Learning to Solve Manipulation Tasks from Non-Expert Users in Human-Centered Environments

Nichola Abdo

Technische Fakultät
Albert-Ludwigs-Universität Freiburg

UNI FREIBURG

Dissertation zur Erlangung des akademischen Grades
Doktor der Naturwissenschaften

Betreuer: Prof. Dr. Wolfram Burgard

Learning to Solve Manipulation Tasks from Non-Expert Users in Human-Centered Environments

Nichola Abdo

Dissertation zur Erlangung des akademischen Grades Doktor der Naturwissenschaften
Technische Fakultät, Albert-Ludwigs-Universität Freiburg

Dekan	Prof. Dr. Oliver Paul
Erstgutachter	Prof. Dr. Wolfram Burgard
	Albert-Ludwigs-Universität Freiburg
Zweitgutachter	Prof. Dr. Gerhard Lakemeyer
	Rheinisch-Westfälische Technische Hochschule Aachen
Tag der Disputation	21. Juli 2017

Zusammenfassung

Ein wichtiges Ziel der Robotik-Forschung ist die Entwicklung intelligenter Service-Roboter, die uns verschiedenste Aufgaben in unserer alltäglichen Umgebung abnehmen können. Um das zu erreichen, müssen Roboter in der Lage sein, ein breites Spektrum von alltäglichen Gegenständen zu manipulieren und diese in gewünschte räumliche Beziehungen zu bringen, zum Beispiel um Objekte in Regale einzuordnen oder einen Tisch zum Abendessen zu decken. Um jedoch intelligent in häuslichen Umgebungen zu operieren, müssen Roboter mit mehreren Herausforderungen umgehen können. Zum Beispiel gibt es typischerweise mehrere gültige Möglichkeiten, alltägliche Aufgaben zu erledigen, die von der Umgebung und den persönlichen Präferenzen der Endnutzer abhängen. Darüber hinaus werden Roboter im Zusammenhang mit beliebigen Aufgaben ständig neuen Objekten begegnen. Solche Faktoren machen es für Experten unmöglich, einen Roboter mit ausreichendem Wissen auszustatten, um mit allen Situationen umzugehen. Stattdessen sollten Roboter in der Lage sein, ständig von ihren Nutzern zu lernen und Aufgaben zu erledigen, ohne dabei ständig nachfragen zu müssen.

In dieser Arbeit stellen wir mehrere Beiträge vor, die es Robotern erlauben, mit diesen Herausforderungen umzugehen, und Manipulationsfähigkeiten durch die Interaktion mit ihren Nutzern und Umgebungen zu erwerben, um wünschenswerte Konfigurationen von Objekten zu erreichen. Als erstes stellen wir einen neuartigen Ansatz vor, der auf Recommender-Systemen basiert und es einem Roboter erlaubt, aus der Beobachtung mehrerer Nutzer Muster für die Organisation von Objekten zu lernen und dadurch persönliche Präferenzen für Nutzer vorherzusagen. Dies ermöglicht einem Roboter, sein Verhalten an einen bestimmten Nutzer anzupassen, wenn er eine Aufgabe wie beispielsweise das Aufräumen von Objekten ausführt. Zweitens schlagen wir neuartige Techniken vor, um multimodale räumliche Bedingungen zu lernen, die notwendig sind, um eine neue Manipulationsaktion aus einer kleinen Anzahl von Demonstrationen zu modellieren und zu generalisieren. Wir stellen ein neuartiges Verfahren vor, das es Robotern ermöglicht, mit Hilfe dieser Modelle sequentielle Manipulationsaufgaben durch Beobachtungen eines Endnutzers zu lernen. Im Gegensatz zu bestehenden Paradigmen und Verfahren erlauben unsere Beiträge einem Roboter, Lösungen von Aufgaben zu improvisieren, indem er Aktionen sequenziert und erwünschte Ziele erreicht, ohne dass der Nutzer einen expliziten Zielzustand spezifizieren muss. Schließlich stellen wir einen neuen Ansatz für das Problem des lebenslangen Lernens von räumlichen Beziehungen vor. Unsere Lösung basiert auf dem Lernen von Distanzmetriken und ermöglicht einem

Roboter, seine bisherigen Kenntnisse über räumliche Beziehungen zwischen Objekten zu nutzen, um neue Beziehungen zu lernen und sie über Objekte verschiedener Formen und Größen zu verallgemeinern.

Wir haben unsere vorgestellten Ansätze mit realen Daten, die wir von verschiedenen Benutzern gesammelt haben, ausgiebig ausgewertet. Unsere Ergebnisse zeigen die Wirksamkeit unserer Techniken beim Lernen und Erreichen von wünschenswerten Objektkonfigurationen im Rahmen alltäglicher Manipulationsaufgaben. Darüberhinaus haben wir unsere Arbeit auf einem echten Roboter implementiert und getestet. Unsere Experimente demonstrieren die Anwendbarkeit unserer Systeme in realistischen Szenarien und zeigen, dass unsere Algorithmen es Robotern ermöglichen, verschiedene Manipulationsaufgaben erfolgreich zu lösen.

Abstract

A key goal of robotics research is to develop intelligent service robots that are capable of undertaking a variety of tasks in our everyday environments. To realize this, robots should be equipped with the capabilities to reason about and manipulate a rich spectrum of everyday objects to achieve desirable spatial relations using them, for example to organize objects on shelves or set a table for dinner.

However, to operate intelligently in domestic environments, robots should overcome several challenges. For example, there usually are several valid ways of attending to everyday chores depending on the environment and the preferences of the end-users. Moreover, robots will constantly encounter novel objects while attending to arbitrary tasks. Such factors render it infeasible for an expert to pre-program a robot with sufficient knowledge to handle all situations. Instead, robots should be able to continuously learn from their users and attend to tasks without having to constantly query them.

