preferences that constitute a particular user type by backpropagating through the learned differentiable models. An advantage of structuring generative models in this way is that we can extract causal relationships between symbols that might form part of the user's specification of the task, as manifested in the demonstrations. We further parameterize these specifications through constraint optimization in order to find a safety envelope under which motion planning can be performed. We show that the proposed method is capable of correctly distinguishing between three user types, who differ in degrees of cautiousness in their motion while performing the task of moving objects with a kinesthetically driven robot in a tabletop environment.

4.1 Introduction

As we move from robots dedicated to a restricted set of pre-programmed tasks to being capable of more general-purpose behavior, there is a need for easy re-programmability of these robots. A promising approach to such easy re-programming is Learning from Demonstration, i.e., by enabling the robot to learn from and reproduce behaviors shown to it by a human expert — Figure 22.

This paradigm lets us get away from having to handcraft rules and allows the robot to learn by itself, including modeling the specifications the teacher might have used during the demonstration. Often such innate preferences are not explicitly articulated, typically being in the form of biases resulting from experience with other potentially unrelated tasks sharing parallel environmental corpora — Figure 23.1. The ability to notice, understand, and reason causally about these 'deviations', whilst still learning to perform the demonstrated task is of significant interest.

Similarly, other methods for Learning from Demonstration as discussed by [9] and [152] in the Reinforcement Learning domain are focused on finding a general mapping from observed state to action, thus modeling the system or attempting to capture the high-level user intentions within a plan. The resulting policies are not generally used as *generative models*. As highlighted by [138] one of the fundamental challenges with robotics is the ability to *reason* about the environment, beyond a state-action mapping.

Thus, when receiving a positive demonstration, we should aim to understand the causal reasons differentiating it from a non-preferential one, rather than merely mimicking the particular trajectory. When people demonstrate a movement associated with a concept, they rarely mean to refer to one singleton trajectory alone. Instead, that instance is typically an element of a set of trajectories sharing particular features. So, we want to find groups of trajectories with similar characteristics that may be represented as clusters in a suitable space. We are interested in learning these clusters so that subsequent new trajectories can be classified according to whether they are good representatives of the class of intended feasible behaviors. Further, we want to distill these specifications into a set of parameterized rules and find a safety envelope that can represent the learned model. For instance, one such rule may be *"The robot*

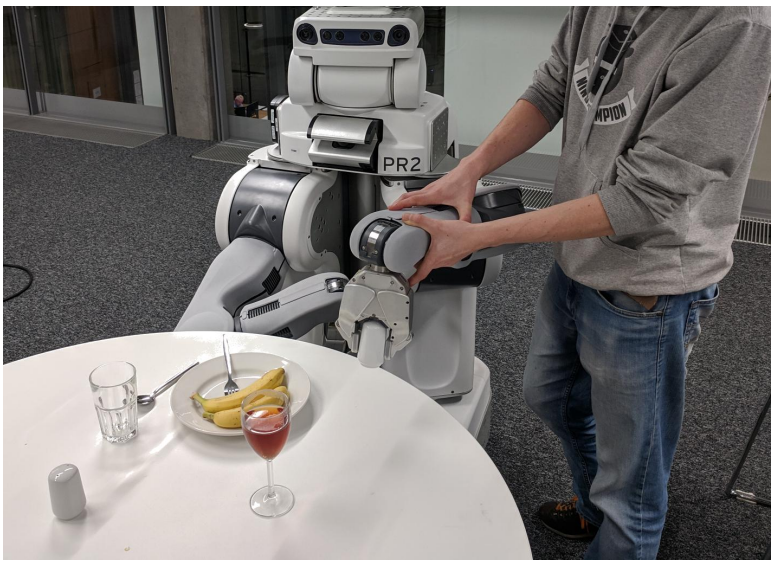


Figure 22: Example setup - the demonstrated task is to return the pepper shaker to its original location—next to the salt shaker. Deciding which objects to avoid when performing the task can be seen as conditioning on the user specifications, implicitly given during a demonstration phase.

should not get closer than T_{min} away from an object". These rules would generalize to unseen world configurations, as they are dependent on object characteristics.

It is often the case that in problems that exhibit great flexibility in possible solutions, different experts may generate solutions that are part of different clusters — Figure 23.2. In cases where we naively attempt to perform statistical analysis, we may end up collapsing to a single mode or merging the modes in a manner that doesn't entirely reflect the underlying semantics (e.g., averaging trajectories for going left/right around an object).

When we talk about task specification, we understand the high-level descriptions of a task-based trajectory and its desired behavior/interaction with a cluttered environment and its symbolic representation through causal analysis. For instance learning the manner, by which the robot end-effector may move above or around objects in the scene. The specifications, as learned by the network, are the observed regularities in human behavior. These rules are then parameterized by performing constrained optimization based on the demonstrations or samples from the learned model.

We present a method for introspecting in the latent space of a model which allows us to relax some of the assumptions illustrated above and more concretely to:

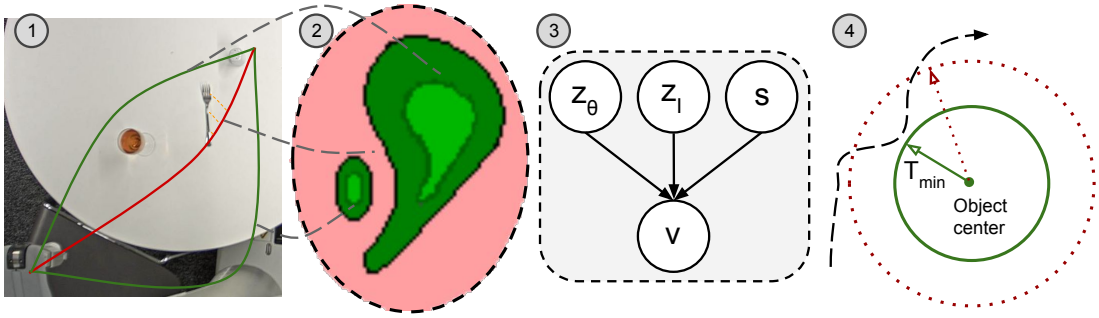


Figure 23: **(1)** Demonstrations that satisfy the user task specification maintain a distance from fragile objects (i.e. a wine glass), or fail to satisfy the specification by moving over sharp items. **(2)** An environment can have multiple clusters of valid trajectories in the latent space, conditioned on user type. **(3)** The validity of trajectories can be represented as a causal model. Whether a trajectory is part of a cluster v is conditioned on the specific path z_θ, the environment z_I, and the specification s. **(4)** The minimum radius from the object centre - T_{min}, which would change the validity of a trajectory.

- find varied solutions to a task by sampling a learned generative model, conditioned on a particular user specification.
- backpropagate through the model to change an initially guessed solution towards an optimal one with respect to the user specification of the task.
- counterfactually reason about the underlying feature preferences implicit in the demonstration, given key environmental features, and to build a causal model describing this.
- find a safety envelope of parameters to sets of rules representing the specifications through constraint optimization that allows their future use in motion planning.

4.2 Related Work

4.2.1 *Learning from Demonstration*

Learning from demonstration involves a variety of different methods for approximating the policy. In some related work, the state space is partitioned and the problem is viewed as one of classification. This allows for the environment state to be in direct control of the robot and to command its discrete actions - using Neural Networks [79], Bayesian Networks [77], or Gaussian Mixture Models [32]. Alternatively, it can be

used to classify the current step in a high-level plan [145] and execute predetermined low-level control.

In cases where a continuous action space is preferred, regressing from the observation space can be achieved by methods such as Locally Weighted Regression [34].

It has been long advocated that reasoning as part of planning is dependent on reasoning about objects, their geometric manifestations, and semantics [138]. This is based on the view that structure within the demonstration should be exploited to better ground symbols between modalities and to the plan.

One way to learn such latent structure can be in the form of a reward function obtained from IRL as described in Brown and Niekum [21], Ng, Russell et al. [109] and Zhifei and Meng Joo [157]. However, it is not always clear that the underlying true reward, in the sense of being *the unique reward* an expert may have used, is reconstructable or even if it can be sufficiently approximated. Combining multiple demonstrations to blend the desired expert response [148] may not recreate an expected output with divergent multi-clustered demonstrations, which we are interested in the current work. Alternatively, solutions that are based on composing smaller policies to mitigate the search for the hierarchical decomposition of the demonstration through direct learning of a goal-scoring metric [4] or through pair-wise ranking [57]. Preference-based reinforcement learning (PbRL) [152], offers methods whose focus is on learning from non-numeric rewards, directly from the guidance of a demonstrator. Such methods are particularly useful for problems in high-dimensional domains, e.g. robotics [73, 80, 81], where a concise numeric reward (unless highly shaped) might not be able to correctly capture the semantic subtleties and variations contained in the expert's demonstration. Thus, in the context of PbRL, the method we propose learns a user specification model using user-guided exploration and trajectory preferences as a feedback mechanism, using definitions from Wirth et al. [152].

4.2.2 *Causality and State Representation*

The variability of environmental factors makes it hard to build systems relying only on correlation data statistics for specifying their state space. Methods that rely

on causality [65, 113], and learning the cause and effect structure [120], are much better suited to supporting the reasoning capabilities required for transfer of core knowledge between situations. Interacting with the environment allows robots to perform manipulations that can convey new information to update the observational distribution or change their surrounding, and in effect perform interventions within the world. Counterfactual analysis helps in a multi-agent situation with assignment of credit [51]. It shows that marginalizing an agent's actions in a multi-agent environment through counterfactuals allows learning a better representative Q-function. In this work, we similarly employ a causal view of the world where we capture the expert preference in the model and evaluate it against a different set of environments, which is prohibitive if we used human subjects.

Learning sufficient state features is an open challenge for LfD Argall et al. [9]. The problem of learning disentangled representations aims at generating a good composition of the latent space, separating the different modes of variation within the data. Promising improvements in disentangling of the latent space with few a priori assumptions, by manipulating the Kullback - Leibler divergence loss of a variational auto-encoder [30, 69]. Denton and Birodkar [43] shows how the modes of variation for content and temporal structure should be separated and can be extracted to improve the quality of the next frame video prediction task if the temporal information is added as a learning constraint. While the disentangled representations may not directly correspond to the factors defining action choices, Johnson et al. [82] adds a factor graph and composes latent graphical models with neural network observation likelihoods.

The ability to manipulate the latent space and separate variability as well as obtain an explanation about behavior is also of interest to the interpretable machine learning field [44].

4.2.3 *Constrained Optimization*

The ability to find an optimal solution under a set of constraints has been well studied, e.g., in [15, 27]. [108] is one representative and state of the art method for propositional satisfiability (SAT). These methods have a history of being applied to

robotics problems for high-level planning, motion planning [55] and stability analysis [88].

In this chapter, we use these methods to efficiently navigate the search space whilst adhering to a set of non-linear constraints. With the development of increasingly more mature libraries for constrained optimization and SAT solving, such as [111], whose CP-SAT solver is based on [132], we can efficiently rewrite the set of specifications as parametrized channeling rules activated under different conditions, which partition the state space of the problem. As a result, we can optimize their respective parameters from the demonstrations.

4.3 Problem Formulation

In this work, we assume that the human expert and robotic agent share multiple static tabletop environments where both the expert and the agent can fully observe the world and can interact with an object being manipulated. The agent can extract RGB images of static scenes and can also be kinesthetically driven while a demonstration is performed. The task at hand is to move an object held by the agent from an initial position p_{init} to a final position p_f on the table, while abiding by certain user-specific constraints. Both p_{init} and $p_f \in \mathbb{R}^P$. The user constraints are determined by the demonstrator's type s, where $s \in S = \{s_1, \ldots, s_n\}$ for n user types.

Let $\mathcal{D} = \{\{\mathbf{x}_1, v_1\}, \ldots, \{\mathbf{x}_N, v_N\}\}$ be a set of N expert demonstrations, where $\mathbf{x}_i = \{I, tr_i^s\}$, $I \in \mathbb{R}^M$ is an RGB image of the tabletop scene, tr_i^s is the trajectory and v_i is a binary label denoting the validity of the trajectory with respect to the user type s. Each trajectory tr_i^s is a sequence of points $\{p_0, \ldots, p_{T_i}\}$, where $p_0 = p_{init}$ and $p_{T_i} = p_f$. The length of the sequences is not constrained—i.e. T is not necessarily the same for different trajectories.

The learning task is to project each $\mathbf{I} \in \mathbb{R}^M$ into $\mathbf{Z_I} \in \mathbb{R}^K$, by an encoder $Z_I = E(I)$, and $tr_i^s \in \mathbb{R}^{PT_i}$ into $\mathbf{Z} \in \mathbb{R}^L$, by Bèzier curve reparameterization, $Z_\theta = Bz(tr_i^s)$, with significantly reduced dimensionality $K \ll M$, $L \ll PT_i$. Both Z_I and Z_θ are used in order to predict the validity $\hat{v}_i, \hat{v}_i = C_s(Z_I, Z_\theta)$ of the trajectory tr_i^s with respect to the user type s. With an optimally-performing agent, $\hat{v}_i \equiv v_i$. For more details see Figure 24.

In order to alter an initial trajectory, we can find the partial derivative of the model with respect to the trajectory parameters with the model conditioned on a specific user type s,

$$\Delta = \frac{\partial C_s(z|\hat{v}=1)}{\partial z_\theta}$$

We can take a gradient step Δ and re-evaluate. Upon achieving a satisfactory outcome, we can re-project z_θ back to a robot-executable trajectory $tr^s = Bz^{-1}(z_\theta)$.

The main feature we want in our model is for the latent space to be structured in a way that would allow us to distinguish between trajectories conforming (or not) to the user specifications. In turn, this generates good trajectories. We further need the model to maintain certain kinds of variability in order to allow us to estimate the causal link between the symbols within the world and the validity of a trajectory, given a specification.

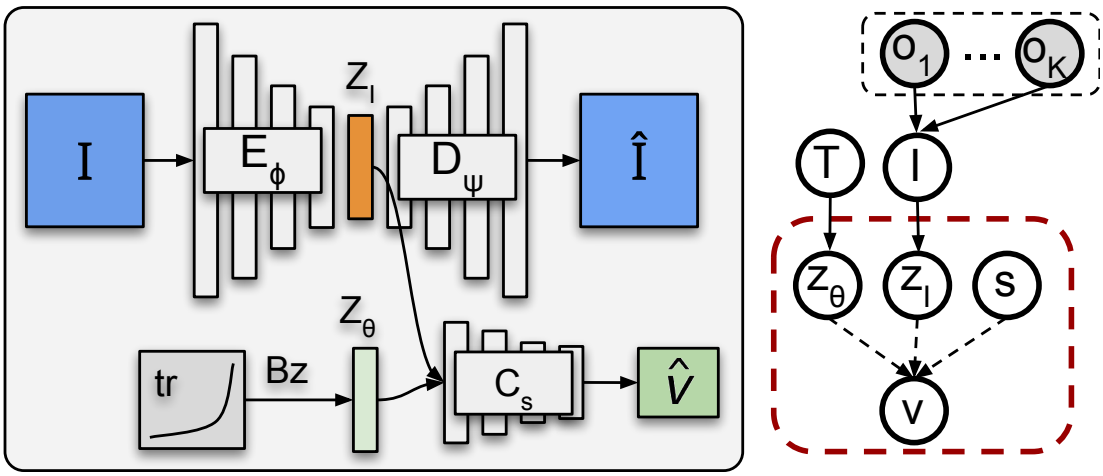


Figure 24: **Left:** Specification model architecture. The environmental image I, $I \in R^{100\times100\times3}$, is passed through an Encoder-Decoder Convolutional Network, with a $16-8-4$ 3x3 convolutions, followed by fully connected layer, to create a compressed representation $Z_I, Z_I \in R^{15}$. It is passed along with the trajectory parameterization $Z_\theta, Z_\theta \in R^2$ through a 3-layer fully connected classifier network that checks the validity of the trajectory $C_s(z)$ with respect to the spec. s. **Right:** The environment, compressed to z_I, is composed of objects $(o_1, .., o_K)$. A trajectory T is parameterized by z_θ, which alongside the factors z_I and user specification s are part of the specification model.

4.4 Specification Model

We use the Deep Variational Auto-Encoder Framework [85] as a base architecture. The full model consists of a convolutional encoder network q_ϕ, parametrized by ϕ, a

deconvolutional decoder network p_ψ, parametrized by ψ, and a classifier network C, comprised of a set of fully-connected layers. The encoder network is used to compress the world representation I to a latent space Z_I, disjoint from the parameterization of the trajectories Z_θ. The full latent space is modeled as the concatenation of the world space and trajectory space $Z = Z_I \cup Z_\theta$ as seen in Figure 24.

$$\begin{aligned} \min_{\psi,\phi,\mathbf{C}} \mathcal{L}(\psi,\phi;I,z_I,z_\theta,v) = \\ &- \alpha \mathbb{E}_{E_\phi(z_I|I)}(logD_\psi(I|z_I)) \\ &+ \beta D_{KL}(E_\phi(z_I|I)||D_\psi(z_I)) \\ &- [v\log(C(z)) + (1-v)\log(1-C(z))] \end{aligned} \tag{11}$$

Parameters — α, β — are added to the terms of the overall loss function — see Equation 11 — so that their importance during learning can be investigated in an ablation study. The values for the three coefficients were empirically chosen in a manner such that none of the separate loss terms overwhelms the gradient updates while optimizing $\mathcal{L}$.