In this thesis, we present several contributions that address these challenges and enable robots to acquire manipulation skills to achieve desirable object configurations by interacting with their users and environments. Firstly, we present a novel approach based on recommender system theory that allows a robot to predict user preferences by extracting patterns of organizing objects across several users. This enables the robot to tailor its behavior to a specific user when solving a task such as organizing objects in containers. Secondly, we propose novel techniques to infer the multi-modal spatial constraints necessary to model and generalize a new manipulation action using a small number of demonstrations. We build on these techniques and propose a novel approach to learning sequential manipulation tasks from non-expert users. Our contribution goes beyond existing paradigms by allowing the robot to sequence actions and improvise feasible task solutions without requiring the user to provide an explicit goal state for planning. Finally, we tackle the problem of lifelong learning of spatial relations and generalizing them to objects of various shapes and sizes. For this, we propose a novel approach based on distance metric learning to allow a robot to leverage its previous knowledge of relations to efficiently learn and generalize new ones.

We extensively evaluate our proposed approaches using real-world data acquired from different users, and demonstrate the effectiveness of our techniques in learning and reproducing desirable object configurations in the context of everyday manipulation tasks. Additionally, we evaluate our implementation on a real robot and demonstrate the applicability of our solutions in realistic scenarios.

Acknowledgements

This thesis would not have been possible without the support, encouragement, and guidance of many people, whom I would like to dearly thank.

First of all, I would like to thank Wolfram Burgard for giving me the unique chance to pursue a PhD at his lab and for providing the fun and collaborative environment needed to work on various projects and explore new research ideas. I greatly appreciate his feedback and guidance throughout my time at the Autonomous Intelligent Systems group (AIS), his constant encouragement to solve hard problems, our wonderful discussions, and the chance of attending so many conferences all over the world.

I had the privilege of collaborating with and learning from many exceptional researchers at AIS. I thank Cyrill Stachniss for being a great mentor since I joined AIS as a masters student and for his meticulous feedback on scientific writing. I owe a great debt of gratitude to Luciano Spinello for his supervision and encouragement and for inspiring me to pursue novel ideas. I thank Gian Diego Tipaldi for always taking the time to brainstorm on the white board and for reminding me to focus on what matters in the long run. A huge thank you goes to Christian Dornhege for always taking the time to give feedback on both high-level ideas and low-level implementation details.

I would also like to thank the Hybris project family in general and my partners in the C1 project in particular. I thank Gerhard Lakemeyer for his efforts in coordinating the research unit, his hospitality during my stays in Aachen, and for agreeing to be a second examiner of this thesis. I would also like to thank Bernhard Nebel for his feedback on my work since my masters studies. I thank him and Matthias Teschner for taking the time to be on my thesis committee. Special thanks go to Tim Niemueller and Andreas Hertle, my partners in the "dream team of competence." I have greatly enjoyed the long days and nights we spent testing our work on the PR2 robot. I also thank Christoph Schwering for our fruitful collaboration.

During my PhD, I had the chance to collaborate with many exceptional researchers from around the world. I would like to thank Jeff Trinkle and Jochen Renz for the interesting discussions and constructive feedback. Special thanks go to Xiaoyu Ge (Gary) for all his efforts during and after his visit to AIS. I would also like to thank Marc Toussaint and Luc De Raedt for the valuable feedback on my work. Special thanks go to Thomas Keller for the constructive feedback and precious insights.

This work would not have been possible without the support and friendship of everyone in my AIS family. Special thanks go to Barbara Frank for continuing to be a great

friend and mentor despite me almost destroying Zora the robot as a masters student. I thank Jürgen Sturm, Felix Endres, Markus Spies, Bastian Steder, Rainer Kümmerle, Maximilian Beinhofer, Benjamin Suger, Jörg Röwekämper, and Michael Ruhnke for their friendship and advice over the years. I thank Henrik Kretzschmar for always taking the time to provide valuable feedback. I thank Jürgen Hess for his help with the PR2 robot and for all the robot demos we did together. I would like to thank Tayyab Naseer, Ayush Dewan, Martina Deturres, Stefano Di Lucia, Tim Caselitz, Freya Fleckenstein, Wera Winterhalter, Christian Dornhege, and Andreas Lars Wachaja for being there through good times and deadlines. Special thanks go to Pratik Agarwal for all the rants we had over the years, to Abhinav Valada for organizing the best parties in Freiburg, to Federico Boniardi for his continuous encouragement (not) to give up, and to Gabriel Oliveira for the lovely distractions. A big shout out goes to Tim Welschehold for taking care of the PR2 robot and to Andreas Eitel for our fun collaborations and for taking thousands of pictures of groceries with me. Many thanks go to Michael Keser and Susanne Bourjaillat for their support with all sorts of technical and administrative issues over the years.

Special thanks go to my officemates over the years for the fun discussions and tolerating my mood while writing the thesis: Bettina Schug, Christoph Sprunk, Markus Spies, Noha Radwan, Mladen Mazuran, Florian Krämer, Daniel Büscher, Jan Wülfing, and Manuel Watter. A big thank you goes to Christoph Sprunk for always going the extra mile in providing help, for introducing me to the wonderful world of TikZ plots, and for the great time we shared as Google interns. I thank my great friend Noha Radwan for keeping the office stocked with snacks and for the insights into the Arabic language. Special thanks go to the one and only Mladen Mazuran for his authenticity and for all the hours we spent discussing C++ and the dark corners of the English language.

I would like to thank all the bright students I had the privilege to co-supervise and collaborate with at AIS. Specifically, I thank Oier Mees for his hard work as a student and colleague, and for his friendship. A huge shout out goes to Philipp Jund for his efforts and dedication and for always going the extra mile.

I am very grateful to Christian Dornhege, Barbara Frank, Tim Welschehold, Andreas Hertle, Oier Mees, Philipp Jund, Andreas Lars Wachaja, Cyrill Stachniss, and Thomas Keller for proof-reading earlier versions of this thesis and providing valuable feedback. Special thanks go to Christian Dornhege, Barbara Frank, Noha Radwan, Freya Fleckenstein, Wera Winterhalter, and Yara Shaban for their encouragement during difficult times. This thesis would not have been possible without you.