To better shape the latent space and to coerce the encoder to be more efficient, the Kullback-Leibler divergence loss term is scaled by a β parameter, as in Higgins et al. [69]. By tuning its value we can ensure that the distribution of the latent projections in Z_I do not diverge from a prior isotropic normal distribution and thus influence the amount of disentanglement achieved in the latent space. A fully disentangled latent space has factorized latent dimensions — i.e. each latent dimension encodes a single data-generative factor of variation. It is assumed that the factors are independent of each other. For example, one dimension would be responsible for encoding the X position of an object in the scene, another for the Y position, third for the color, etc. Higgins et al. [71] and Chen et al. [31] argue that such low-dimensional disentangled representations, learned from high-dimensional sensory input, can be a better foundation for performing separate tasks - trajectory classification in our case.

Moreover, we additionally add a binary cross-entropy loss (last term in the loss function) associated with the ability of the full latent space Z to predict whether a

trajectory tr^s associated a world I satisfies the semantics of the user type s - $\hat{v}$. We hypothesize that backpropagating the classification error signal through Z_I would additionally enforce the encoder network to not only learn factorized latent representations that ease reconstruction, but also trajectory classification. The full loss can be seen in Equation 11.

4.5 Causal Modeling

Naturally, our causal understanding of the environment can only be examined through the limited set of symbols, O, that we can comprehend about the world. In this part, we work under the assumption that an object detector is available for these objects (as the focus of this work is on elucidating the effect of these objects on the trajectories rather than on the lower level computer vision task of object detection per se). Given this, we can construct specific world configurations to test a causal model and use the above-learned specification model as a surrogate to inspect the validity of proposed trajectories. We assume that by understanding the minimum number of required demonstrations per scene, we can learn a model that reflects the expert decisions.

If we perform a search in the latent space z_θ, we can find boundaries of trajectory validity. We can intervene and counterfactually alter parameters of the environment and specifications and see the changes in the trajectory boundaries. By looking at the difference of boundaries in cases where we can test for associational reasoning, we can causally infer whether

- the different specifications show alternate valid trajectories
- a particular user type reacts to the existence of a specific symbol within the world.

4.5.1 *Specification Model Differences*

We are interested in establishing the causal relationship (further described in Chapter.2.2) within the specification model as shown on Figure 24. We define our Struc-

tural Causal Model (SCM), following the notation of Peters, Janzing and Scholkopf [115] as

$$\mathfrak{C} := (\mathbf{S}, P_{\mathbf{N}}),\ S = \{X_j := f_j(\mathbf{PA}_j, N_j)\}$$

where nodes $\mathbf{X} = \{Z_\theta, Z_I, S, V\}$ and $\mathbf{PA}_j = \{X_1, X_2, ..X_n\} \backslash \{X_j\}$. Given some observation $\mathbf{x}$, we can define a counterfactual SCM $\mathfrak{C}_{\mathbf{X}=\mathbf{x}} := (\mathbf{S}, P_{\mathbf{N}}^{\mathfrak{C}|\mathbf{X}=\mathbf{x}})$, where $P_{\mathbf{N}}^{\mathfrak{C}|\mathbf{X}=\mathbf{x}} := P_{N|\mathbf{X}=\mathbf{x}}$

We cannot logistically perform counterfactuals using the data and humans, but by relying on the learned models to have encapsulated the expert representations, we can perform the causal analysis on those surrogate models.

We can choose a particular user specification $s \sim p(S), s \neq s_{\mathbf{x}}$ and use the specification model to confirm that the different specification models behave differently given a set of trajectories and scenes, i.e. the causal link $s \rightarrow v$ exists by showing:

$$\mathbb{E}\left[P_v^{\mathfrak{C}}|\mathbf{X} = \mathbf{x}\right] \neq \mathbb{E}\left[P_v^{\mathfrak{C}}|\mathbf{X} = \mathbf{x}; do(S := s)\right] \tag{12}$$

We expect different user types to generate a different number of valid trajectories for a given scene. Thus, by intervening on the user type specification we anticipate the distribution of valid trajectories to be altered, signifying a causal link between the validity of a trajectory within a scene to a specification.

4.5.2 *Symbol Influence on Specification Models*

We want to measure the response of the specification models of intervening in the scene and placing additional symbols within the world. We use the symbol types $O = \{o_1, .., o_k\}$ as described in Section. 4.7.1. To accomplish this, for each symbol within the set we augment the scene I, part of the observation $\mathbf{x}$ with symbol o, such that $I_{new} = I \cup o$. We do not have the ability to realistically remove objects from the scene, for this reason, our augmentation involves adding such objects, which can be

interpreted as applying an additional overlay of the object on the image. If we observe that the entailed distributions of $P_v^{\mathfrak{C}|\mathbf{X}=\mathbf{x};do(Z_I:=z_{Inew})}$ changes i.e.

$$\mathbb{E}\left[P_v^{\mathfrak{C}}|\mathbf{X}=\mathbf{x}\right] \neq \mathbb{E}\left[P_v^{\mathfrak{C}}|\mathbf{X}=\mathbf{x}; do(Z_I := z_{Inew})\right] \tag{13}$$

then the introduced object o has a causal effect upon the validity of trajectories conditioned upon the task specification $s_{\mathbf{x}}$.

We investigate the intervention of all symbol types permutated with all task-space specifications to build an understanding of the relationship between the manner of execution and the influence of the symbols on it.

4.6 Parameterization of Specifications

This part of the work aims to provide a closed system that decomposes demonstrations into a set of parametrized rules. We have shown methods for ways to construct a model that encapsulates such specifications, using causal analysis to extract symbols that influence the demonstrations. Further, relying on these outputs, we use constraint optimization to find optimal parameters for a set of predefined rules representing the specifications.

We rely on the CP-SAT solver in Or-tools, [111], and formulate a set of rules that can be understood as corresponding to a point in the trajectory being in collision with an object, being in a region of influence of an object or in free-space. We formally define this in the following manner:

$$f(p_i) = \begin{cases} inf, & \text{if } p_i \leq T_{min} \\ ||p_i - p_{obj-k}||_2, & \text{if } p_i \leq T_{min} + T_{object-k} \text{ for any object } k \\ 0, & \text{otherwise.} \end{cases} \tag{14}$$

We would have a penalty constraint that $\sum_i f(p_i) < F_{max}$ for any trajectory $tr = \{p_1, p_2, ..., p_{T_i}\}$, where F_{max} is chosen as the attention buffer for the demonstrator. We are interested in providing a maximum or minimum safety envelope for the

trajectory and would, thus maximize/minimize $\mathcal{L}_T = \sum_k T_{object-k} + T_{min}$. We can observe how the requirements for positive or negative change with the different safety target.

For each point in a trajectory, we add a set of constraints representing the different channels as seen under Equation 14. The sum of penalties for each trajectory is added as a constraint conditioned on the validity of the trajectory. We would then find a feasible or optimal solution for the parameters - $T_{min}, T_{object-1}, ..., T_{object-K}$ under the minimum/maximum cost function.

4.7 Experimental Setup

4.7.1 *Dataset*

Figure 25: Items used for the generation of the training (green) and test (red) scenes.

The environment chosen for the experiment consists of a top-down view of a tabletop on which a collection of items, *O={utensils, plates, bowls, glasses}* - Figure 25, usually found in a kitchen environment, have been randomly distributed. The task that the demonstrator has to accomplish is to kinesthetically move a robotic arm gently holding a pepper shaker from one end on the table to the other (p_{init} =bottom left, p_f=top right) by demonstrating a trajectory, whilst following their human preferences around the set of objects — see Figure 26. The demonstrators are split into user types S, $S = \{careful, normal, aggressive\}$ based on the trajectory interaction with the environment. The semantics behind the types are as follows: the *careful* user tries to

avoid going near any objects while carrying the pepper shaker, the *normal* user tries to avoid only cups, and the *aggressive* user avoids nothing and tries to finish the task by taking the shortest path from p_{init} to p_f.

Agent Input The agent observes the tabletop world and the user demonstrations in the form of 100x100 pixel RGB images $I, I \in \mathbb{R}^{100\times100\times3}$. The demonstrator — see Figure 22 — is assigned one of the types in S, has to produce a number of possible trajectories, some that satisfy the semantics of their type and some that break it — Figure 23.1. As specified in Section 4.3, each trajectory tr^s is a sequence if points $\{p_0, \ldots, p_T\}$, where $p_0 = p_{init}$ and $p_{T_i} = p_f$. Each point $p_j, j \in \{0, \ldots, T\}$ represents the 3D position of the agent's end effector with respect to a predefined origin. However, all kinesthetic demonstrations are performed in a 2D (XY) plane above the table, meaning that the third coordinate of each point p_j carries no information ($P = 2$). An efficient way to describe the trajectories is by using a Bèzier curve representation — see Mortenson [107]. The parameterization of a single trajectory becomes the 2D location of the central control point parametrized by θ, together with p_{init} and p_f. However, the initial and final points for each trajectory are the same and we can omit them. Thus, with respect to the formulations in Section 4.3 $L = 2$ and $Z_\theta \in \mathbb{R}^2$.

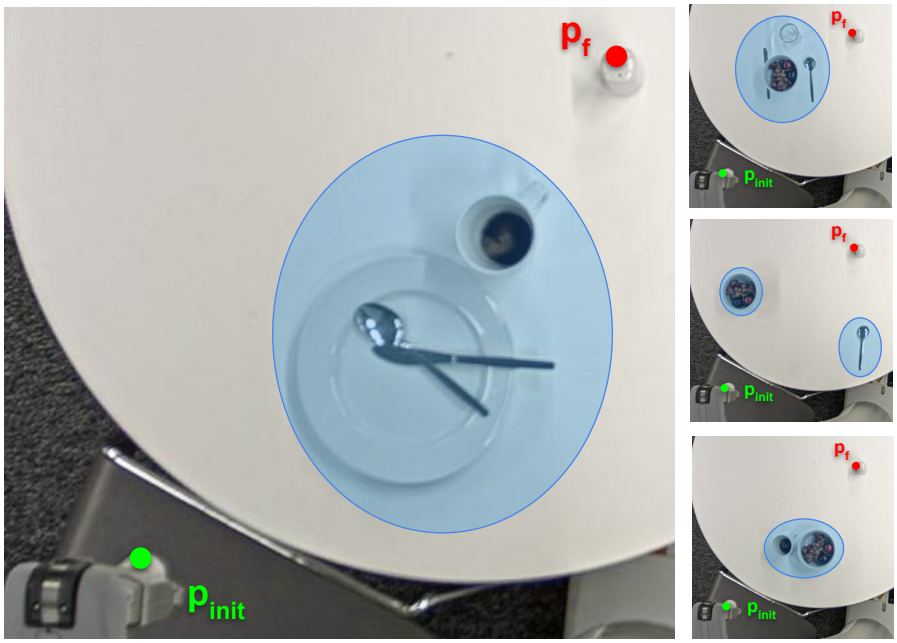


Figure 26: Sample images used to represent example scenes. p_{init} and p_f are as defined in Section 4.3. Blue blobs represent potential obstacles in the scene, which some user types might want to avoid, and are only drawn for illustrative purposes.

In total, for each user type $s \in S$, 20 scenes are used for training, with 10 trajectories per scene. The relationship between the number of trajectories per scene and the model's performance is explored in Section 4.8. For evaluation purposes additional 20 scenes are generated, using a set of new items that have not been seen before — see Figure 25.

4.7.2 *Evaluation*

We evaluate the performance of the model by its ability to correctly predict the validity of a trajectory with a particular specification. We perform an ablation study with the full model ($\alpha \neq 0,\ \beta \neq 0$), AE model ($\alpha \neq 0,\ \beta = 0$), and classifier ($\alpha = 0,\ \beta = 0$). We investigate how the performance of the model over unseen trajectories varies with a different number of trajectories used for training per scene. We randomize the data used for training 10 times and report the mean.

As a baseline we use an IRL model $r_s(p, I)$, such that the policy π producing a trajectory tr^s that is optimal w.r.t.:

$$\arg\max_{tr^s} \sum_{i=0}^{N} r_s(p_i, I)$$

Additionally, we test the ability of the learned model to alter an initially suggested trajectory to a valid representative of the user specification. We assess this on the test set with completely novel objects by taking 30 gradient steps and marking the validity of the resulting trajectory.

We perform a causal analysis of the model with respect to the different user specifications and evaluate the difference in their expected behavior. Additionally, we intervene by augmenting the images to include specific symbols and evaluate the difference of the expectation of their entailed distribution. This highlights how different specifications react differently to certain symbols.

We conclude by finding optimal maximum and minimum parameters for a set of rules that the motion controller can use to plan with varying levels of safety vs travel time. We perform constrain optimization on the task of moving a drill on a workbench robot assembly area as shown in Figure 27 and report results in Section 4.8.4. We

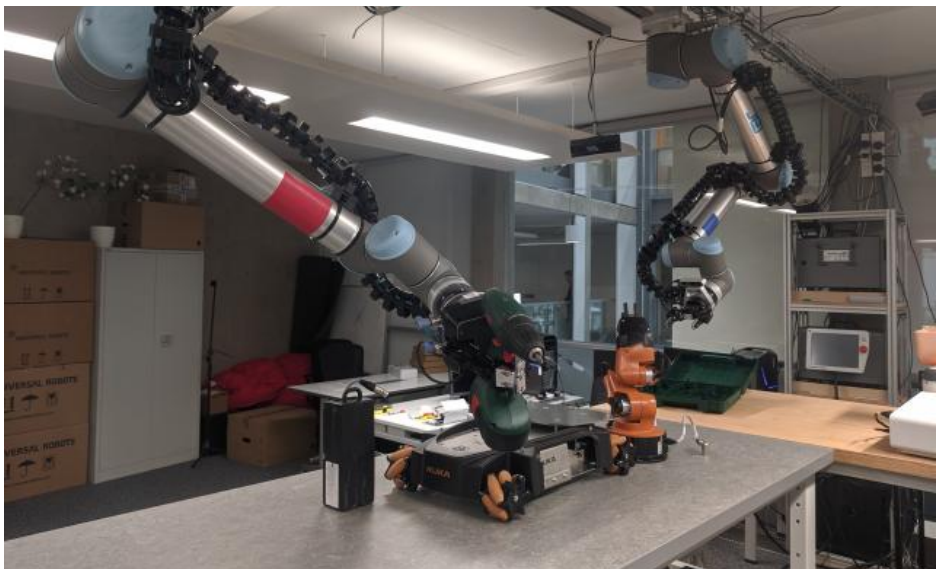


Figure 27: An additional task of moving the the drill to the work space of the other robot.

obtain demonstrations in a representative simulated 2D environment, such that the demonstrated trajectories no longer need to adhere to the Bèzier representation.

The aim is to find the rule parameterization based on Equation 14, such that this representation can, later on, be used for motion planning optimization as an additional cost. We would aim to extract the limits of the parameters to create an envelope of possible costs and not a bound of the geometric models that represent the objects.

4.8 Results

In this section we show how modeling the specifications of a human demonstrator's trajectories, in a table-top manipulation scenario within a neural network model, can be later used to infer causal links through a set of known features about the environment.

4.8.1 *Model Accuracy*

We show the accuracy of the specification model in Figure 28 and on our website[1]. Changing the number of trajectories shown within a scene has the highest influence on the performance going from $72\%[67.3 - 77.5]$ for a single trajectory to $99\%[97.8 - 99.8]$

1 Website on https://sites.google.com/view/learnspecifications

[2] when using 9 trajectories. The results illustrate that the models benefit from having an auto-encoder component to represent the latent space. However, the solutions asymptotically approach perfect behavior as the number of trajectories per scene increases. Interestingly, the IRL baseline shows the need for much more information in order to create an appropriate policy.

If we look into the latent space of the trajectory — Figure 29 — we can see that the trajectory preferences have clustered and there exists an overlap between the different model specifications. It also illustrates what the models' specifications can show about the validity of the trajectory.