I am grateful to my friends around the world for always being there through ups and downs. A special shout out goes to Spiro Elissa, Ahmad Abdel-Yaman, Yara Shaban, Elie Nino, Kareem Naouri, Lina Ejeilat, Rania Ejeilat, Mais Qsous, Mohammed Barghouthi, Fouad Mardini, Farid Awad, Ramez Bahu, Ala Al-Far, Nico Offeddu, Vivian Castano Fresen, Julian Kröger, Deepak Sanan, Thomas Huber, and the Dorsel family.

Above all, I thank my family in Jordan for their unconditional love and support.

Dedicated to the loving memory of my father
Victor Abdo, 1947-2006

This work has been supported by the German Research Foundation (DFG) under research unit FOR 1513 (HYBRIS) and grant number EXC 1086. Their support is gratefully acknowledged.

Contents

1. Introduction

Robots are changing our lives. Automated machines and manipulators have revolutionized our industries and boosted our economies by accurately and tirelessly welding, assembling, and mass-producing numerous products we use in our daily lives, from our smartphones to our cars. Our cars are becoming increasingly more capable of driving themselves, promising a new era of safe transportation in which we waste less time in traffic and reduce loss of life due to accidents caused by human errors. Additionally, robots continue to play a large role in the health sector by assisting doctors in complicated medical and surgical procedures.

To a certain extent, robots have also made their way to our daily lives at home or work. The most ubiquitous example of this are vacuum-cleaning robots, currently navigating the apartments of millions of people who rely on them to keep their floors clean (iRobot Corp, 2017). Robots are also assisting warehouse workers in transporting heavy items (Fetch Robotics, 2017) and have even started delivering pizzas to our doorsteps (Domino's Pizza Inc., 2017; Marble, 2017) and tooth paste to our hotel rooms (Savioke, 2017).

Indeed, recent advancements in computer vision and perception, fueled by rapid progress in deep learning techniques and an increasing availability of datasets, have brought us closer than ever to robots that are capable of perceiving and understanding everyday, unstructured environments (Lai et al., 2011a; Zhou et al., 2014; He et al., 2015; Eitel et al., 2015). Nowadays, robots are also capable of constructing detailed (semantic) maps of their environments and localizing themselves in them while safely negotiating various terrains and structures such as stairs (Thrun et al., 2005; Nüchter and Hertzberg, 2008; Henry et al., 2010; Rusu, 2010; Oßwald et al., 2011; Hornung et al., 2013; Endres et al., 2014; Feng et al., 2015). Moreover, advancements in motion planning, grasping, and reinforcement learning have enabled robots to safely and reliably reach for, grasp, and manipulate everyday objects (Ratliff et al., 2009; Dogar and Srinivasa, 2011; Sucan et al., 2012; Prattichizzo and Trinkle, 2016; Kavraki and LaValle, 2016; Englert and Toussaint, 2016; Levine et al., 2016).

Yet one domain in which robotics has not lived up to our aspirations so far is personal, assistive robotics. While robots continue to explore the surface of Mars and the depths of oceans, they have not yet found their way to our kitchens and living rooms beyond specialized tasks such as vacuum-cleaning or as social companions (Jibo, 2017). For decades, we have envisioned service robots that operate amongst us, improving the

quality of our lives by assisting the elderly and disabled with their daily needs, and undertaking a wide variety of mundane chores such as tidying up and cleaning. Whereas robots continue to get better at folding and unfolding laundry (Miller et al., 2012; Doumanoglou et al., 2014), cleaning (Hess et al., 2012), loading the dishwasher (Saxena et al., 2008), operating appliances and cooking (Beetz et al., 2011; Sung et al., 2015), and assisting the disabled with basic tasks such as shaving or drinking (Chen et al., 2012; Schröer et al., 2015), such robots can typically handle a limited range of pre-specified situations. There exist numerous challenges that have prevented the majority of such robots from leaving the realms of research laboratories and science fiction.

To solve a complex task such as tidying up an apartment, robots require the means to combine their sensory data, manipulation skills, and navigation modules to compute feasible and safe plans based on the specific environment they are operating in. Today's high-level controllers address this by reasoning on both a high (symbolic) and a low (geometric) level simultaneously. Integrated task and motion planners (Dornhege et al., 2009; Plaku and Hager, 2010; Kaelbling and Lozano-Pérez, 2011, 2013; Srivastava et al., 2014; Gharbi et al., 2015) have made it possible to efficiently compute high-level plans based on symbolic state representations that abstract the continuous nature of the world (e.g., `pick up cup`, `go to kitchen`, `open cupboard`, etc.), while ensuring the feasibility of such plans on the geometric level (e.g., can the robot reach the cup or go through the kitchen door?). Furthermore, knowledge-based, high-level controllers enable robots to reason about the information they need to successfully solve such a task (e.g., is this the correct cup?), and to handle sensor noise and unexpected events (Petrick and Bacchus, 2002; Rockel et al., 2013; Lemaignan and Alami, 2013; Tenorth and Beetz, 2013; Schwering et al., 2016; Gierse et al., 2016; Hofmann et al., 2016).

Despite being able to deal with complex domains, the vast majority of the above planning paradigms share the same strong assumption: the user or an expert provides the robot with a concrete, explicit goal to achieve before the robot can compute a plan for solving the task. The most common form of such a goal is a logical expression specifying which conditions should be satisfied (or negated) for the task to be successfully solved. For example, such a goal expression can describe a specific cup that the robot should fetch from the kitchen and place on the living room table, or the fact that all cups should be moved back to the kitchen and placed in the dishwasher. Similarly, a robot with the task of assembling a new chair should be provided with an exact geometric description of what the chair should look like.

Whereas specifying such a goal could be straightforward or somehow feasible in some cases, this does not apply to a vast portion of the everyday tasks we expect robots to undertake. The main reason for this is the variations across domestic environments, and hence the personalized and ever-changing needs and preferences of their users. Imagine, in the not-too-distant future, a user who has just brought home some groceries and placed them on her kitchen table. By pushing a button on her smartphone, the user

then notifies her robot that it should put away the groceries by placing each item in its correct place in the kitchen as she would prefer (e.g., refrigerator door, lower cupboard shelf, kitchen counter, etc.). Unless the robot has encountered these exact same items before, solving this task using a state-of-the-art planner would require the user to specify the preferred location to place each item. Obviously, this is extremely inconvenient for the user, and renders practical robotic applications infeasible. At the same time, programming a robot with a set of rules that cater to all possible environments and users is a daunting task for an expert. This is due to the fact that user preferences often stem from factors such as culture, common sense, and personal taste, which are hard to model or account for using a fixed set of rules. Additionally, the robot may need to adapt the solution of the task based on practical considerations with respect to the specific environment it is operating in such as the number of cupboards and other containers in the kitchen.