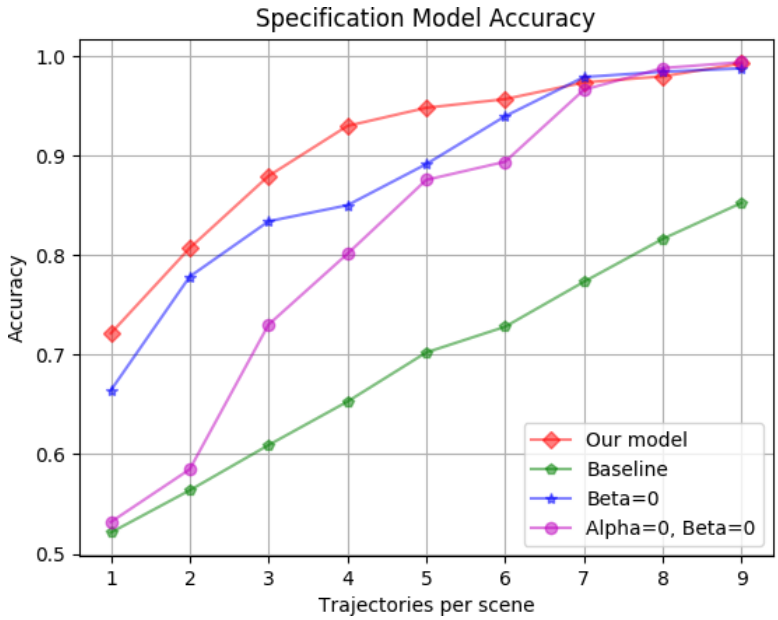
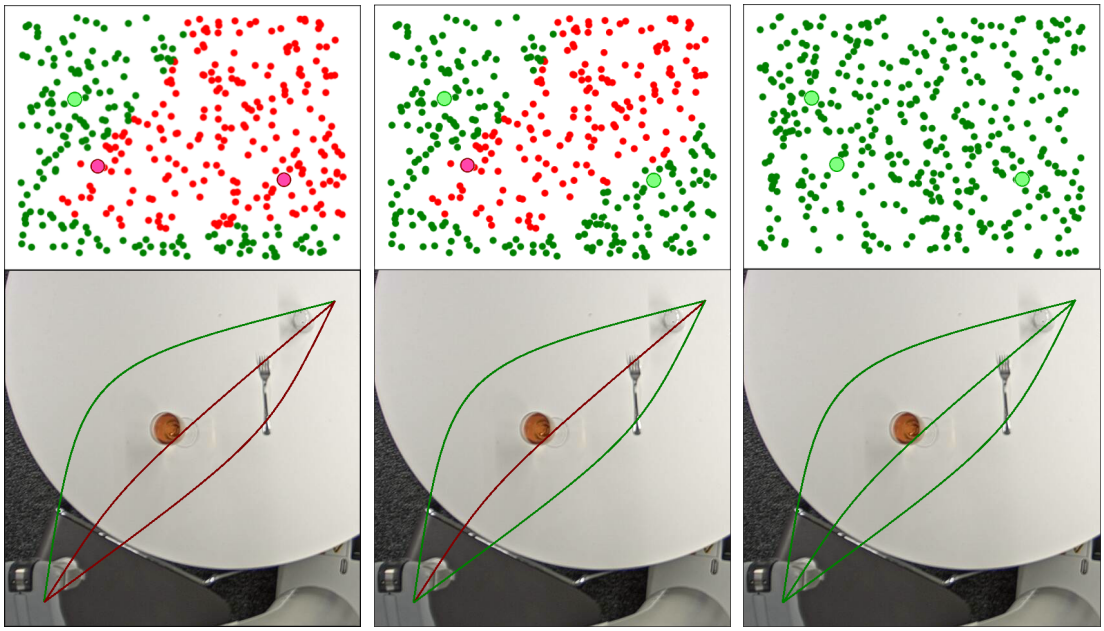


Figure 28: The accuracy of the different models with respect to the number of trajectories used within a scene. The lines indicate the mean accuracy with 10 different seed randomizations of the data. As the number of trajectories per scene increases, the performance of all models improves, but especially with a lower number of trajectories, our full model shows the biggest gains.

4.8.2 *Trajectory Backpropagation*

We can use the learned specification model and perturb an initially suggested trajectory to suit the different user types by backpropagating through it and taking gradient steps within the trajectory latent space.

2 The numbers in brackets indicate the first and third quartile.



(a) Careful (b) Normal (c) Aggressive

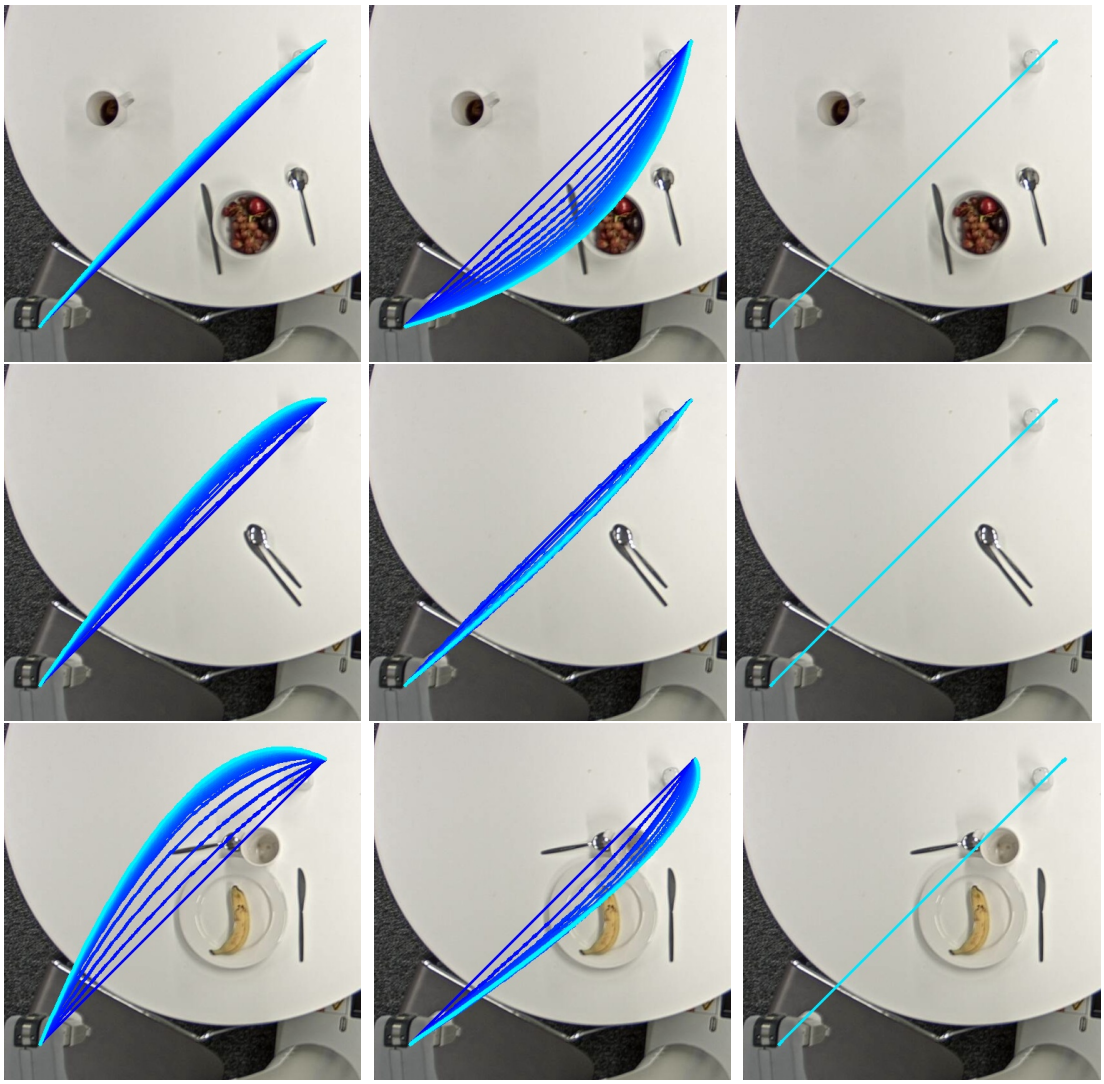
Figure 29: Sampling of the latent trajectory space — Z_θ — of the preference model with different specifications. It can be observed how for the same region in the latent trajectory space — e.g. bottom right — the different user types have different validity values for the same trajectory — e.g. normal vs. careful user types around the cutlery and glass.

Based on the unseen object test scenes, the models were evaluated under the different specifications and the results can be found in Table 3. Individual trajectory movements can be seen in Figure 30.

The first row of Figure 30 shows that the careful user type steering away from both the cup and bowl/cutlery, whereas in the normal user type, the model prefers to stay as far away from the cup as possible, ignoring the bowl. The model conditioned on the aggressive user type does not alter its preference of the trajectory, regardless of its passing through objects. The second row illustrates a situation, where the careful model shifts the trajectory to give more room to the cutlery, in contrast to the normal case. The final row highlights a situation, where the resulting trajectories vastly differ depending on the conditioning of the specification model.

4.8.3 *Causal Analysis*

In Table 4 we can see the mean of the entailed distribution depending on the type of intervention performed. The results of Equation 12 can be seen in the first column



(a) Careful (b) Normal (c) Aggressive

Figure 30: An initial trajectory (seen in dark blue) is used as a base solution to the task for difference scenes — rows 1, 2, 3. Furthermore, the parametrisation z_θ for each initial trajectory is continuously updated so that it better abides by the semantics of the different user specifications — columns a,b,c. It can be seen that as the gradient steps in Z_θ are taken, the resulting intermediate trajectories are shifted to accommodate the preference of the model until the final trajectory (light blue) is reached. Color change from dark to light blue designates progressive gradient steps.

Table 3: The success rate of perturbing a non valid trajectory into a valid one under different user specifications.

User Types	Success rate
Careful	75%
Normal	95%
Aggressive	100%

under "No intervention". It shows the expected likelihood $E[p(v|X = x, S = s)]$ of the validity of a trajectory given a set of observations with different user specifications. Conditioning on the different types of user specifications, we can see that the validity increases (from 0.43 to 1.0), meaning a higher number of possible solutions can be identified. The variety of solutions can be seen in Figure 29. This naturally follows the human assumption about the possible ways to solve a task with different degrees of carefulness. In the case of the final user type, all of the proposed trajectories have successfully solved the problem.

In the subsequent columns on Table 4 we can see the mean probability of validity for when we intervene in the world and position randomly a symbol of a different type within the scene. By comparing the value with the ones in the first column (as discussed above), we can assess the inequality in Equation 13.

Table 4: The respective distributions of validity $p(v|X = x, S = s)$ with different user types depending on the intervention performed for a random trajectory to be valid under the user specification. The first column shows the mean distribution over the information obtained over the observations. The cells in bold indicate significant change with respect to the no intervention column. Those cells highlight a change, which is interpreted as a causal link between the intervened symbol and the user type.

User Types	No Intervention	Bowl	Plate	Cutlery	Glass
Safe	0.43	**0.27**	**0.28**	**0.31**	**0.30**
Normal	0.62	0.62	0.63	0.62	**0.48**
Aggressive	1.00	1.00	1.00	1.00	1.00

In the case of a safe user specification, adding a symbol of any type decreases the probability of choosing a valid trajectory (from 0.43 down to 0.27). This indicates that the model reacts under the internalized specification to reject previously valid trajectories that interact with the intervened object.

For the normal user type, significant changes are observed only when we introduce a glass within the scene. This means it doesn't alter its behavior with respect to any of the other symbols.

In the last case, the aggressive user type doesn't reject any of the randomly proposed trajectories, and that behavior doesn't change with the intervention. It suggests the specification model, in that case, is not reacting to the scene distribution.

Based on these observations, we can postulate that the specification model has internalized rules such as *"If I want to be careful, I need to steer away from any objects on the table"* or *"To find a normal solution, look out for glass-like objects."*.

This type of causal analysis allows us to introspect on the model preference and gives us an understanding of the decision-making capabilities of the model.

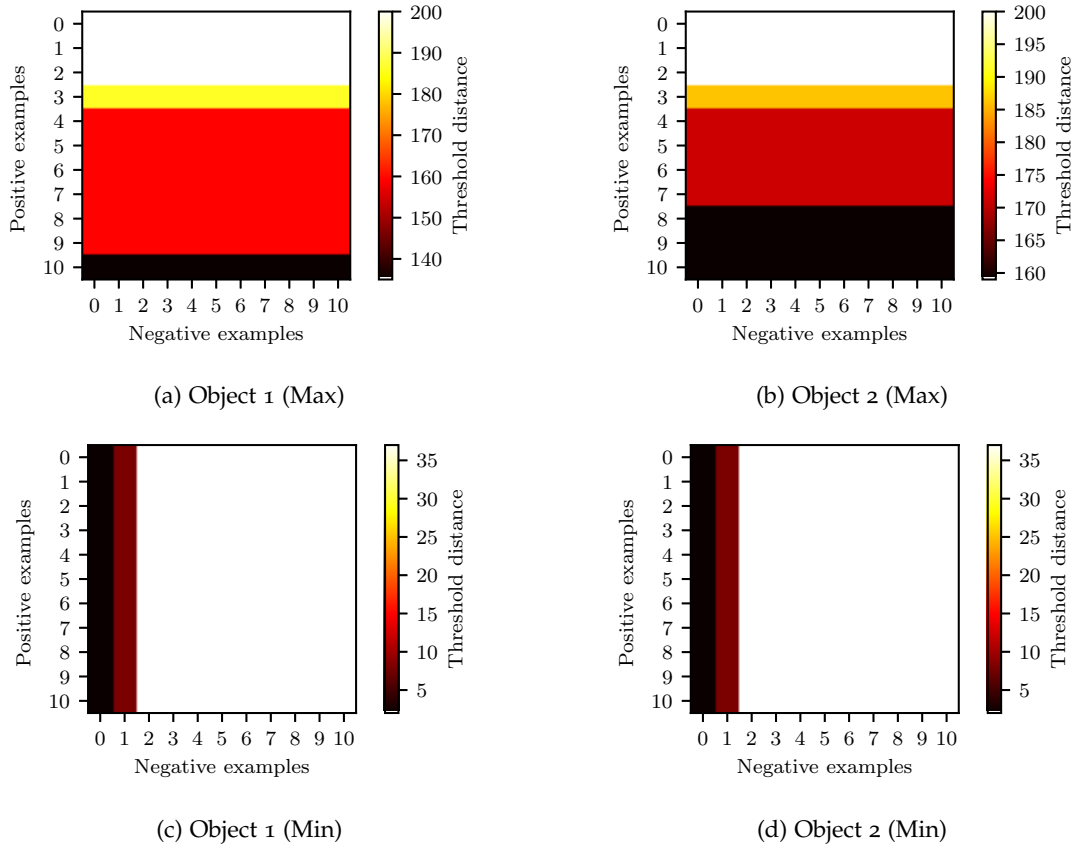
4.8.4 *Parameterization of Task-Space Specifications*

Based on the demonstrated trajectories, we can find parameterization of the rules specified in Equation 14 for a world with 2 distinct objects. We can observe the resulting parameters for object distance for 3 different participants in Table 5. We are measuring the distances in pixel units, and as the camera is orthogonal to the surface, they can be transformed into real-world distances.

Table 5: The object threshold distances found from demonstrations of different participants. The values in brackets indicate the radius when optimizing for the minimal $\mathcal{L}_T$ vs the maximum.

	T_{min}	$T_{min} + T_{object-1}$	$T_{min} + T_{object-2}$
User 1	(36) 59	(37) 135	(37) 159
User 2	(37) 45	(38) 145	(38) 145
User 3	(48) 52	(49) 152	(49) 152

In Figure 31 we can observe the progression of these threshold distances when we alter the number of valid and invalid examples. This allows us to better choose where the future focus should be when obtaining demonstrations for alternative tasks. If we look at Figure 31a-31b to increase the confidence that we have found a maximum safety boundary, we need to counter-intuitively provide more positive examples. Whereas if we are interested in the minimum safety envelope, Figure 31c-31d illustrates that



(a) Object 1 (Max)

(b) Object 2 (Max)

(c) Object 1 (Min)

(d) Object 2 (Min)

Figure 31: The transition of the threshold distance ($T_{min} + T_{object-k}$) for different number of positive and negative examples. We can see the impact of increasing the number of trajectories when we want to find an optimally maximum/minimum distance around an object.

we need to give invalid trajectories. Thus, the true underlying object distance will lie between the observed maximum and minimum boundaries.

The resulting boundaries around the symbols do not necessarily represent the object boundaries, but the expert representation of the min/max expected distance of interaction around them. Combining the rules in Equation 14 and the values in Table 5 allows us to create an additional cost map that can be used to perform motion planning in the scene following the user expectations. Combining this with the causal analysis gives us the ability to incorporate only the required symbols within the planning framework.

4.9 Limitations of Policy and Task Structure

Performing the causal analysis on the surrogate model creates some limitations around complex behaviours, where the surrogate model is non-trivial. There is an assumption around the representation of both the world configuration (image to VAE latent space) and the trajectory (list of points to Bézier fit).