The same challenges apply to a variety of tasks that we envision service robots to perform in human-centered environments. There typically are numerous ways of solving tasks such as sorting groceries or clothes, arranging books or personal items on shelves, or setting a desk for work or a table for dinner. Rather than following an arbitrary strategy when solving these tasks, several studies have revealed that people expect service robots to attend to those tasks in a manner that corresponds to the personal preferences of their end users (Cakmak and Takayama, 2013; Forlizzi and DiSalvo, 2006; Dautenhahn et al., 2005; Pantofaru et al., 2012; Ray et al., 2008; Smarr et al., 2014; Cha et al., 2015). For example, the results of Pantofaru et al. (2012) show that people exhibit strong feelings with respect to robots organizing personal items, suggesting the need for the robot to ask humans to make decisions about where to store them. At the same time, Pantofaru et al. emphasize that "a robot should not continuously nag its owner." Instead, we seek to design service robots that are capable of intelligently inferring the best way to behave in a specific environment autonomously by leveraging their previous observations and experiences, while continuously augmenting and adapting their knowledge as needed.

In addition to predicting the preferences of users for solving a particular task, robots should be able to continuously learn new arbitrary skills and tasks from their users. Everyday tasks such as organizing objects and tidying up are domains which involve reasoning about desirable spatial relations between objects. A service robot attending to such tasks has to manipulate objects in order to achieve arbitrary spatial configurations that users prefer, e.g., stacking plates or boxes, aligning shoes in a row, or setting a desk by placing the mouse diagonally with respect to the keyboard and the coffee mug behind the monitor on the opposite side with respect to a photo frame. Moreover, robots should be able to solve such tasks even if they do not have prior semantic knowledge about the involved objects. At the same time, it is clear that we cannot program robots with sufficient skills and knowledge to handle all potential tasks in domestic environments

before deploying them.

In this context, learning from demonstrations has emerged as a key paradigm for transferring knowledge from non-expert users to robots (Billard et al., 2008; Argall et al., 2009). By observing demonstrations of a desirable behavior by a human teacher, this approach enables a robot to learn a model of the demonstrated skills and to imitate them on its own without requiring an expert to program them. However, for practical applications, robots should be able to learn using a small number of demonstrations. This often introduces ambiguity with respect to the intention of the teacher when learning complex tasks. Consider, for example, a teacher demonstrating to the robot how to arrange objects on the table in a desirable way. In the general case, the teacher may demonstrate a few valid ways of manipulating and arranging the objects. Without prior knowledge about the task, there typically are several ways of interpreting the demonstrations, rendering it challenging to infer the latent intention of the teacher. Did the teacher intend to place the cup at a specific location on the table, or are the spatial constraints between the cup and the other objects more important for solving the task? How can the robot reproduce the task when faced with new conditions such as a different table height or if some of the objects are missing?

Clearly, adopting a single model for a task based on limited training data will significantly limit the robot's ability to generalize and solve it in new situations. Therefore, it is important to endow robots with the flexibility to reason about multiple valid interpretations of the teacher's demonstrations and to continuously update the learned models from new experiences. More importantly, robots should be able to adapt their knowledge by *improvising* new solutions that align with the preferences of their users and without requiring users to constantly specify how to deal with each new situation. This is a key requirement for robots to be able to learn from non-expert users and then autonomously infer the best way to behave based on the current state they observe.

Another challenge that service robots have to overcome in order to cope with everyday tasks is the ability to generalize useful spatial relations they have learned to new objects. For example, having learned to place a toy inside a basket or hang a clothes hanger in the closet, a robot should be able to achieve the corresponding spatial relations using objects of different shapes and sizes, e.g., placing a book in a drawer or hanging a key chain on a rack, respectively. However, today's robots are typically restricted to reasoning about spatial relations that have been hand-coded by experts for specific domains. In such cases, the expert provides the robot with a library of symbols that represent abstractions of geometric states (e.g., `inside`) as well as the conditions in which these relations are satisfied for a set of objects, e.g., a classifier mapping the poses of the objects to the truth value of the symbol. This greatly limits the ability of a robot to solve tasks using objects that are not accounted for in its prior knowledge. Additionally, our everyday environments and tasks involve arbitrary spatial relations that cannot be accounted for beforehand using expert knowledge. In fact, our world encompasses a continuous

spectrum of spatial relations that do not easily lend themselves to being modeled using a finite set of discrete concepts. It is therefore, crucial to endow robots with the ability to continuously learn such relations from their users and environments and to imitate them using new objects.

The above considerations and challenges facing service robots lead us to posing the following research questions, which we address in this thesis:

- How can a robot autonomously predict the personalized preferences of its users with respect to organizing objects, e.g., when tidying up?
- Given a small number of demonstrations of a new manipulation skill such as grasping or placing an object, how can a robot infer several valid interpretations of the demonstrations and autonomously choose the best way to reproduce the skill in a new situation?
- Given a small number of demonstrations of a new task, how can a robot learn models of the goal of the task and the involved skills? Accordingly, how can a robot use this knowledge to *improvise* a suitable solution of the task without querying the user with an explicit goal state to achieve?
- How can a robot generalize its knowledge about everyday spatial relations to new objects of different sizes and shapes?

In the scope of this thesis, we tackle the problems raised by these questions and provide solutions for them.

1.1. Key Contributions

In this thesis, we present several contributions to the field of robotics research in general, and to the subfields of service robotics and learning manipulation skills from demonstrations in particular. Our contributions address the challenges highlighted above, bringing us closer to deploying intelligent service robots in human-centered environments. The novel techniques we present in this thesis are all motivated by the same underlying philosophy: *i)* domestic robots should consider the preferences of their users: task solutions and even spatial relations are typically subjective concepts that vary across users and environments, *ii)* to handle tasks in domestic environments, future service robots should rely less on expert design as a source of knowledge, and more on lifelong learning through interacting with their users and environments, and *iii)* we should equip service robots with the ability to adapt their knowledge and improvise solutions even when faced with new situations and without constantly querying their users. By developing approaches that take the above points into consideration, we place

the user at the center of robotic design. This is a crucial requirement for intelligent service robots operating amongst humans in domestic environments.

Firstly, we present a novel technique for predicting user preferences with respect to organizational tasks and based on the theory of recommender systems. This enables robots to learn models of user preferences in a lifelong manner by observing various users and environments, and to tailor tasks to individual users accordingly. Secondly, we tackle the challenge of learning models of skills from a small number of ambiguous, non-expert demonstrations. We present novel techniques that enable a robot to construct a flexible model of a demonstrated skill such that it is able to reproduce it in previously unseen situations. Thirdly, a key contribution of this thesis is a novel approach that allows a robot to simultaneously learn models of task goals and skills from a small number of demonstrations. We propose an algorithm that leverages these models to compute a sequence of actions to reproduce the task by improvising solutions based on the current state and without requiring the user to specify a goal explicitly. Finally, we propose a novel approach based on distance metric learning that enables a robot to learn spatial relations in a lifelong manner and generalize them to new objects. We briefly describe these contributions in the following.

Predicting user preferences for organizational tasks In Chapter 3, we address the challenge of predicting the preferences of users with respect to organizing objects in domestic environments by presenting a novel approach based on collaborative filtering, a popular paradigm from the field of recommender systems theory. Our contribution is the first to bridge the gap between the worlds of personalized recommendations and service robotics, thereby enabling robots to learn patterns of user preferences for the domain of organizing objects in containers, e.g., shelves or boxes. We extract these patterns from a corpus of observations of how users in different environments tend to organize objects. Accordingly, and using only a few observations of a new environment or a small number of queries to the user, we enable a service robot to predict the preferences of the user and thus the best way to organize the objects for her based on the number of available containers. To deal with novel objects for which we have no data, we propose a method that complements standard collaborative filtering by leveraging information mined from the Web. Our extensive evaluation using crowdsourcing data from over 1,200 users demonstrates the effectiveness of our approach in predicting user preferences, as well as its applicability on a real robot.

Learning action models from teacher demonstrations In Chapter 4, we address the challenge of acquiring new skills from teacher demonstrations. In particular, we consider the problem of learning new manipulation *actions* such as reaching for or placing an object using a small number of teacher demonstrations. A key challenge in this context is selecting an appropriate feature representation in terms of spatial constraints for

modeling the new action. The majority of learning from demonstrations techniques rely on an expert to specify the relevant features for the action (e.g., the pose of the robot's end-effector relative to an object) and focus on learning the action on the motion level (i.e., *how* to imitate). In contrast to that, we rely on existing solutions to compute feasible motions, and instead tackle the problem of inferring the best feature space for modeling a new action (i.e., *what* to imitate). Specifically, we tackle learning the relevant spatial constraints between the involved objects before and after applying the action. This is crucial for successfully generalizing teacher demonstrations and for sequencing several actions when planning. By learning these constraints, we eliminate the need for an expert to specify the relevant spatial relations for every new action. Due to the small number of demonstrations and the large number of potentially relevant (and often contradictory) spatial constraints, we tackle the ambiguity in the demonstrations from two perspectives. Firstly, we present a novel approach that leverages prior expert knowledge about manipulation actions *in general* to bootstrap learning the new action. We formulate this as a model selection problem in which the robot reasons about multiple valid models of the action based on the demonstrations, and selects the most likely model to reproduce the action based on the current situation. Secondly, we propose a novel approach that does not require the availability of such prior expert knowledge about actions. Instead, we overcome the ambiguity in the demonstrations in a data-driven manner, and learn an *implicit action model* corresponding to a multi-modal distribution of goals based on a mixture of possible interpretations of the teacher demonstrations. Using our approaches, the robot avoids the need to commit to a single model of the action beforehand, thereby achieving flexibility and generalization. Through experiments with simulated and real-world data, we demonstrate the effectiveness of our method and its ability to reproduce an action successfully in previously-unseen states.

Simultaneously learning task goal and action models from teacher demonstrations
The above approaches either focused on inferring a preferred goal state for a task while relying on existing actions and planners to achieve it (Chapter 3) or focused on learning new action models from teacher demonstrations (Chapter 4). In Chapter 5, we tackle the problem of simultaneously learning models for the goals and actions associated with a task from a small number of non-expert teacher demonstrations. We introduce *teach and improvise*, a novel learning from demonstrations framework that enables a robot to compute plans for solving a manipulation task by generalizing and sequencing the learned actions. For this, we build our work on Monte Carlo tree search (MCTS), a popular decision-making framework typically used for solving Markov Decision Processes (MDPs). We leverage the implicit action models we introduce in Chapter 4 and extend the standard MCTS tree structure to reason about multiple interpretations of each action while searching for promising task solutions. Our technique does not assume any semantic knowledge about the task such as a library of predefined symbols to model

spatial relations. Instead, we adopt a continuous, multi-modal model of spatial relations based on the relative poses between the objects. Moreover, a key contribution of our approach is that, as opposed to standard planning paradigms, we do not require the user to specify a goal state to achieve. Instead, we formulate solving the task as an optimization problem in which the robot aims to maximize the likelihood of satisfying the intention of the teacher with respect to relevant spatial relations between the objects. This allows the robot to *improvise* physically feasible solutions in an anytime fashion, i.e., based on the budget of iterations or time available for solving the task, and to adapt these solutions based on the initial state. Finally, our approach enables the robot to continuously improve its performance by *practicing* the task in simulation and subsequently updating the learned action models without requiring additional teacher demonstrations. Through extensive experiments in simulation and with a real robot, we demonstrate the effectiveness and flexibility of our approach in improvising solutions from arbitrary initial states and for a variety of tasks.