Further constraint is in regards to symbol knowledge - to perform augmentation of the input of the surrogate model, we rely on knowledge about the symbols, and in the current configuration the symbols have no co-occurrence restrictions. If they did, it would imply there is a common confounder and the SCM would be insufficient.

When we perform rule parameterization, it relies not only on knowledge about the symbols, but having a correct prior about their composition — a good fit can occur when the programmatic representation of the specification matches the internal human model. This representation needs to be handcrafted. A possible extension would be to use symbolic AI and symbol manipulation to jointly parameterize and find the rules themselves [114].

Extracting these symbols and performing policy optimization is further explored in Chapter 5, where we relax the assumption of symbol knowledge with the inductive property that salient trajectory states are associated with symbols. We use this to build up a list of salient regions.

4.10 Conclusion

Learning behavioral types is essential for completing interactive human-robot tasks. It helps avoid nuisance and promotes better foresight into human actions and plans. Being able to decompose those user types into interpretable and reusable models is of high importance.

In this chapter, we demonstrate how to construct and use a generative model to differentiate between behavioral types, derived from expert demonstrations. We show how performance changes with the number of trajectories illustrated in a scene. Additionally, by using the same learned model, it is possible to change any solution to

satisfy the preference of a particular user type, by taking gradient steps in the latent space of the obtained model.

Performing causal analysis allows for the extraction of causal links between the occurrence of specific symbols within the scene and the expected validity of a trajectory. The models exhibit different behaviors with regard to the different symbols within the scene leading to correctly inferring the underlying specifications that the humans were using during the demonstrations.

Further, by assuming an underlying set of specifications that users follow, it is possible to find the safety envelope boundaries for the objects within the scene. Additionally, we investigate what type of demonstrations would help move the minimum/maximum side of this boundary toward the optimum.

This chapter concludes by showing a method that converts demonstrations into a set of functions that represent the underlying specifications. Those are specifically linked to objects within the world and are causally discarded for uninteresting objects.

This can increase the diversity of policies that can be composed to solve long horizon tasks. The next Chapter will look at how interventions can be performed during the demonstration, and how those can help make the policies more robust and highlight any salient parts of the task area.

5

LfII: Learning from Inverse Intervention

In this chapter we look at how a 'physical' dialogue between the expert demonstrator and the agent can help the learner with the core structure of the problem and remove possible variations of the task.

LfD facilitates sample-efficient learning of task structure that would have been difficult to induce automatically. This is typically a supervised or semi-supervised learning process. The robustness of the resulting policy is highly dependent on the variability of the demonstrations which the learning agent receives. With temporally extended tasks, it is likely that variability may be limited around hard-to-execute segments or specific sub-goals, especially if the demonstrator is used to acting habitually in certain ways. In this chapter we propose augmenting LfD with interventions initiated by the robot learner (LfII). Thus, the robot would *query* the human demonstrator about the underlying structure of the problem by intervening (in time and space) where it believes the demonstration could be modified while preserving the underlying 'concepts'.

This is a form of data augmentation — as discussed in Chapter 2 — enabling the essential variability in the demonstrations to be captured, towards achieving improved robustness of the resulting controller to perturbations. We demonstrate this first in a 2D navigation example and then with a robot control task involving a Panda arm performing table-top manipulation. We show that this process helps identify salience regions where intervention usefully augments the original data set.

5.1 Introduction

The main benefit of robot learning is the enhanced adaptability enabling applicability in a broader range of domains and environments compared to predefined controllers. However, this is crucially dependent on the learning system being exposed to sufficient variations while training, in order for the models to acquire all essential aspects of the underlying 'concepts'. Robot learning researchers have typically succeeded through careful thought around ways to achieve suitable diversity, e.g., to changes in visual appearance and shape of objects, articulation, and movement trajectories.

Learning from Demonstration [11, 126] is an effective mechanism by which one is able to obtain examples of the execution of a task respecting such allowable variations. In this paradigm, the expert provides data through e.g. kinesthetic teaching, teleoperation, or illustration. However, the problem remains for the robot to infer what the underlying concepts are, and crucially what are the allowable variations in the spatio-temporal data associated with a specific task. When we require high levels of reliability in the final learned models, e.g., in some industrial or medical applications, we must think carefully about the *coverage* achieved within the dataset and the extent to which the long tail of the data distribution has been captured to facilitate eventual inference [17, 52, 146]. A compounding factor is that within demonstrations of realistic robot tasks, there is often a significant degree of self-similarity and redundancy, due to the nature of the task or even the comfort spaces of the demonstrator.

We aim to tackle these challenges by allowing the robot to intervene during the demonstration, querying about the underlying structure of the problem with a system overview shown in Figure 32. As a result, the robot pushes the expert to visit potential regions where their policy may perform worse. This intervention is equivalent to the robot asking the question: *"Is this variation task-relevant or can I safely alter your demonstration in this way?"*. Assessed from an active learning perspective, additional data points in regions where the policy is already in accordance with the concepts implicit in the demonstrations would not contribute substantially to better generalization. In our proposed approach, the intervention is performed online at data collection time, in contrast to methods like Dagger [121]. Our approach differs from

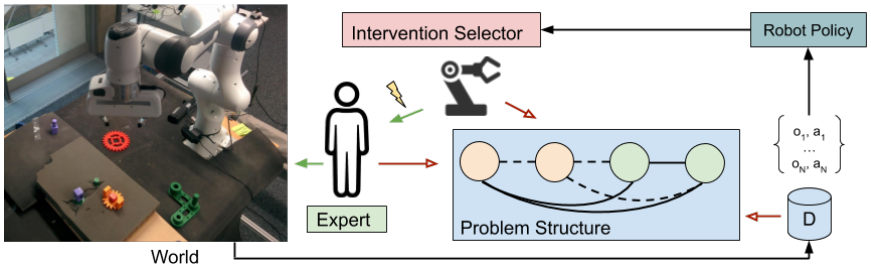


Figure 32: The forward pass to obtain new demonstrations follows the black and green arrows - the expert is demonstrating the task, interrupted by the intervention strategy to augment the demonstration. The resulting trajectories are then stored and used to update the robot policy, which is key to determining where interventions are applied. As an offline "backward" pass (black and red arrows), the problem structure is identified by creating a problem graph and assessing the salience of the different nodes. This is done by (1) creating trajectories that avoid particular nodes in the graph or (2) using the expert to help assess the value of the node from a sampled reconstructed image.

the one described by Spencer et al. [136] in that the initiative is taken by the robot - rather than the human telling a robot (which acts as a passive receiver of information), here the robot actively modifies trajectories at demonstration time to ask if a certain variation is valid under the implied concept.

This is especially relevant to long-horizon tasks, as the demonstration trajectory is bound to be visiting a sequence of important sub-goal areas. Typically, all sub-goals after the first one will have a limited initiation set, often overlapping with the termination of the previous one as shown by Angelov et al. [4] and Dautenhahn and Nehaniv [41]. This could lead to fragile control policies, as the data may not be diverse around these points, implying that even a small disturbance could push the robot state outside the support of the data. This provides additional information in regards to how vital it is to visit particular areas of the state space for successful task execution. If the robot intervenes and moves away from an area, but is brought back by the expert, it gives useful information about the salience of that region.

In this chapter we make the following contributions:

- We propose Learning from Inverse Intervention (LfII) during demonstration, which diversifies the demonstrations obtained from the expert. This is an online data augmentation approach based on performing targeted interventions to elicit information from the human expert. This method is agnostic to the underlying robot or state space.

- We demonstrate increased robustness of the resulting policies, due to wider coverage of the distribution of states.
- We show the identification of salient locations, in both a state-based and vision-based setting.

We evaluate the proposed approach first in a navigation task that requires visiting a sequence of states with full observability of the state. Next, we demonstrate this method applied to a robotics inspection task in a static and dynamic environment, with the observable state space corresponding to a camera view on the robot. We demonstrate greater robustness to random perturbations, and also that we can identify salient locations and generate corresponding images.

5.2 Related work

5.2.1 *Salience Identification*

In situations with multiple or ambiguous goals, a good strategy to clarify the objective is to query the human expert. The work of Tellex et al. [143] provides a method where a robot in a failed state of the task graph, can ask a human participant to provide help or insight into the problem to aid the solution. In our work, the algorithm initiates visual queries for the human to confirm the salience of a particular state.

An alternative method for evaluating salience is through partitioning a large task into sub-problems as with options [50, 70, 139] to construct a hierarchical policy. The termination states of options relate to task structure. A method to identify these sub-problems is through frequency analysis or "bottleneck" states as described by Stolle and Precup [137]. However, these methods are often limited to discrete tasks with plentiful data. In our work, we build on both these approaches and extend them to a continuous state space, sparse-data setting (i.e. limited number of rollouts), where the trajectories are generated from human demonstrations. While the previous methods use passive observations, we can counter-factually assess if visiting particular parts of the state space is necessary for solving the task.

This is similar to evaluating salience in demonstrations through mapping them back to a reward structure using IRL [21], confidence based methods [33], through key-frame detection [78], or relying on Semi-MDP graphs as constraints [67]. In our case, we use counterfactual trajectory and image generation to assess importance.

5.2.2 *Demonstration Strategies*

Data from imitation learning is typically used in a supervised learning setting to create an initial/prior policy, such as in Kim et al. [84] and Kober, Bagnell and Peters [87]. Or relying on simulation as a proxy to ask humans for performance evaluation [134]. This can be further optimized by alternative reinforcement learning strategies.

Methods for augmenting the rollouts of the agent with human help, such as Dagger [121] and GAIL [72] are typically offline. Here, distribution matching is performed as an augmentation of the data. Similar to our method, Spencer et al. [136] uses interventions online, altering the robot trajectories. An advantage of our method is that the *robot* performs an intervention on the human online, during the demonstration, in effect holding a kinesthetic discussion about the structure of the problem.

The way that the agent chooses when to intervene in this dialogue is similar to the work on curiosity, e.g., episodic curiosity - based on errors in observations [124], or by following errors in the dynamics predictions [23]. In the current chapter, the agent's drive for exploration comes from the alignment of the internal policy to the current demonstration that is being obtained — in other words — the new trajectory not being sufficiently informative.

5.3 Problem Formulation

When people demonstrate tasks to each other, the semantics would not always be clear without the benefit of clarification questions [118]. When teaching sensorimotor skills, this takes the form of kinesthetic nudges. Our main goal is to embody this behavior in robot learning.

Thus, in contrast to the robot merely exploring on its own, starting from an initial trajectory, or the human user providing verbal labeling of trajectories or piece-wise annotations post-hoc, in our model the robot proposes deviations to the demonstrations in response to which the human expert either corrects (hence providing information about the need for specificity) or ignores that variation from which the robot infers redundancy in task specification.

In technical terms, this means that we compare the expert demonstration against the current robot policy using cosine similarity. The robot policy is updated after *each* demonstration to encode the newly learned information.

To assess saliency, we use the nodes constructed by the algorithm above with additional robot rollouts (which are chosen to avoid regions of the space similar to the interventions during the demonstration) or through Visual Q&A (directly asking the expert for clarification of the state). This is how the human expert provides feedback during*the demonstration process.

To compare this paradigm against alternatives, we have chosen examples that are similar to prior work, e.g. D. Precup et al. [137] count-based bottleneck state detection - represented in our baselines as intervention type 'none' and 'random_eps' (dependent on the original policy) to the proposed methods - 'conditional_random', 'conditional_lfii'.

Then, going beyond the scenarios of static scenes, we adopt the example from Burke, Hristov and Ramamoorthy [24], showing that our approach can cope with a dynamic scene. This example is chosen to be simple enough to clearly bring out where count-based methods would be ineffective and how our proposed approach fares.

5.3.1 *Diverse trajectory demonstration*

Consider the MDP setting wherein the learning agent models the environment through nodes in a graph, with actions causing transitions between them. In another variation, a group of observational states would map to nodes in that graph.

In the first part of our work, we are interested in using trajectories from a task to infer the underlying variability of robot states that map to the different nodes in

this graph. Intervention is used to expand the scope of observations for a node. As we construct the graph through different roll-outs, obtained from the trajectories, we should note that each node needs to contain states from each trajectory. This builds up the underlying representation of the task, which may not be necessarily identifiable in the domain of vision-based robot control.

As part of task learning, we incrementally construct the graph, $G = (V, E)$, where V is the set of vertices or nodes and $E \subseteq \{\{x, y\} | (x, y) \in V^2 \wedge x \neq y\}$ is the set of edges or transition between nodes. In the beginning, the graph is fully connected. As learning proceeds, we aim to reduce the edges to highlight critical nodes.

The set of nodes $V = \{v_1, v_2, ...\}$ is initialized from at least 2 demonstration trajectories, $D = \{demo_1, demo_2, ...\}$, where $demo_i$ is a sequence of observations $demo_i = \{o_1^i, o_2^i, ...\}$. Both v and o are in $\mathcal{R}^d$, which is the underlying dimensionality of the observations. We set v_1 such that it is a hyper-sphere containing at least one observation from each demonstration $o^i, o^j, ...$ When we receive another demonstration - $demo_i$ coming from the expert policy Π, we use Welzl's algorithm [150] to recompute the hyper-sphere parameters (center and radius), such that each of the nodes in V now also contains observations from the new trajectory.

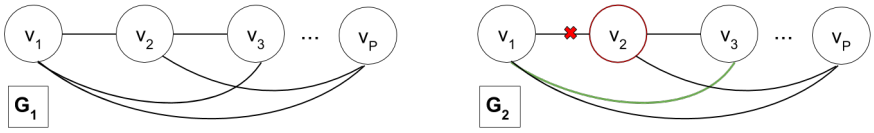


Figure 33: G_1 represents the initially connected graph after node generation. G_2 is an example of an attempt to decrease the number of edges connecting the nodes, to surface the underlying structure of the problem. There is an intervention between the edge connecting v_1 and v_2, making the next possible action transition the agent to v_3 and beyond.

When we obtain demonstrations (from Π), we aim to get high diversity of observations, especially in locations where our model policy π generates actions misaligned to the demonstrations. In order to do so, we compare the running average of the last k steps in terms of how well the policy matches the current demonstration $\Delta_\pi = \frac{1}{k} \sum_{j=i-k}^{i} cos(\pi(o_j), \Pi(o_j))$. If the demonstration aligns well with the robot policy, $\Delta_\pi < e_r$, we perform an intervention of magnitude M (chosen empirically) and direction $\hat{V}_I$, storing those locations in D_I. This intervention is performed during the demonstration by creating a low gain attractor point at the intervened location, whilst

the robot is impedance controlled to follow the human guidance [112]. This is summarised in Algorithm 1 along with the auxiliary task of creating the task salience graph.

The exploration of the agent is modulated by changing e_r - increasing it would trigger more exploratory behavior. An expansion along the lines of $\epsilon-$greedy methods would therefore be $p(1-\epsilon)*e_r + p(\epsilon)*1$. The intervention $\hat{V}_I$ can be in a random direction (*'conditional random'*) or orthogonal to the current policy vectors (*'conditional lfii'*). An advantage of our method is choosing the right location along the trajectory to perform the intervention, rather than performing it based on some noise variable.

Algorithm 1: Learning from intervening during demonstration

Input: Robot - **robot**, exploration const - $\mathbf{e_r}, k, M$, total number of demonstrations - N
Output: Dataset - **D**, Control Policy - **ß**

add a set of initial demonstrations $\mathbf{D} \leftarrow \{demo_1, demo_2, ..., demo_n\}$;
init nodes $\mathbf{v} = compute\_nodes(\mathbf{D})$;
compute control policy $\pi from \mathbf{D}$;
for *next* $demo_i, i \in [n, N]$ **do**
    $D_I = \{\}$;
    for *step* p *in* $demo_i$ **do**
        $\Delta_\pi = \frac{1}{k}\sum_{j=p-k}^{p} cos(\pi(o_j), \Pi(o_j))$;
        if $\Delta_\pi <= \mathbf{e_r}$ **then**
            // demonstration aligns well with robot understanding; let's intervene;
            $V_I \leftarrow \pi(o_i) \times random\_vector$ // not collinear to $\pi(o_i)$;
            $\hat{V}_I = M * \frac{V_I}{|V_I|}$;
            robot$.execute(\hat{V}_I).duration(k)$;
            append $demo_i\{p, ..., p+k\}$ to D_I
    update **v** with $demo_i$;
    append $demo_i$ to **D**;
    update policy π from $\mathbf{D} \setminus D_I$;
return $\mathbf{D}, \pi, \mathbf{v}$;

5.3.2 *State salience*

The diverse set of trajectories obtained above gives nodes in the graph that are representative of different locations within the state space. Identifying sub-cliques in the graph would highlight those nodes that connect these subgroups, thus are important for traversing the graph. We aim to uncover these salient nodes with the

expert in the loop. We present two methods to tackle this node exclusion problem, using either a static query or a dynamic plan within the environment.