Generalizing Spatial Relations to New Objects The methods above either relied on predefined spatial relations for solving a task (e.g., an object being on a shelf as in Chapter 3), or modeled spatial relations as distributions defined over relative poses between objects (Chapters 4 and 5). Those models are unable to generalize relations to objects of varying shapes and sizes, thereby limiting the robot to solving tasks using previously-known objects. We address this issue in Chapter 6 by first proposing a novel descriptor that encodes a pairwise spatial relation based only on 3D models of the objects and their poses, and without assuming prior semantic knowledge about the objects or the relation. Using this feature representation, we then leverage large margin nearest neighbor, a popular distance metric learning approach, to learn a metric that captures the similarities between relations. For training, our approach relies only on partially labeled data describing how similar pairs of relations are to each other, and which the robot accumulates from non-expert users over time and in an interactive manner. Using the learned metric, we bootstrap learning a new relation by relating it to similar, previously-learned ones. Our approach reduces the problem of imitating a relation to one of minimizing the *distance* between two scenes, even if they involve objects of varying shapes and sizes. Our work goes beyond existing approaches that learn a model (e.g., a classifier) for each relation individually, and that are therefore limited to reasoning about a finite number of relations. Our extensive evaluation with real-world data demonstrates the effectiveness of our method in reasoning about a continuous spectrum of spatial relations and generalizing them to new objects.

1.2. Publications

Parts of the research presented in this thesis have undergone international peer review. In the following, we list the corresponding publications in chronological order.

- N. Abdo, L. Spinello, W. Burgard, and C. Stachniss. Inferring what to imitate in manipulation actions by using a recommender system. In *Proc. of the IEEE International Conference on Robotics and Automation (ICRA)*, 2014.

- N. Abdo, L. Spinello, C. Stachniss, and W. Burgard. Collaborative filtering for learning user preferences for robotic tasks. In *Workshop on Learning Plans with Context from Human Signals at Robotics: Science and Systems (R:SS)*, 2014.

- N. Abdo, C. Stachniss, L. Spinello, and W. Burgard. Robot, Organize my Shelves! Tidying up objects by predicting user preferences. In *Proc. of the IEEE International Conference on Robotics and Automation (ICRA)*, 2015. **Finalist for best service robotics paper, best student paper, and best conference paper awards.**

- N. Abdo, C. Stachniss, L. Spinello, and W. Burgard. Organizing objects by predicting user preferences through collaborative filtering. In *the International Journal of Robotics Research (IJRR)*, 35(13):1587–1608, 2016.

- O. Mees, N. Abdo, M. Mazuran, and W. Burgard. Metric learning for generalizing spatial relations to new objects. Accepted for publication at *the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2017.

Moreover, the following publications and preprints of the author of this thesis present work related to perception as well as (combined) task and motion planning. They are, however, outside the main focus of this thesis and therefore not covered (in detail). The works concern motion planning in environments with deformable obstacles, learning symbolic representations of manipulation actions, robust high-level reasoning and sensor fusion, active perception, and the recognition of everyday objects.

- N. Abdo, H. Kretzschmar, and C. Stachniss. From low-level trajectory demonstrations to symbolic actions for planning. In *Proc. of the Workshop on Combining Task and Motion Planning for Real-World Applications (TAMPRA) at the International Conference on Automated Planning and Scheduling (ICAPS)*, 2012.

- N. Abdo, H. Kretzschmar, L. Spinello, and C. Stachniss. Learning manipulation actions from a few demonstrations. In *Proc. of the IEEE International Conference on Robotics and Automation (ICRA)*, 2013.

- T. Niemueller, N. Abdo, A. Hertle, G. Lakemeyer, W. Burgard, and B. Nebel. Towards deliberative active perception using persistent memory. In *Proc. of the Workshop on AI-based Robotics at the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2013.

- C. Schwering, T. Niemueller, G. Lakemeyer, N. Abdo, and W. Burgard. Sensor fusion in the epistemic situation calculus. In *Proc. of the International Cognitive Robotics Workshop (CogRob) at the European Conference on Artificial Intelligence (ECAI)*, 2014.

- C. Schwering, T. Niemueller, G. Lakemeyer, N. Abdo, and W. Burgard. Sensor fusion in the epistemic situation calculus. In *Journal of Experimental & Theoretical Artificial Intelligence*, 28(5):871-887, 2016.

- P. Jund, N. Abdo, A. Eitel, and W. Burgard. The Freiburg groceries dataset. *arXiv preprint*, arXiv:1611.05799, 2016.

1.3. Collaborations

This thesis covers work that involved collaborations with other researchers, which we outline in the following.

- As supervisor of this thesis, Wolfram Burgard contributed to all parts through scientific discussions.

- Chapter 3: The work on predicting user preferences using collaborative filtering was supervised by Luciano Spinello and Cyrill Stachniss (see Abdo et al. (2014b, 2015, 2016) for the related publications).

- Chapter 4: The work on feature selection based on expert templates was supervised by Luciano Spinello and Cyrill Stachniss (see Abdo et al. (2014a)). The work on implicit action models was supervised by Gian Diego Tipaldi and supported through discussions with Christian Dornhege.

- Chapter 5: The work on the teach-and-improvise framework was supervised by Christian Dornhege and Gian Diego Tipaldi. Gian Diego Tipaldi contributed during the early stages of this work to the formulation of the problem and with ideas concerning the use of the implicit action models in the framework. Christian Dornhege contributed to the teach-and-improvise algorithm through discussions and his expertise in planning. Additionally, Christian Dornhege contributed his expertise with the MoveIt! motion planning library for performing the feasibility checks and for realizing the algorithm on the PR2 robot.

- Chapter 6: The work on distance metric learning for generalizing spatial relations is an extension of the Master's thesis of Oier Mees, which the author of this thesis co-supervised with Mladen Mazuran (see Mees et al. (2017)). The author of this thesis and Mladen Mazuran contributed substantially to the ideas and formulation of the metric learning problem and the novel descriptor for modeling spatial relations. The author of this thesis and Mladen Mazuran also contributed to data collection and the prototyping of the implementation of the descriptor. The implementation and evaluation of the approach was realized by Oier Mees.