- Option 1: Through augmented trajectory — We generate a trajectory within the latent representation of the planning policy to exclude clusters to be intervened on (see Figure 33) by minimizing the deviation from the policy itself - external effort. We then ask the human to judge the success of the policy;
- Option 2: Through visual Q&A — Relying on a generative model of the state space allows us to ask the question *"Is going through this node [image/sequence of images] necessary to complete the task?"*. This is especially useful in situations where we need to avoid failures. We provide samples that are exponentially distributed from the center of the node to evaluate salience.

5.4 Experimental Setup

In this section, we apply LfII during demonstrations in two distinct cases. In the first one, we examine the situation where the observation/state space is simply structured, such as Euclidean space, as is common in robot navigation. In the second case, we consider observations from higher-dimensional space, e.g. camera video stream, with the 'state' being built from the latent representation of a machine learning algorithm.

Structured Space In this first experiment, we have a robot perform a 2D navigation task, loosely inspired by the room traversal problem in [139] and converted to a continuous domain. The objective is to visit locations loc, such that ($loc_x = x, loc_y = cos(x) = sin(x)$ for $x \in [0..4\pi]$), which repeat every 2π, for a total of 4 points (red targets on Figure 35). To succeed in reaching a location, the agent needs to be within $\epsilon_{range} = 0.2$ from the target location. The agent begins at $(loc_x = 0, loc_y = uniform(-\pi/2, \pi/2))$. We perform the demonstrations either through the keyboard (total of 8 directional actions) or through an expert policy Π, which navigated in a direction towards the next unvisited location with directional noise of $0.1rad$ and a step magnitude of 0.1. For the control policy, learned by the agent, we used a Gaussian Process with RBF kernel with scale 1.

The agent has access to its location - 'loc' and has the ability to perform an action that would change the location by *delta* - the traversed distance in x and y. It has to learn the policy for traversing the space whilst visiting the above specified locations, as well as extracting regions of the space that are considered salient whilst being demonstrated the task. Perfect performance would be achieved by completely extracting the target locations.

We investigate obtaining demonstrations under 4 settings — (1) none - the demonstrations are obtained and used to learn the control policy, (2) *random_eps* - we perform an intervention with probability $\epsilon = 0.05$ in a random direction with the same magnitude as the expert steps, (3) *conditional_random* - we use the condition defined in Section 5.3.1 to perform a random intervention (as 2), (4) *conditional_lfii* - when the condition is triggered, we perform an exploratory intervention, orthogonal to the currently learned control strategy. This explores trajectory augmentation options including offline methods (1), greedy exploration (2), and ways to perform the intervention (3, 4).

We evaluate how well the resulting nodes match the ground truth area of interest, with good overlap suggesting that it is a viable method to locate salient locations within the state space. Those can be identified within the trajectory for the construction and evaluation of the policy.

Further, we employ the two options to evaluate whether the particular nodes are salient or not. For Option 1 we construct a trajectory using the learned dynamics model of the task and sample intervention locations and orientation of the applied force maximizing the distance from the center of the node. We rollout the trajectory in the world and evaluate if (1) the resulting path crosses through the intervention node (as defined by center and radius) and (2) if it has succeeded in solving the task. If both conditions are valid, we argue that that particular node is not salient, as it was not needed for solving the task. Alternatively, we collect the resulting nodes.

For Option 2, we substitute the human evaluation with an expert evaluation policy that samples from the node using the following limited exponential strategy.

```
func sample_limitedExp(c, r):
    v = unit_vector(Uniform(-pi, pi), dims=|c|)
    do: dist = ExpDistribution(scale= r) until dist < r
    return c + dist * v
```

The policy averages the samples and assesses if they are within the region of interest for the task. Upon success, the node is kept as salient.

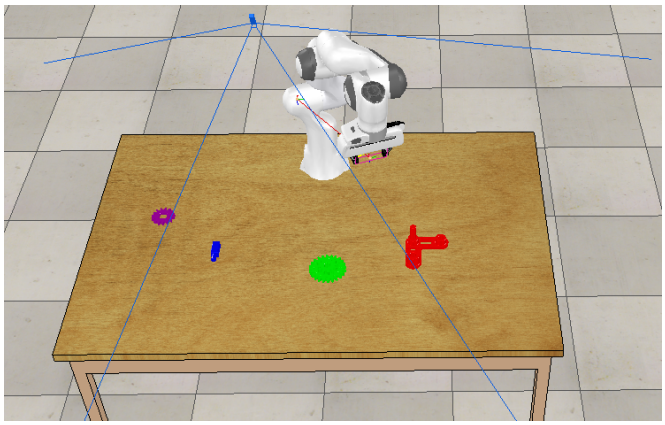


Figure 34: Tabletop inspection task. The robot needs to visit in a sequence the red base, green gear, blue peg and finally, the purple small gear. The blue ray-projecting camera illustrates the perspective from which the network observes the scene. In the second robot experiment, the parts are moving on the tabletop surface.

Learned Space The last experiment is based on an inspection task (defined in [24]), where the Panda robot is sequentially inspecting different parts of a gear assembly. This is implemented in Coppeliasim, as shown in Figure 34. The latent space for the algorithm is learned from the visual observations of the robot. This is achieved through a Res-Net style VAE network with additional perceptual loss. The generative part of the network aimed to produce sharp images, which can be used in visual Q&A. That would allow the expert to "answer" questions about the salience of the different nodes. The demonstrations were either not augmented, or were in the form of perturbations as described above.

The agent receives as input high dimensional image information and performs an action resulting in the 3D offset and motion of the robot arm. The goal of the agent is to learn a policy that solves the demonstrated task robustly and uncovers salient areas of the task, later reconstructed from visual inspection from the latent space of the model.

The VAE neural network is fully convolutional based on 7 Res-Net style blocks (see Table 6 in the encoder and decoder (see Table. 7)that take the 256×256 RGB image to create a 512-dimensional latent space following a Gaussian distribution. The first two trajectories were collected and used to initially train the network over 100 epochs with linear loss decay. Subsequently, after each new trajectory, the network was

fine-tuned with all available data up to that point. The full loss used was a combination of the reconstruction loss, β-weighted KL-Divergence loss, and perceptual feature loss. The β value was set to 0.1 and the perceptual loss was calculated using the mean L2 difference between the features of the original image and the reconstructed on a pre-trained on ImageNet VGG19 network.

Residual Down	Residual Up
out = ReLu(skip + aug)	out = ReLu(skip + aug)
BatchNorm2D(c=cout)	BatchNorm2D(c=cout)
Conv2D (k=3, p=1, c=cout)	Conv2D (k=3, p=1, c=cout)
AvgPool2D(scale=2)	Upsample(scale_factor = scale, mode = "nearest")
ReLu()	ReLu()
BatchNorm2D(c=cin//2)	BatchNorm2D(c=cin//2)
aug = Conv2D (k=3, p=1, c=cin)	aug = Conv2D (k=3, p=1, c=cin)
Conv2D (k=3, p=1, c)	Conv2D (k=3, p=1, c)
skip = AvgPool2D(scale=2)	skip = Upsample(scale_factor = scale, mode = "nearest")

Table 6: The residual blocks have the following ResNet-like structure.

Encoder	Decoder
	Output Image [256, 256, 3]
logvar = Conv2D(2048, 512)	Conv2D(32, 3)
μ=Conv2D(2048, 512)	ResUp(64, 32)
ResDown(1024, 2048)	ResUp(128, 64)
ResDown(512, 1024)	ResUp(256, 128)
ResDown(256, 512)	ResUp(512, 256)
ResDown(128, 256)	ResUp(1024, 512)
ResDown(64, 128)	ResUp(2048, 1024)
ResDown(3, 64)	ResUp(512, 2048)
Input Image [256, 256, 3]	Latent Input [512, 1, 1]

Table 7: The ResBlocks from 6 are joined to form the Encoder and Decoder.

The resulting latent space was used to train the dynamics and action policy. Similar to the above case, they are Gaussian Processes with RBF kernels. They were trained on those parts of the data that do not include any interventions. The action policy generates 3D offsets of the robot end-effector. These policies are retrained after each update of the latent space.

We investigate the robustness of the resulting policies to action perturbation. We actively perturb with $\epsilon = 0.1$ and a displacement of $\times 2$ the magnitude used in interventions during the learning process. We measure the minimum distance (and standard deviation) of the resulting trajectory to the target locations of the different parts of the task over 10 runs.

Finally, we evaluate the salience of the nodes through visual Q&A with a human expert. The inspection task primes the policy that particular locations in the state space need to be visited. As a result, given that all trajectories need to pass through these particular nodes, we perform an inspection on the ordered list of nodes by volume starting from the smallest.

To increase the complexity of the representation learning task, we augment the world, by making the assembly parts that need to be inspected move on the tabletop surface. This would force the latent space of the neural network to represent a finer level of discrimination between different states.

5.5 Results

Structured Space The first experiment involved a 2D navigational task. We evaluated the performance of the model conditioned on the different types of data that are obtained with the varying intervention strategies.

We separate the work-space of the policy into 4 regions. Table 8 shows the success of different policies. A fully robust policy under coverage driven verification can reach the target from all possible starting locations with a score of 1. We see that for Region 0, where the initiation set of all of the policies is larger, the performance of the models is higher than for other regions. In the current instance, after 12 demonstrations, the policy with no interventions performs best. This is possibly due to a better distribution of the initial location. For the following regions, we observe that having a targeted intervention increases the performance of the control policy. This highlights that higher variability in demonstrated trajectories increases the region of operation of the policy. Thus, it is harder for perturbations to push the policy outside the region of support.

Table 8: Performance of different policies starting from uniformly sampled locations within the corresponding region. The value indicates what part of these trajectories reaches the region goal. The regions are selected as follows - starting locations until goal 0 - region 0, between goals 0 and 1 - region 1, etc.

Intervention type	Region 0	Region 1	Region 2	Region 3
none	***0.55***	0.20	0.38	0.37
random_eps	0.54	0.27	0.35	0.42
conditional random	*0.38*	0.28	0.34	0.41
conditional LfII	*0.46*	**0.64**	**0.42**	**0.49**

The increased variability of trajectories should also have the effect of better encapsulating salient nodes from the samples. We evaluate this by calculating the IoU between the best matching node and the ground truth regions. Table 9 shows how well the node creation algorithm matches the target areas. We observe that greater variability methods have a better overlap value. Usually, those methods overestimate the region rather than giving a partition.

Table 9: IoU of the clusters in regards to the ground truth regions of interest after only 12 trajectories. Region 0 illustrates that with good randomization of the starting location, the IoU improves in all methods. To sustain this improvement, we need active intervention during the demonstration (conditional random and LfII).

Intervention type	Region 0	Region 1	Region 2	Region 3
none	*0.45*	0.05	0.10	0.08
random_eps	*0.53*	0.06	0.09	0.10
conditional random	*0.45*	0.23	0.27	0.21
conditional LfII	***0.55***	**0.85**	**0.71**	**0.46**

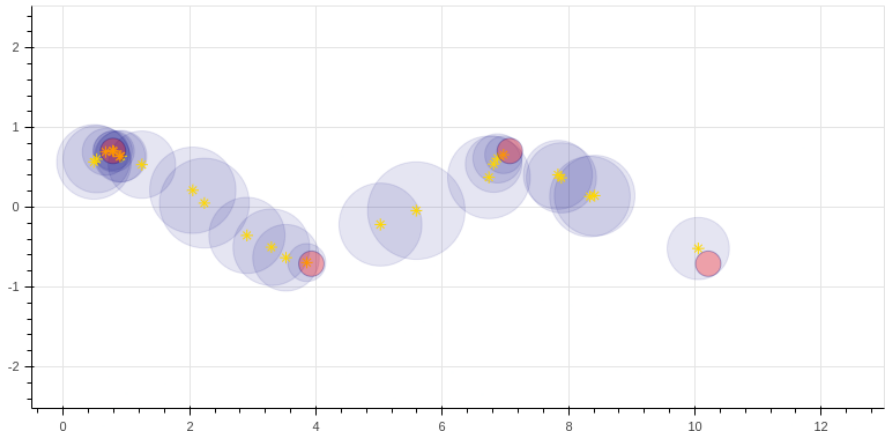
To evaluate the true salience of the nodes, we test two options - (1) intervening in a trajectory to create one that avoids particular nodes, (2) performing visual Q&A with a human expert.

We sample points along the trajectory, alongside a vector of intervention for $1e^5$ cases, selecting those with the largest distance from the node. These are then the nodes to be avoided. If a trajectory succeeds despite avoiding the node, then that node is considered not salient.

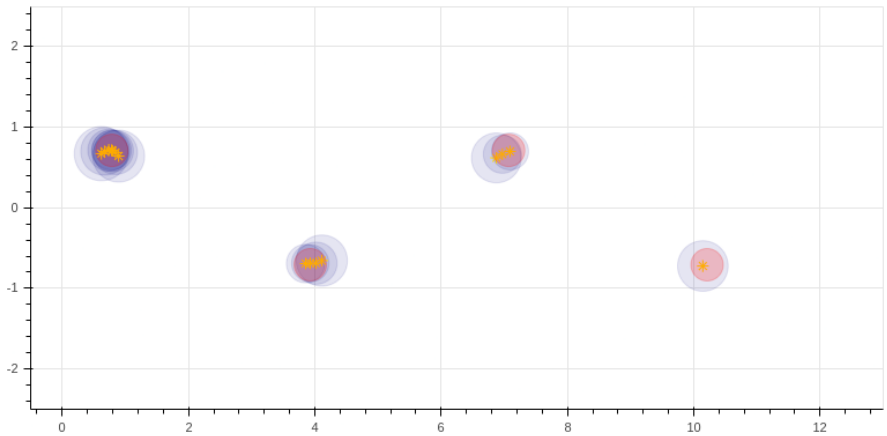
Figure 35a shows the resulting salient nodes, showing a $\sim$ 90% decreases in the number of possible salient nodes. A few larger areas were unavoidable and remained

positive, despite not being relevant to the task. This shows a more proactive control strategy could have been used.

In the case of Visual Q&A, Figure 35b, the nodes were successfully cleared up, with only those overlapping substantially with the true areas of interest remaining. This shows that the method is capable of correctly identifying salient locations without overburdening the human expert.



(a) Trajectory Intervention

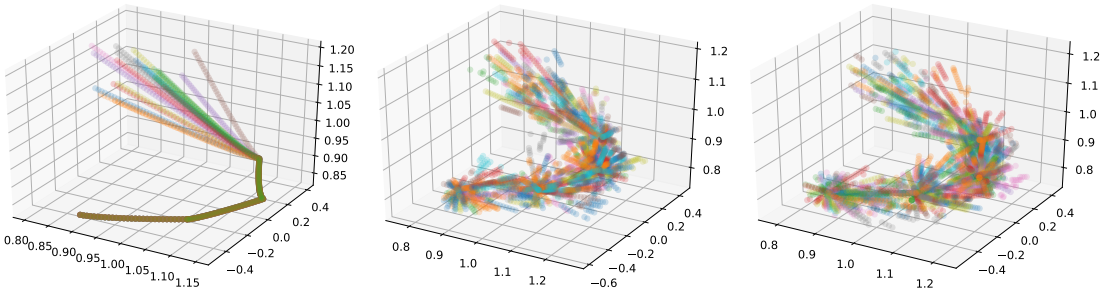


(b) Visual Q&A

Figure 35: Important clusters after trajectory generation through sampling and those after point Visual Q&A. The blue regions are proposed nodes, red are true areas of interest.

Learned Space For this experiment, the Panda robot needs to sequentially inspect parts of an assembly on a tabletop. During the demonstrations, a set of 3 different intervention strategies was used, producing a total of 22 trajectories. The first two trajectories were used to initialize the nodes in the MDP graph and to train the latent space and control models.

The difference in trajectories can be seen in Figure 36.



(a) No intervention (b) Conditional random (c) Conditional LfII

Figure 36: The 22 trajectories used for the task - we can observe that when using interventions, the state space coverage increases and would provide better representation.

Through having greater variability in the trajectories, we assess their resistance to external disturbances. We simulate this by adding an action disturbance. The results after these trajectories can be seen in Table 10. The minimum distance between the trajectory and the part is measured across 10 different rollouts. For part 1 the performance of the methods is similar. As earlier, the initiation set expands and better covers the state space. For part 2,3,4 not only does the LfII strategy generate trajectories much closer to the target, but also the variance of these trajectories is lower. This suggests that better constructed latent space due to the larger variance of images makes it easier for the control policy to choose correct actions.

Table 10: The robustness of the policy is evaluated by performing perturbations in the action space with freq 0.1, with a magnitude twice as large as that used during training, in a direction tangential to the target. In the table below we observe the minimum distance from the different parts, under different policies, and the standard deviation in brackets, across 10 runs. We can see that using the LfII we improve both the minimum distance to the parts and the variance with which those are reached. The scores for Part 1 are similar, as the random initialisation helps to generalize.

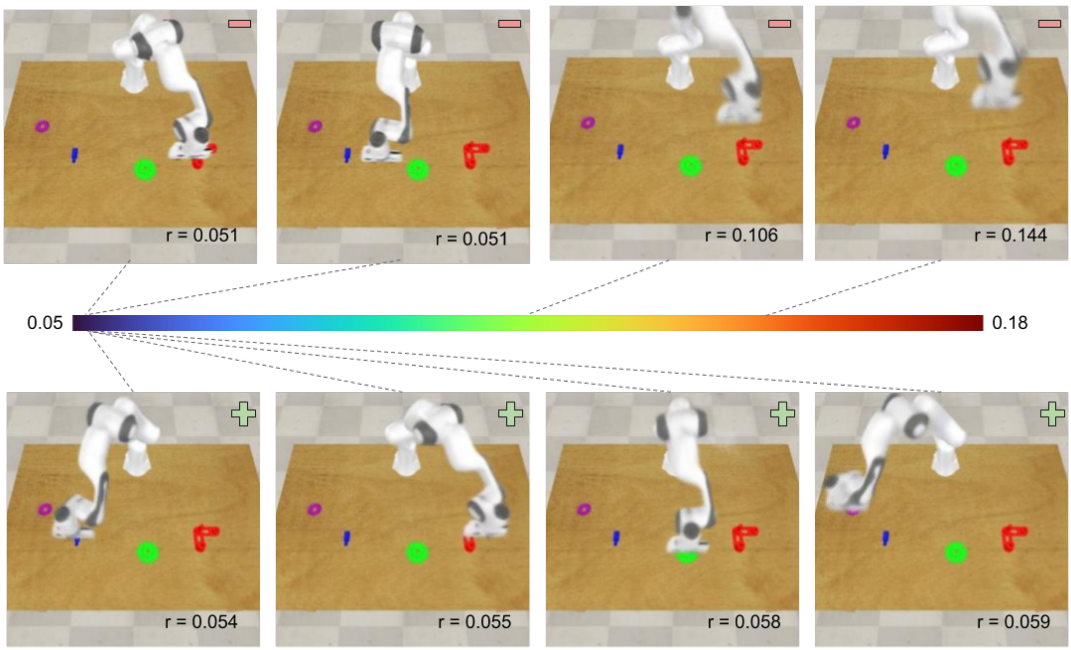
	Part 1	Part 2	Part 3	Part 4
none	*0.151 (0.096)*	0.143 (0.074)	0.237 (0.161)	0.299 (0.258)
conditional random	***0.134 (0.114)***	0.058 (0.028)	0.118 (0.103)	0.093 (0.182)
conditional LfII	*0.158 (0.092)*	**0.039 (0.022)**	**0.065 (0.041)**	**0.033 (0.043)**

In the case of the learned latent space, we need a generative model to visually recreate points within it. Figure 38 shows a variety of samples from different locations. It shows annotated samples from the visual Q&A with a human about the salience of different states. A base assumption that allows for the nodes to be annotated relatively quickly (200 images for under 5 minutes) is that smaller nodes, through which all of the trajectories pass, are inherently salient. Thus, ordering the nodes speeds up the

process. In the case of conditional LfII based interventions in a static environment, as the volume of states that the node covers increases, the probability of the node being salient decreases. However, this does not hold for the case where it is dynamic or no interventions are performed, as the majority of nodes have limited volume.

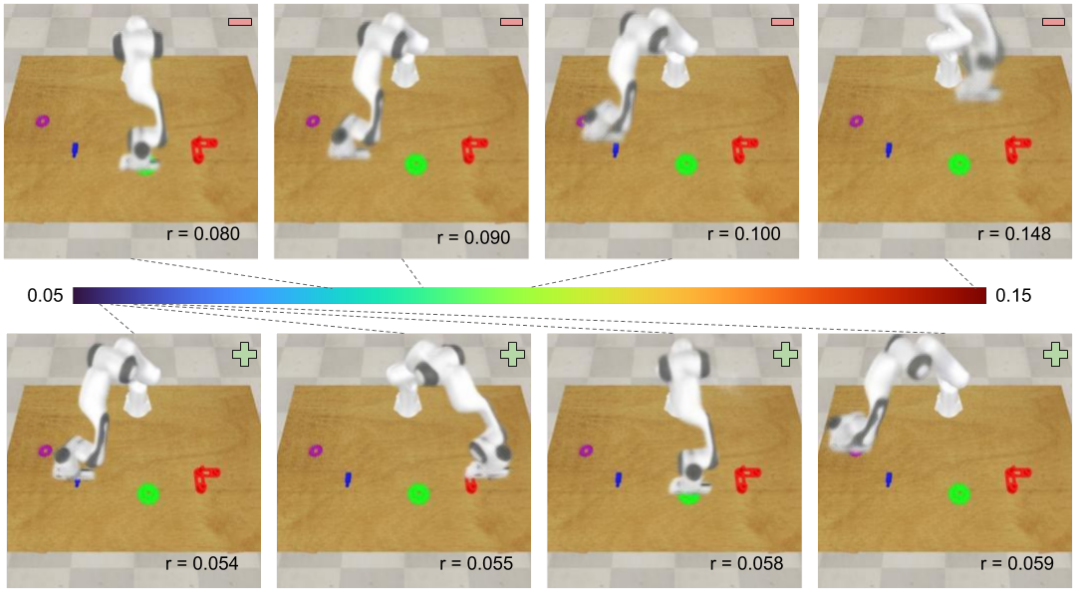
Figure 38b shows correctly identified salient locations with a dynamic scene. Increasing the complexity by adding moving objects, which are a target for the algorithm, still allows us to extract salient locations within the latent space.

In both cases, this method allows trajectories in the latent space of the problem to be clustered and salient points for the structure of the task to be identified. Future extension of this work would use the current node representation to better guide the directions of the intervention to jointly optimize the exploration and structure identification.

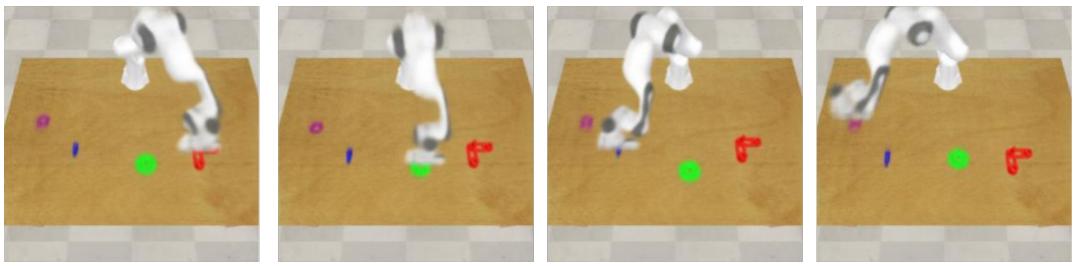


(a) No Intervention - static scene

Figure 37: Reconstructed images based on the center of clusters with different radii. This shows a static no intervention scene.



(a) Conditional LfII - static scene



(b) Conditional LfII - dynamic scene

Figure 38: Reconstructed images based on the center of clusters with different radii. In the static LfII case, there is a clear separation between the salient regions and in-between nodes after the first 25% of nodes. In the dynamic scene this assumption is less strong, but useful to identify salient nodes.

5.6 Limitations of LfII

In the process of finding task structure, there is an inherent limit on the type and scale of the intervention. It needs to balance (1) policy exploration, and (2) human willingness to work with the system. Larger interventions would improve the exploration but would be spending 'emotional will' on working with a system that is resisting the demonstration. This can be better balanced in case the latent space is better explored independently through standard RL techniques around violation of expectations.

In the current exploration strategy, only the policy performance is taken into account, and an improvement to the solution would jointly perform the exploration and intervention generation from the potential MDP graph.

5.7 Conclusion

We present a method for Learning from Inverse Intervention (LfII), in which a robot augments the expert demonstrations by intervening in search for less explored states. We show that this method is particularly well suited to sequential tasks where demonstrations might have had limited variability. In the process of altering the trajectory, the robot obtains useful information about the underlying structure of the problem. As a result, the learned policies have increased robustness and larger safe sets of operation. Similarly, using intervention strategies or through a form of visual Q&A, we evaluate areas of the state space for salience.

6

Conclusion

6.1 Key Ideas

With the transformation towards autonomous on-demand manufacturing, there is a need for a system that bridges the gap between rigid engineered solutions and versatile learning-based adaptations.

Some of the requirements this imposes is around fast adaptation that solves multi-step processes. This dissertation has presented methods that combine Learning from Demonstration — a tool to define and specify the required task, with hierarchical hybrid control strategies, permitting the use of diverse policies for robot control. It shows that this approach provides a flexible method to tackle long-horizon tasks. Further, it illustrates that examining the control strategy and the demonstrations themselves provide innate information about the structure of the problem. Creating a generative surrogate model of the control policy gives the ability to perform causal analysis. This provides the ability to extract agent interactions with symbols in the world by hypothesising different configurations. Furthermore, static demonstrations, especially of long-horizon tasks, are prone to following a single mode/manner within the data. This work shows that allowing the robot to interact with the expert during the demonstration can provide increased robustness and highlight possible salient parts of the task.

6.1.1 *Sequentially composing policies*

The temporal abstraction of the control strategy through a hierarchical structure is a common method to decrease the length of the actions an agent needs to make to provide some useful 'work'.

Formulating the task learning process as an RL problem is not always feasible with high-powered or -valued platforms. Additionally, it usually enforces the lower level policies to be represented in the same paradigm as the whole system. This work uses a library of diverse controllers, with a few demonstrations as a guide on the right sequence to complete the task.

The combination of a dynamics model with an estimator provides a powerful way to evaluate how well each controller would progress the current state towards the goal of the task. Independently each policy is not constrained on the type of input/output space and provides the needed flexibility. In return, this allows for the hierarchical policy to be represented analytically based on future state estimation.

The framework shows robustness to noise in the dynamic models, as well as the Goal Score Estimator models. It is demonstrated on a real-world problem of a gear assembly task with the PR2 robot.

6.1.2 *Policy and Task Structure*

Increasing the diversity of policies has the advantage of allowing the possible variations of how to complete a task with the method above to expand. The variation of policies can be seen as a slight alteration of the specifications of the demonstrators.

Using causal analysis one can identify that the specifications can differ causally in regards to known symbols. This is possible by learning a surrogate model, which can be investigated through evaluating counterfactual scenarios. Additionally, it is feasible to learn a parameterization of these interactions in simple scenarios. The generative surrogate model has further advantages that allow a proposed policy to be interpolated between different specifications — e.g. making a 'delicate' trajectory faster, as well as finding multiple bundles of possible solutions.

This post-data collection analysis is useful to gain insight into the structure of the policy itself. However, when the robot interacts with the demonstrator, as in the case of LfII, the augmented trajectories are informative in regards to finding a better approximation of the true task policy. The robot can intervene in situations where the demonstration is not informative, pushing the data collection process to regions that can better differentiate between the overparameterized policies [37]. During this process, the agent can build an MDP of possible salient locations for the task — natural sub-problem transitions. To confirm these salience clusters, the robot can either perform a trajectory that attempts to complete the task by excluding a particular region of the task space or alternatively, by using the generative model to create a visual representation of the state and obtain an answer through Visual Q&A with the expert.

6.2 Future Work

6.2.1 *Structure embedding and transfer*

One of the fundamental problems in Artificial Intelligence (AI) is around the embedding of structure or priors within the system. In the current work, there is a strict separation between the prior knowledge in the form of a causal graph and the demonstration bias.

However, a powerful system for demonstrations would come through embedding common statistical similarities — *"common sense reasoning"* — that can be extended and transferred to new situations.

It is common knowledge that e.g. people in high visibility vests behave in a structurally different model than ones without, albeit sharing some common plans. Or similarly that a Philip's head screw would behave comparably to a slot screw or even a Torx one. The commonality in one level of the knowledge hierarchy should provide a prior to its common higher and lower levels. The existence of a set of screws on a plate would be sufficient to (1) bias the possible activation policies, given the robot is holding a screwdriver. Also (2) in the case of interactive learning that it provides

little information gain to intervene on the demonstration to "poke" the metal plate on which the screws are mounted.

The development of the CycL knowledge base [59] has shown that pure symbol manipulation is insufficient to capture the full representation of a domain.

This combination of symbolic planning and manipulation or additional natural language instructions do not share a common representation with causal models or a neural detection pipeline. To reach the next step of performance, there need to be improvements in the addition of such reasoning capabilities to improve the overall system safety and ability to work under uncertainty.

6.2.2 *Temporal memory tasks*

Sequential compositionality is a powerful tool that allows for independent algorithms to be sequenced to solve a wider-reaching task. Working beyond the scope of each individual policy requires a general structure that implicitly or explicitly embeds the historic actions in the current state of the world.

To increase the classes of problems that the current method can be applied to, there needs to be a memory mechanism that stores the temporal sequence of actions that have been executed. Embedding a Neural Turing Machine [58] can be used to store and retrieve past actions, and Graph Neural Networks [125] to reason with additional constraints of time and dynamics for the future execution plan.

6.3 Concluding Remarks

In conclusion, this dissertation presents work on embedding different forms of inductive biases to solve temporally extended tasks and elicit additional structure within the task execution.

First, it focused on sequential policy composition in a hierarchical framework by relying on expert demonstrations. This increased the diversity of possible policies used to solve a real-world assembly problem.

Further, the demonstrator's specifications can be extracted by performing causal analysis on a learned surrogate model, allowing for increasing the policy variation by interpolation. It also highlights that these specifications can be parameterized in relation to the detected symbols.

Lastly, this work shows that interventions between the demonstrator and the agent in the form of a *"physical dialogue"* increase the overall performance and robustness of the system. It presents two methods for investigating the salience of regions within the task — through intervening during the demonstration or by using the generative model in visual Q&A.

Bibliography

[1] Pieter Abbeel, Adam Coates and Andrew Y Ng. 'Autonomous helicopter aerobatics through apprenticeship learning'. In: *The International Journal of Robotics Research* 29.13 (2010), pp. 1608–1639.

[2] P. Andonov, A. Savchenko, P. Rumschinski, S. Streif and R. Findeisen. 'Controller Verification and Parametrization Subject to Quantitative and Qualitative Requirements'. In: *IFAC-PapersOnLine* 48.8 (2015), pp. 1174 –1179.

[3] Marcin Andrychowicz, Filip Wolski, Alex Ray, Jonas Schneider, Rachel Fong, Peter Welinder, Bob McGrew, Josh Tobin, Pieter Abbeel and Wojciech Zaremba. 'Hindsight experience replay'. In: *Advances in Neural Information Processing Systems*. 2017, pp. 5048–5058.

[4] Daniel Angelov, Yordan Hristov, Michael Burke and Subramanian Ramamoorthy. 'Composing Diverse Policies for Temporally Extended Tasks'. In: *IEEE Robotics and Automation Letters* 5.2 (2020), pp. 2658–2665.

[5] Daniel Angelov, Yordan Hristov and Subramanian Ramamoorthy. 'DynoPlan: Combining Motion Planning and Deep Neural Network based Controllers for Safe HRL'. In: *The Multi-disciplinary Conference on Reinforcement Learning and Decision Making (RLDM)* (2019).

[6] Daniel Angelov, Yordan Hristov and Subramanian Ramamoorthy. 'Using Causal Analysis to Learn Specifications from Task Demonstrations'. In: *Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems*. AAMAS '19. Montreal QC, Canada: International Foundation for Autonomous Agents and Multiagent Systems, 2019, pp. 1341–1349. ISBN: 978-1-4503-6309-9. URL: http://dl.acm.org/citation.cfm?id=3306127.3331841.

[7] Daniel Angelov, Yordan Hristov and Subramanian Ramamoorthy. 'From demonstrations to task-space specifications. Using causal analysis to extract rule parameterization from demonstrations'. In: *Autonomous Agents and Multi-Agent Systems* 34.45 (2020). DOI: 10.1007/s10458-020-09471-w.

[8] Daniel Angelov and Subramanian Ramamoorthy. 'Learning from demonstration of trajectory preferences through causal modeling and inference'. In: *Robotics: Science and Systems Workshop on Causal Imitation in Robotics (CIR 2018)*. Pittsburgh, Pennsylvania, 2018.

[9] Brenna D. Argall, Sonia Chernova, Manuela Veloso and Brett Browning. 'A survey of robot learning from demonstration'. In: *Robotics and Autonomous Systems* 57.5 (2009), pp. 469 –483. ISSN: 0921-8890.