2. Problem Formulation

In the course of this thesis, we consider problems that address how a robot can learn to solve tasks in the context of human-centered environments. We tackle these problems by proposing novel techniques that enable a robot to learn models of actions, tasks, and spatial relations between objects. Our work is flexible in that it is able to leverage different sources of information for learning. On the one hand, we address learning in settings in which the robot has no prior knowledge about the task. In this context, we propose techniques that are able to learn and generalize based on a small number of examples provided by non-expert teachers. On the other hand, given expert knowledge or large amounts of data gathered across different users and environments, we propose novel methods that enable a robot to use this knowledge to bootstrap learning new models. In this chapter, we introduce the concepts and definitions needed to formalize and present our work in the subsequent chapters. Moreover, we concretely define the problems we address in this thesis and highlight where they lie on the spectrum of assumptions and prior knowledge available to the robot.

2.1. Preliminaries and Notation

In this work, we consider the problem of enabling a robot to solve tasks in domestic environments. Specifically, we focus on goal-oriented manipulation tasks in which the aim of the robot is to achieve arrangements of objects to satisfy spatial constraints that are desirable by its users. In this section, we introduce the components involved in such tasks, as well as the corresponding notation that we adopt throughout this work.

2.1.1. Objects and State Representation

We consider tasks in which a robot has to change the state of the environment by manipulating *objects* in order to achieve a desirable goal state. Each task is associated with a non-empty set of objects $\mathcal{O} = \{o_1, \ldots, o_{|\mathcal{O}|}\}$ that are used to define the *state* $\mathbf{s}$ and to model desirable goals. In this work, we assume the robot is able to identify objects

of interest using existing perception techniques, which allow it to map each perceived object instance in the scene to a unique id and to estimate its pose. Additionally, the techniques we present in this thesis are able to leverage different sources of prior knowledge about the objects. For example, the robot can use prior semantic knowledge (e.g., the type of an object) as well as geometric information (e.g., the shape or 3D model of an object) in order to generalize a learned task to new objects. In the general case, we rely on a function $g_o(o_k, \mathbf{s})$ to represent an object o_k in a state $\mathbf{s}$, i.e., $g_o(o_k, \mathbf{s}) = \mathbf{o}_k$. An example of $\mathbf{o}_k$ is a point cloud of o_k or an id based on its semantic type, e.g., cup_1.

We define the state $\mathbf{s}_t$ at time t in terms of the poses of all objects expressed relative to a stationary world frame. We describe the pose of an object $o_k \in \mathcal{O}$ in $SE(3)$ as a homogeneous transformation matrix $\mathbf{T}_k$, composed of a 3D translation vector $\mathbf{t}_k$ and 3D rotation matrix $\mathbf{R}_k$, i.e.,

$$\mathbf{T}_k = \begin{bmatrix} \mathbf{R}_k & \mathbf{t}_k \\ \mathbf{0} & 1 \end{bmatrix}. \tag{2.1}$$

We denote the pose of object o_k at time t by $\mathbf{T}_k(t)$. Furthermore, we use ${}^l\mathbf{T}_k$ to denote the pose of object o_k relative to o_l. Similarly, ${}^l\mathbf{T}_k(t)$ denotes the pose of o_k relative to o_l at time t. Moreover, in this work, we consider learning goal-oriented tasks and actions in which the robot manipulates objects to achieve desirable goals. Therefore, we do not consider the complete joint state of the robot as part of our state representation. Instead, we only consider the pose $\mathbf{T}_{ee}(t)$ and joint configuration $\mathbf{o}_{ee}(t)$ (e.g., gripper opening) of its end-effector o_{ee} as part of our state space.

Finally, note that the dimensionality of the state depends on the number of objects in the environment. In general, we aim for flexible approaches that allow for solving a task using only a subset of the objects $\mathcal{O}$ typically associated with it, e.g., if some of the objects are missing. Therefore, we denote the set of objects that exist in state $\mathbf{s}_t$ as $\mathcal{O}(\mathbf{s}_t) \subseteq \mathcal{O}$.

2.1.2. Pairwise Relations

To model the goals of actions and tasks in terms of desirable object configurations, we rely on *spatial relations* between the involved objects. Specifically, we adopt pairwise relations between objects as the building blocks for such models. We denote the pair consisting of objects o_k and o_l as $p_{(k,l)} = \{o_k, o_l\}$, and the set of pairs over all objects in $\mathcal{O}$ as $\mathcal{P}$. Moreover, we denote the set of pairs over the subset of objects $\mathcal{O}(\mathbf{s}_t)$ existing in state $\mathbf{s}_t$ as $\mathcal{P}(\mathbf{s}_t) \subseteq \mathcal{P}$.

We represent a pairwise spatial relation between two objects o_k and o_l using a relation function $f_\mathcal{R}$ that maps the pair $p_{(k,l)}$ and the current state $\mathbf{s}$ to a K-dimensional feature representation $\mathbf{r}_{(k,l)}$ of the relation, i.e., $f_\mathcal{R}(p_{(k,l)}, \mathbf{s}) = \mathbf{r}_{(k,l)} \in \mathbb{R}^K$.

In general, one can design or learn the function $f_\mathcal{R}$ to model the relation between o_k and o_l on different levels of abstraction, see Figure 2.1. For instance, on a semantic

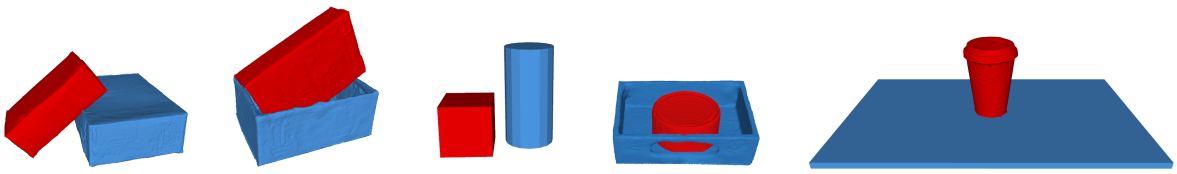


Figure 2.1.: Five examples of spatial relations between pairs of objects. In this thesis, we adopt pairwise relations as building blocks for learning models of actions and tasks to allow a robot to achieve object configurations that are desired by its users. For this, we either leverage existing expert models of relations in the form of high-level predicates (e.g., `inside`), or aim to model relations on a low, geometric level in the absence of prior semantic knowledge about the objects. Additionally, we address the problem of generalizing a learned spatial relation to objects of various shapes and sizes.