[10] Saurabh Arora and Prashant Doshi. 'A survey of inverse reinforcement learning: Challenges, methods and progress'. In: *arXiv preprint arXiv:1806.06877* (2018).

[11] Christopher G Atkeson and Stefan Schaal. 'Robot learning from demonstration'. In: *International Conference on Machine Learning*. Vol. 97. Citeseer. 1997, pp. 12–20.

[12] Jason Baldridge and Miles Osborne. 'Active learning and the total cost of annotation'. In: *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing*. 2004, pp. 9–16.

[13] Andrew G Barto and Sridhar Mahadevan. 'Recent advances in hierarchical reinforcement learning'. In: *Discrete event dynamic systems* 13.1-2 (2003), pp. 41–77.

[14] Rodrigo Benenson, Stefan Popov and Vittorio Ferrari. 'Large-scale interactive object segmentation with human annotators'. In: *CVPR*. 2019.

[15] Dimitri P Bertsekas. *Constrained optimization and Lagrange multiplier methods*. Academic press, 2014.

[16] Lucas Beyer, Olivier J. Henaff, Alexander Kolesnikov, Xiaohua Zhai and Aaron van den Oord. 'Are we done with ImageNet?' In: *arXiv preprint arXiv:2002.05709* (2020).

[17] Arjun Nitin Bhagoji, Daniel Cullina, Chawin Sitawarin and Prateek Mittal. 'Enhancing robustness of machine learning systems via data transformations'. In: *2018 52nd Annual Conference on Information Sciences and Systems (CISS)*. IEEE. 2018, pp. 1–5.

[18] Mariusz Bojarski, Davide Del Testa, Daniel Dworakowski, Bernhard Firner, Beat Flepp, Prasoon Goyal, Lawrence D Jackel, Mathew Monfort, Urs Muller, Jiakai Zhang et al. 'End to end learning for self-driving cars'. In: *arXiv preprint arXiv:1604.07316* (2016).

[19] Alejandro Bordallo, Fabio Previtali, Nantas Nardelli and Subramanian Ramamoorthy. 'Counterfactual reasoning about intent for interactive navigation in dynamic environments'. In: *Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ International Conference on*. IEEE. 2015, pp. 2943–2950.

[20] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang and Wojciech Zaremba. 'Openai gym'. In: *arXiv preprint arXiv:1606.01540* (2016).

[21] Daniel S. Brown and Scott Niekum. *Machine Teaching for Inverse Reinforcement Learning: Algorithms and Applications*. 2018. arXiv: 1805.07687 [cs.LG].

[22] Daniel Brown, Wonjoon Goo, Prabhat Nagarajan and Scott Niekum. 'Extrapolating Beyond Suboptimal Demonstrations via Inverse Reinforcement Learning from Observations'. In: *International Conference on Machine Learning*. 2019.

[23] Yuri Burda, Harri Edwards, Deepak Pathak, Amos Storkey, Trevor Darrell and Alexei A Efros. 'Large-scale study of curiosity-driven learning'. In: *arXiv preprint arXiv:1808.04355* (2018).

[24] Michael Burke, Yordan Hristov and Subramanian Ramamoorthy. 'Hybrid system identification using switching density networks'. In: *Conference on Robot Learning (CoRL)*. 2019.

[25] Michael Burke, Svetlin Penkov and Subramanian Ramamoorthy. 'From Explanation to Synthesis: Compositional Program Induction for Learning From Demonstration'. In: *Robotics: Science and Systems (R:SS)* (2019).

[26] Robert R Burridge, Alfred A Rizzi and Daniel E Koditschek. 'Sequential composition of dynamically dexterous robot behaviors'. In: *The International Journal of Robotics Research* 18.6 (1999), pp. 534–555.

[27] Richard H Byrd, Peihuang Lu, Jorge Nocedal and Ciyou Zhu. 'A limited memory algorithm for bound constrained optimization'. In: *SIAM Journal on Scientific Computing* 16.5 (1995), pp. 1190–1208.

[28] Daniel Cabrera, Jonathan F Thomas, Jeffrey L Wiswell, James M Walston, Joel R Anderson, Erik P Hess and M Fernanda Bellolio. 'Accuracy of 'my gut feeling:'comparing system 1 to system 2 decision-making for acuity prediction, disposition and diagnosis in an academic emergency department'. In: *Western Journal of Emergency Medicine* 16.5 (2015), p. 653.

[29] Satyan Chandra, In Jae Chung, Adnan Esmail, Matthew Blum and Rishabh Bhandari. *Wiring system architecture*. US Patent App. 16/231,314. 2019.

[30] T. Q. Chen, X. Li, R. Grosse and D. Duvenaud. 'Isolating Sources of Disentanglement in Variational Autoencoders'. In: *ArXiv e-prints* (Feb. 2018). arXiv: 1802.04942 [cs.LG].

[31] Xi Chen, Yan Duan, Rein Houthooft, John Schulman, Ilya Sutskever and Pieter Abbeel. 'Infogan: Interpretable representation learning by information maximizing generative adversarial nets'. In: *Advances in neural information processing systems*. 2016, pp. 2172–2180.

[32] Sonia Chernova and Manuela Veloso. 'Confidence-based Policy Learning from Demonstration Using Gaussian Mixture Models'. In: *Proceedings of the 6th International Joint Conference on Autonomous Agents and Multiagent Systems*. AAMAS '07. Honolulu, Hawaii: ACM, 2007, 233:1–233:8. ISBN: 978-81-904262-7-5.

[33] Sonia Chernova and Manuela Veloso. 'Confidence-based policy learning from demonstration using gaussian mixture models'. In: *Proceedings of the 6th international joint conference on Autonomous agents and multiagent systems*. 2007, pp. 1–8.

[34] William S Cleveland and Clive Loader. 'Smoothing by local regression: Principles and methods'. In: *Statistical theory and computational aspects of smoothing*. Springer, 1996, pp. 10–49.

[35] Felipe Codevilla, Eder Santana, Antonio M López and Adrien Gaidon. 'Exploring the Limitations of Behavior Cloning for Autonomous Driving'. In: *arXiv preprint arXiv:1904.08980* (2019).

[36] Gabriella Contardo, Ludovic Denoyer and Thierry Artières. 'A meta-learning approach to one-step active learning'. In: *arXiv preprint arXiv:1706.08334* (2017).

[37] Alexander D'Amour, Katherine Heller, Dan Moldovan, Ben Adlam, Babak Alipanahi, Alex Beutel, Christina Chen, Jonathan Deaton, Jacob Eisenstein, Matthew D Hoffman et al. 'Underspecification Presents Challenges for Credibility in Modern Machine Learning'. In: *arXiv preprint arXiv:2011.03395* (2020).

[38] Navneet Dalal and Bill Triggs. 'Histograms of oriented gradients for human detection'. In: *2005 IEEE computer society conference on computer vision and pattern recognition (CVPR'05)*. Vol. 1. IEEE. 2005, pp. 886–893.

[39] Shubhomoy Das, Md Rakibul Islam, Nitthilan Kannappan Jayakodi and Janardhan Rao Doppa. 'Active Anomaly Detection via Ensembles: Insights, Algorithms, and Interpretability'. In: *arXiv:1901.08930* (2019).

[40] Shubhomoy Das, Weng-Keen Wong, Thomas Dietterich, Alan Fern and Andrew Emmott. 'Discovering Anomalies by Incorporating Feedback from an Expert'. In: *ACM Trans. Knowl. Discov. Data* 14.4 (June 2020). ISSN: 1556-4681. DOI: 10.1145/3396608. URL: https://doi.org/10.1145/3396608.

[41] Kerstin Dautenhahn and Chrystopher L Nehaniv. 'Sensory-motor primitives as a basis for imitation: Linking perception to action and biology to robotics'. In: *MA: MIT Press* (2002).

[42] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li and L. Fei-Fei. 'ImageNet: A Large-Scale Hierarchical Image Database'. In: *CVPR09*. 2009.

[43] E. Denton and V. Birodkar. 'Unsupervised Learning of Disentangled Representations from Video'. In: *ArXiv e-prints* (May 2017). arXiv: 1705.10915 [cs.LG].

[44] F. Doshi-Velez and B. Kim. 'Towards A Rigorous Science of Interpretable Machine Learning'. In: *ArXiv e-prints* (Feb. 2017). arXiv: 1702.08608 [stat.ML].

[45] Gary L Drescher. 'Made-up minds: a constructivist approach to artificial intelligence'. PhD thesis. Massachusetts Institute of Technology, 1989.

[46] Yilun Du and Karthik Narasimhan. 'Task-Agnostic Dynamics Priors for Deep Reinforcement Learning'. In: *arXiv preprint arXiv:1905.04819* (2019).

[47] Meng Fang, Yuan Li and Trevor Cohn. 'Learning how to active learn: A deep reinforcement learning approach'. In: *arXiv preprint arXiv:1708.02383* (2017).

[48] Richard E Fikes, Peter E Hart and Nils J Nilsson. 'Learning and executing generalized robot plans'. In: *Artificial intelligence* 3 (1972), pp. 251–288.

[49] Chelsea Finn, Xin Yu Tan, Yan Duan, Trevor Darrell, Sergey Levine and Pieter Abbeel. 'Learning Visual Feature Spaces for Robotic Manipulation with Deep Spatial Autoencoders'. In: *IEEE International Conference on Robotics and Automation (ICRA)* (2016).

[50] Carlos Florensa, Yan Duan and Pieter Abbeel. 'Stochastic neural networks for hierarchical reinforcement learning'. In: *arXiv preprint arXiv:1704.03012* (2017).

[51] Jakob Foerster, Gregory Farquhar, Triantafyllos Afouras, Nantas Nardelli and Shimon Whiteson. *Counterfactual Multi-Agent Policy Gradients*. 2017. arXiv: 1705.08926 [cs.AI].

[52] Lex Fridman, Daniel E Brown, Michael Glazer, William Angell, Spencer Dodd, Benedikt Jenik, Jack Terwilliger, Aleksandr Patsekin, Julia Kindelsberger, Li Ding et al. 'MIT advanced vehicle technology study: Large-scale naturalistic driving study of driver behavior and interaction with automation'. In: *IEEE Access* 7 (2019), pp. 102021–102038.

[53] Carlos E Garcia, David M Prett and Manfred Morari. 'Model predictive control: theory and practice—a survey'. In: *Automatica* 25.3 (1989), pp. 335–348.

[54] Carles Gelada, Saurabh Kumar, Jacob Buckman, Ofir Nachum and Marc G Bellemare. 'DeepMDP: Learning Continuous Latent Space Models for Representation Learning'. In: *arXiv preprint arXiv:1906.02736* (2019).

[55] Malik Ghallab, Dana Nau and Paolo Traverso. *Automated Planning: theory and practice*. Elsevier, 2004.

[56] Shromona Ghosh, Felix Berkenkamp, Gireeja Ranade, Shaz Qadeer and Ashish Kapoor. 'Verifying Controllers Against Adversarial Examples with Bayesian Optimization'. In: *CoRR* abs/1802.08678 (2018). arXiv: 1802.08678.

[57] Matthew Gombolay, Reed Jensen, Jessica Stigile, Sung-Hyun Son and Julie Shah. 'Apprenticeship scheduling: Learning to schedule from human experts'. In: AAAI Press/International Joint Conferences on Artificial Intelligence. 2016.

[58] Alex Graves, Greg Wayne and Ivo Danihelka. 'Neural turing machines'. In: *arXiv preprint arXiv:1410.5401* (2014).

[59] Russell Greiner and Douglas B Lenat. 'A Representation Language Language.' In: *AAAI*. Vol. 1. 1980, pp. 165–169.

[60] Daniel H Grollman and Odest Chadwicke Jenkins. 'Dogged learning for robots'. In: *Proceedings 2007 IEEE International Conference on Robotics and Automation*. IEEE. 2007, pp. 2483–2488.

[61] Abhishek Gupta, Vikash Kumar, Corey Lynch, Sergey Levine and Karol Hausman. *Relay Policy Learning: Solving Long-Horizon Tasks via Imitation and Reinforcement Learning*. 2019. arXiv: 1910.11956 [cs.LG].

[62] David Ha and Jurgen Schmidhuber. 'World Models'. In: *arXiv preprint arXiv:1803.10122* (2018).

[63] Danijar Hafner, Timothy Lillicrap, Ian Fischer, Ruben Villegas, David Ha, Honglak Lee and James Davidson. 'Learning latent dynamics for planning from pixels'. In: *arXiv preprint arXiv:1811.04551* (2018).

[64] Jean Harb, Pierre-Luc Bacon, Martin Klissarov and Doina Precup. 'When waiting is not an option: Learning options with a deliberation cost'. In: *Thirty-Second AAAI Conference on Artificial Intelligence*. 2018.

[65] M. Harradon, J. Druce and B. Ruttenberg. 'Causal Learning and Explanation of Deep Neural Networks via Autoencoded Activations'. In: *ArXiv e-prints* (Feb. 2018). arXiv: 1802.00541 [cs.AI].

[66] Anna Harutyunyan, Peter Vrancx, Pierre-Luc Bacon, Doina Precup and Ann Nowe. 'Learning with options that terminate off-policy'. In: *Thirty-Second AAAI Conference on Artificial Intelligence*. 2018.

[67] B. Hayes and B. Scassellati. 'Discovering task constraints through observation and active learning'. In: *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*. 2014, pp. 4442–4449.

[68] Danny Hernandez and Tom B Brown. 'Measuring the Algorithmic Efficiency of Neural Networks'. In: *arXiv preprint arXiv:2005.04305* (2020).

[69] I. Higgins, L. Matthey, A. Pal, C. Burgess, X. Glorot, M. Botvinick, S. Mohamed and A. Lerchner. 'beta-vae: Learning basic visual concepts with a constrained variational framework'. In: (2016).

[70] Irina Higgins, Nicolas Sonnerat, Loic Matthey, Arka Pal, Christopher P Burgess, Matko Bosnjak, Murray Shanahan, Matthew Botvinick, Demis Hassabis and Alexander Lerchner. 'Scan: Learning hierarchical compositional visual concepts'. In: *arXiv preprint arXiv:1707.03389* (2017).

[71] Irina Higgins, Nicolas Sonnerat, Loic Matthey, Arka Pal, Christopher P Burgess, Matko Bosnjak, Murray Shanahan, Matthew Botvinick, Demis Hassabis and Alexander Lerchner. 'SCAN: Learning Hierarchical Compositional Visual Concepts'. In: (2018).

[72] Jonathan Ho and Stefano Ermon. 'Generative adversarial imitation learning'. In: *Advances in neural information processing systems*. 2016, pp. 4565–4573.

[73] Yordan Hristov, Daniel Angelov, Michael Burke, Alex Lascarides and Subramanian Ramamoorthy. 'Disentangled Relational Representations for Explaining and Learning from Demonstration'. In: *Conference on Robot Learning (CoRL)*. 2019.

[74] Jessie Huang, Fa Wu, Doina Precup and Yang Cai. 'Learning safe policies with expert guidance'. In: *Advances in Neural Information Processing Systems*. 2018, pp. 9105–9114.

[75] Glenn A Iba. 'A heuristic approach to the discovery of macro-operators'. In: *Machine Learning* 3.4 (1989), pp. 285–317.

[76] Auke Jan Ijspeert, Jun Nakanishi, Heiko Hoffmann, Peter Pastor and Stefan Schaal. 'Dynamical movement primitives: learning attractor models for motor behaviors'. In: *Neural computation* 25.2 (2013), pp. 328–373.

[77] T. Inamura. 'Acquisition of probabilistic behavior decision model based on the interactive teaching method'. In: *Proc. 9th Int'l Conf. on Advanced Robotics* (1999), pp. 523–528. URL: https://ci.nii.ac.jp/naid/20000105704/en/.

[78] Inigo Iturrate, Esben Ostergaard, Martin Rytter and Thiusius Savarimuthu. 'Learning and correcting robot trajectory keypoints from a single demonstration'. In: Apr. 2017, pp. 52–59. DOI: 10.1109/ICCAR.2017.7942660.

[79] Maja J Matari'c. 'Sensory-Motor Primitives as a Basis for Imitation: Linking Perception to Action and Biology to Robotics'. In: *MIT Press, Cambridge, MA, USA* (Nov. 1999).

[80] Ashesh Jain, Shikhar Sharma, Thorsten Joachims and Ashutosh Saxena. 'Learning preferences for manipulation tasks from online coactive feedback'. In: *The International Journal of Robotics Research* 34.10 (2015), pp. 1296–1313.

[81] Ashesh Jain, Brian Wojcik, Thorsten Joachims and Ashutosh Saxena. 'Learning trajectory preferences for manipulators via iterative improvement'. In: *Advances in neural information processing systems*. 2013, pp. 575–583.

[82] M. J. Johnson, D. Duvenaud, A. B. Wiltschko, S. R. Datta and R. P. Adams. 'Composing graphical models with neural networks for structured representations and fast inference'. In: *ArXiv e-prints* (Mar. 2016). arXiv: 1603.06277 [stat.ML].

[83] Leslie Pack Kaelbling, Michael L Littman and Andrew W Moore. 'Reinforcement learning: A survey'. In: *Journal of artificial intelligence research* 4 (1996), pp. 237–285.

[84] H Jin Kim, Michael I Jordan, Shankar Sastry and Andrew Y Ng. 'Autonomous helicopter flight via reinforcement learning'. In: *Advances in neural information processing systems*. 2004, pp. 799–806.

[85] D. P Kingma and M. Welling. 'Auto-Encoding Variational Bayes'. In: *ArXiv e-prints* (Dec. 2013). arXiv: 1312.6114 [stat.ML].

[86] Ross A Knepper, Todd Layton, John Romanishin and Daniela Rus. 'Ikeabot: An autonomous multi-robot coordinated furniture assembly system'. In: *2013 IEEE International Conference on Robotics and Automation*. IEEE. 2013.

[87] Jens Kober, J Andrew Bagnell and Jan Peters. 'Reinforcement learning in robotics: A survey'. In: *The International Journal of Robotics Research* 32.11 (2013), pp. 1238–1274.

[88] Kai Henning Koch, Katja Mombaur and Philippe Soueres. 'Optimization-based walking generation for humanoid robot'. In: *IFAC Proceedings Volumes* 45.22 (2012), pp. 498–504.

[89] J Zico Kolter, Pieter Abbeel and Andrew Y Ng. 'Hierarchical apprenticeship learning with application to quadruped locomotion'. In: *Advances in Neural Information Processing Systems*. 2008, pp. 769–776.

[90] Richard E Korf. *Learning to solve problems by searching for macro-operators*. Tech. rep. Carnegie-Mellon University, Pittsburgh, Dept of Computer Science, 1983.

[91] Alex Krizhevsky, Ilya Sutskever and Geoffrey E Hinton. 'Imagenet classification with deep convolutional neural networks'. In: *Advances in neural information processing systems*. 2012, pp. 1097–1105.

[92] James J Kuffner and Steven M LaValle. 'RRT-connect: An efficient approach to single-query path planning'. In: *Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No. 00CH37065)*. Vol. 2. IEEE. 2000, pp. 995–1001.

[93] Alina Kuznetsova et al. 'The Open Images Dataset V4: Unified image classification, object detection, and visual relationship detection at scale'. In: *IJCV* (2020).

[94] Noah K Lee. 'Total Automation: The Possibility of Lights-Out Manufacturing in the Near Future'. In: *Missouri S&T's Peer to Peer* 2.1 (2018), p. 4.

[95] S. Levine, C. Finn, T. Darrell and P. Abbeel. 'End-to-End Training of Deep Visuomotor Policies'. In: *ArXiv e-prints* (Apr. 2015). arXiv: 1504.00702.

[96] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár and C Lawrence Zitnick. 'Microsoft coco: Common objects in context'. In: *European conference on computer vision*. Springer. 2014, pp. 740–755.

[97] David G Lowe. 'Distinctive image features from scale-invariant keypoints'. In: *International journal of computer vision* 60.2 (2004), pp. 91–110.

[98] Tomas Lozano-Perez, Matthew T. Mason and Russell H. Taylor. 'Automatic Synthesis of Fine-Motion Strategies for Robots'. In: *The International Journal of Robotics Research* 3.1 (1984), pp. 3–24.

[99] Wunderman Commerce Ltd. *The Amazon Prime Effect - Setting a new standard for customer loyalty*. 2018.

[100] Jianlan Luo, Eugen Solowjow, Chengtao Wen, Juan Aparicio Ojea, Alice M. Agogino, Aviv Tamar and Pieter Abbeel. *Reinforcement Learning on Variable Impedance Controller for High-Precision Robotic Assembly*. 2019. arXiv: 1903.01066 [cs.RO].

[101] Kevin M. Lynch and Matthew T. Mason. 'Dynamic Nonprehensile Manipulation: Controllability, Planning, and Experiments'. In: *The International Journal of Robotics Research* 18.1 (1999), pp. 64–92.

[102] Matthew T. Mason. 'Mechanics and Planning of Manipulator Pushing Operations'. In: *The International Journal of Robotics Research* 5.3 (1986), pp. 53–71. DOI: 10.1177/027836498600500303.

[103] Matthew Thomas Mason. 'Manipulator grasping and pushing operations'. In: (1982).

[104] Maja J Mataric. 'Reward functions for accelerated learning'. In: *Machine Learning Proceedings 1994*. Elsevier, 1994, pp. 181–189.

[105] Yasser Mohammad and Toyoaki Nishida. 'Tackling the Correspondence Problem'. In: *Active Media Technology*. Ed. by Tetsuya Yoshida, Gang Kou, Andrzej Skowron, Jiannong Cao, Hakim Hacid and Ning Zhong. Cham: Springer International Publishing, 2013, pp. 84–95.

[106] Stephen L Morgan and Christopher Winship. *Counterfactuals and causal inference*. Cambridge University Press, 2015.

[107] Michael E Mortenson. *Mathematics for computer graphics applications*. Industrial Press Inc., 1999.

[108] Matthew W Moskewicz, Conor F Madigan, Ying Zhao, Lintao Zhang and Sharad Malik. 'Chaff: Engineering an efficient SAT solver'. In: *Proceedings of the 38th annual Design Automation Conference*. ACM. 2001, pp. 530–535.

[109] Andrew Y Ng, Stuart J Russell et al. 'Algorithms for inverse reinforcement learning.' In: *International Conference on Machine Learning*. 2000, pp. 663–670.

[110] Christopher Null and Brian Caulfield. *Fade To Black The 1980s Vision of "lights-out" Manufacturing, Where Robots Do All the Work, Is a Dream No More.* 2003.

[111] Vincent Furnon Nikolaj van Omme Laurent Perron. *or-tools user's manual*. Tech. rep. Google, 2014.

[112] Christian Ott. *Cartesian Impedance Control of Redundant and Flexible-Joint Robots*. 1st ed. Springer Publishing Company, Incorporated, 2008. ISBN: 3540692533.

[113] Judea Pearl. *Causality: Models, Reasoning and Inference*. 2nd. New York, NY, USA: Cambridge University Press, 2009. ISBN: 052189560X, 9780521895606.

[114] Svetlin Penkov and Subramanian Ramamoorthy. 'Program Induction to Interpret Transition Systems'. In: (2017).

[115] J. Peters, D. Janzing and B. Scholkopf. *Elements of Causal Inference: Foundations and Learning Algorithms*. Cambridge, MA, USA: MIT Press, 2017.

[116] Hieu Pham, Melody Guan, Barret Zoph, Quoc Le and Jeff Dean. 'Efficient Neural Architecture Search via Parameters Sharing'. In: *International Conference on Machine Learning*. 2018, pp. 4095–4104.

[117] Doina Precup. 'Eligibility traces for off-policy policy evaluation'. In: *CS Department Faculty Publication Series* (2000).

[118] Matthew Purver, Jonathan Ginzburg and Patrick Healey. 'On the means for clarification in dialogue'. In: *Current and new directions in discourse and dialogue*. Springer, 2003, pp. 235–255.

[119] Alberto Rodriguez, Matthew T Mason and Steve Ferry. 'From caging to grasping'. In: *The International Journal of Robotics Research* 31.7 (2012), pp. 886–900. DOI: 10.1177/0278364912442972.

[120] M. Rojas-Carulla, M. Baroni and D. Lopez-Paz. 'Causal Discovery Using Proxy Variables'. In: *ArXiv preprint* (Feb. 2017). arXiv: 1702.07306 [stat.ML].

[121] Stéphane Ross, Geoffrey Gordon and Drew Bagnell. 'A reduction of imitation learning and structured prediction to no-regret online learning'. In: *Proceedings of the fourteenth international conference on artificial intelligence and statistics*. 2011, pp. 627–635.

[122] Olga Russakovsky et al. 'ImageNet Large Scale Visual Recognition Challenge'. In: *International Journal of Computer Vision (IJCV)* 115.3 (2015), pp. 211–252. DOI: 10.1007/s11263-015-0816-y.

[123] Eric L Sauser, Brenna D Argall, Giorgio Metta and Aude G Billard. 'Iterative learning of grasp adaptation through human corrections'. In: *Robotics and Autonomous Systems* 60.1 (2012), pp. 55–71.

[124] Nikolay Savinov, Anton Raichuk, Raphaël Marinier, Damien Vincent, Marc Pollefeys, Timothy Lillicrap and Sylvain Gelly. 'Episodic curiosity through reachability'. In: *arXiv preprint arXiv:1810.02274* (2018).

[125] Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner and Gabriele Monfardini. 'The graph neural network model'. In: *IEEE Transactions on Neural Networks* 20.1 (2008), pp. 61–80.

[126] Stefan Schaal. 'Learning from Demonstration'. In: *Proceedings of the 9th International Conference on Neural Information Processing Systems*. Denver, Colorado: MIT Press, 1996, pp. 1040–1046.

[127] Stefan Schaal. 'Dynamic movement primitives-a framework for motor control in humans and humanoid robotics'. In: *Adaptive motion of animals and machines*. Springer, 2006, pp. 261–280.

[128] K Schwab. 'The fourth industrial revolution by Klaus Schwab'. In: *Translated by KJ Song, Mega-study Corporation, Seoul* (2016).

[129] Ozan Sener and Silvio Savarese. 'Active learning for convolutional neural networks: A core-set approach'. In: *arXiv preprint arXiv:1708.00489* (2017).

[130] Pierre Sermanet, Corey Lynch, Yevgen Chebotar, Jasmine Hsu, Eric Jang, Stefan Schaal, Sergey Levine and Google Brain. 'Time-contrastive networks: Self-supervised learning from video'. In: *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2018, pp. 1134–1141.

[131] Burr Settles. *Active learning literature survey*. Tech. rep. University of Wisconsin-Madison Department of Computer Sciences, 2009.

[132] Paul Shaw, Vincent Furnon and Bruno De Backer. 'A constraint programming toolkit for local search'. In: *Optimization Software Class Libraries*. Springer, 2003, pp. 219–261.

[133] Yanyao Shen, Hyokun Yun, Zachary C Lipton, Yakov Kronrod and Animashree Anandkumar. 'Deep active learning for named entity recognition'. In: *arXiv preprint arXiv:1707.05928* (2017).

[134] Elaine Schaertl Short, Adam Allevato and Andrea L Thomaz. 'SAIL: simulation-informed active in-the-wild learning'. In: *2019 14th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*. IEEE. 2019, pp. 468–477.

[135] Karen Simonyan and Andrew Zisserman. 'Very Deep Convolutional Networks for Large-Scale Image Recognition'. In: *International Conference on Learning Representations*. 2015.

[136] Jonathan Spencer, Sanjiban Choudhury, Matthew Barnes, Matthew Schmittle, Mung Chiang, Peter Ramadge and Siddhartha Srinivasa. 'Learning from Interventions'. In: *Proceedings of the Robotics Science and Systems Conference* (2020).

[137] Martin Stolle and Doina Precup. 'Learning Options in Reinforcement Learning'. In: *Abstraction, Reformulation, and Approximation*. Ed. by Sven Koenig and Robert C. Holte. Berlin, Heidelberg: Springer Berlin Heidelberg, 2002, pp. 212–223. ISBN: 978-3-540-45622-3.

[138] Niko Sunderhauf, Oliver Brock, Walter Scheirer, Raia Hadsell, Dieter Fox, Jurgen Leitner, Ben Upcroft, Pieter Abbeel, Wolfram Burgard, Michael Milford et al. 'The limits and potentials of deep learning for robotics'. In: *The International Journal of Robotics Research* 37.4-5 (2018), pp. 405–420.

[139] Richard S Sutton, Doina Precup and Satinder Singh. 'Between MDPs and semi-MDPs: A framework for temporal abstraction in reinforcement learning'. In: *Artificial intelligence* 112.1-2 (1999), pp. 181–211.

[140] Christian Szegedy, Sergey Ioffe, Vincent Vanhoucke and Alexander A Alemi. 'Inception-v4, inception-ResNet and the impact of residual connections on learning'. In: *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*. 2017, pp. 4278–4284.

[141] Mingxing Tan and Quoc Le. 'EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks'. In: *International Conference on Machine Learning*. 2019, pp. 6105–6114.

[142] Russ Tedrake, Ian R. Manchester, Mark Tobenkin and John W. Roberts. 'LQR-trees: Feedback Motion Planning via Sums-of-Squares Verification'. In: *The International Journal of Robotics Research* 29.8 (2010), pp. 1038–1052.

[143] Stefanie Tellex, Ross Knepper, Adrian Li, Daniela Rus and Nicholas Roy. 'Asking for help using inverse semantics'. In: *Robotics: Science and Systems Foundation* (2014).

[144] Garrett Thomas, Melissa Chien, Aviv Tamar, Juan Aparicio Ojea and Pieter Abbeel. 'Learning Robotic Assembly from CAD'. In: *2018 IEEE International Conference on Robotics and Automation (ICRA)* (2018).

[145] A. Thomaz and C. Breazeal. 'Tutelage and socially guided robot learning'. In: *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (IEEE Cat. No.04CH37566)* 4 (2004), 3475–3480 vol.4.

[146] Frederik Träuble, Elliot Creager, Niki Kilbertus, Anirudh Goyal, Francesco Locatello, Bernhard Schölkopf and Stefan Bauer. 'Is Independence all you need? On the Generalization of Representations Learned from Correlated Data'. In: *arXiv preprint arXiv:2006.07886* (2020).

[147] Mel Vecerik, Oleg Sushkov, David Barker, Thomas Rothorl, Todd Hester and Jon Scholz. 'A practical approach to insertion with variable socket position using deep reinforcement learning'. In: *2019 International Conference on Robotics and Automation (ICRA)*. IEEE. 2019, pp. 754–760.

[148] Najdan Vukoviundefined, Marko Mitiundefined and Zoran Miljkoviundefined. 'Trajectory Learning and Reproduction for Differential Drive Mobile Robots Based on GMM/HMM and Dynamic Time Warping Using Learning from Demonstration Framework'. In: *Eng. Appl. Artif. Intell.* 45.C (Oct. 2015), pp. 388–404. ISSN: 0952-1976. DOI: 10.1016/j.engappai.2015.07.002. URL: https://doi.org/10.1016/j.engappai.2015.07.002.

[149] Baiyang Wang and Diego Klabjan. 'Regularization for unsupervised deep neural nets'. In: *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*. 2017, pp. 2681–2681.

[150] Emo Welzl. 'Smallest enclosing disks (balls and ellipsoids)'. In: *New Results and New Trends in Computer Science*. 1991.

[151] Jim Winkens, Rudy Bunel, Abhijit Guha Roy, Robert Stanforth, Vivek Natarajan, Joseph R Ledsam, Patricia MacWilliams, Pushmeet Kohli, Alan Karthikesalingam, Simon Kohl et al. 'Contrastive training for improved out-of-distribution detection'. In: *arXiv preprint arXiv:2007.05566* (2020).

[152] Christian Wirth, Riad Akrour, Gerhard Neumann and Johannes Furnkranz. 'A survey of preference-based reinforcement learning methods'. In: *The Journal of Machine Learning Research* 18.1 (2017), pp. 4945–4990.

[153] Yan Xu, Ran Jia, Lili Mou, Ge Li, Yunchuan Chen, Yangyang Lu and Zhi Jin. *Improved Relation Classification by Deep Recurrent Neural Networks with Data Augmentation*. 2016. arXiv: 1601.03651 [cs.CL].

[154] T. Yu, C. Finn, A. Xie, S. Dasari, T. Zhang, P. Abbeel and S. Levine. 'One-Shot Imitation from Observing Humans via Domain-Adaptive Meta-Learning'. In: *ArXiv e-prints* (Feb. 2018). arXiv: 1802.01557.

[155] Tianhe Yu, Chelsea Finn, Annie Xie, Sudeep Dasari, Tianhao Zhang, Pieter Abbeel and Sergey Levine. 'One-shot imitation from observing humans via domain-adaptive meta-learning'. In: *arXiv preprint arXiv:1802.01557* (2018).

[156] Andy Zeng, Shuran Song, Johnny Lee, Alberto Rodriguez and Thomas Funkhouser. 'TossingBot: Learning to Throw Arbitrary Objects with Residual Physics'. In: *Robotics: Science and Systems* (2019).

[157] Shao Zhifei and Er Meng Joo. 'A survey of inverse reinforcement learning techniques'. In: *International Journal of Intelligent Computing and Cybernetics* 5.3 (2012), pp. 293–311.

[158] Jia-Jie Zhu and José Bento. 'Generative adversarial active learning'. In: *arXiv preprint arXiv:1702.07956* (2017).