level, an expert can provide the robot with a hand-crafted model of $f_{\mathcal{R}}$ based on a set of K predefined relations (e.g., `on top`, `inside`, `on the same shelf`, etc.), such that each dimension of $\mathbf{r}_{(k,l)}$ captures whether the corresponding relation is satisfied or not in the state $\mathbf{s}$ (or the degree to which it is satisfied). In the absence of such prior knowledge, one could model $f_{\mathcal{R}}$ to describe the spatial relation between o_k and o_l based on their relative pose in $\mathbf{s}$, i.e., a 6D representation of ${}^{l}\mathbf{T}_{k}$. Moreover, given 3D models of the objects, one can learn a function $f_{\mathcal{R}}$ to compute a feature representation of spatial relations that additionally encodes the geometries of the objects. In this work, we adopt different models of $f_{\mathcal{R}}$ depending on the domain and the corresponding prior knowledge available to the robot for modeling actions and tasks.

2.1.3. Point-to-Point Actions

In order to achieve a desirable goal state of a task, the robot relies on executing *actions* to change the state of the environment. In this work, we consider learning so-called *point-to-point*, goal-directed actions from demonstrations, which can be modeled in terms of the pairwise relations between the objects before and after applying an action. We aim to learn the desirable goal states when applying the action (i.e., *what* to imitate), and not the trajectory or motion demonstrated by the teacher (i.e., *how* to imitate). For example, the goal of an action could be to move a cup on a tray to a desirable position next to a plate on the table. We want to enable the robot to infer likely goal poses for the cup as a result of executing this action, irrespective of the specific trajectory needed to achieve this goal. Having inferred a goal state to achieve, we assume the existence of a suitable motion planning technique or a controller that is capable of computing and executing the corresponding motion.

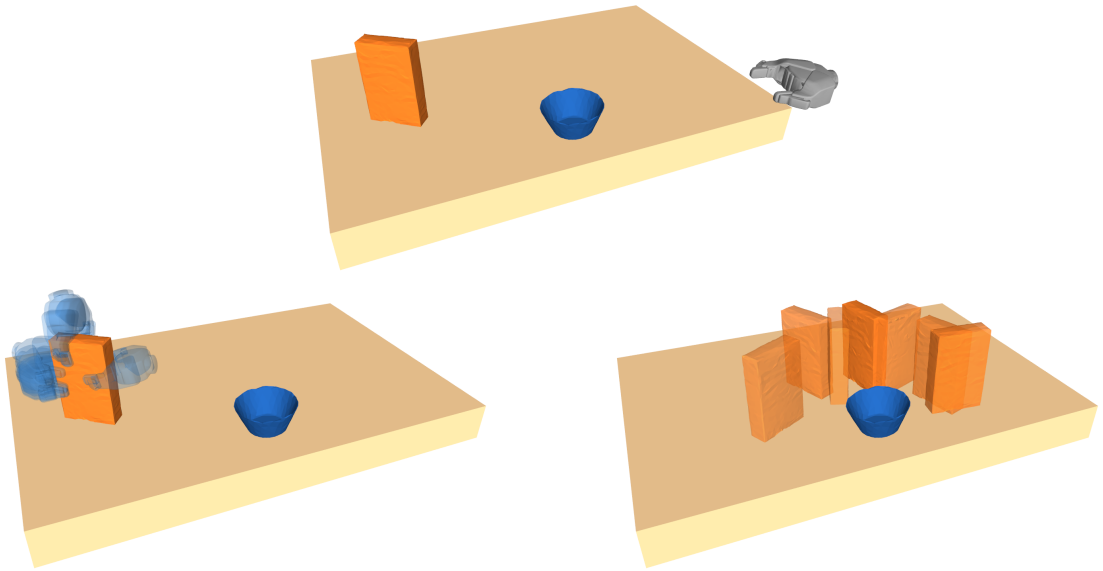


Figure 2.2.: We consider learning point-to-point actions from teacher demonstrations, and which we model in terms of the state before and after applying the action. Top: a random state that we use to illustrate the action of moving the box of cereals. Bottom left: we aim to learn a distribution of *state affordability* for this action. This allows us to reason about initial states that afford applying the action with a high likelihood, e.g., states in which the end-effector is in a grasping pose relative to the box. Bottom right: we learn a goal probability distribution of the action that captures multiple modes of the intention of the teacher with respect to desirable spatial relations between the objects after applying the action, e.g., placing the box around the bowl to set the table.

We denote the set of actions that the robot can execute by $\mathcal{A} = \{a^{(1)}, \ldots, a^{(|\mathcal{A}|)}\}$. We aim to learn actions from non-expert teacher demonstrations. In this context, we assume the teacher does not provide the robot with the conditions necessary for the action to succeed, e.g., an object must be grasped by the robot before it can move it. Instead, we assume this knowledge is implicit in the demonstrations and adopt a data-driven approach to construct a model $p(\mathbf{s}_t \mid a)$ of the starting states $\mathbf{s}_t$ in which the teacher applied the action a. We refer to this distribution as the *state affordability*, which captures the likelihood that a starting state $\mathbf{s}_t$ *affords* applying the action based on the demonstrations. We use this concept to formulate constraints that ensure that the robot only applies an action in states with high affordability to increase the likelihood of the action succeeding as intended by the teacher, see Figure 2.2-bottom left.

Additionally, we associate each action $a \in \mathcal{A}$ with a probability distribution (transition

Figure 2.3.: Top: examples of goal states for everyday tasks such as organizing objects on shelves, setting a table, or tidying up a desk. In this thesis, we aim to enable a robot to learn and solve manipulation tasks that can be modeled in terms of spatial relations between the involved objects. Bottom: rather than requiring the user to provide the robot with an explicit goal state to achieve when solving a task, we aim to enable the robot to infer valid ways of solving a task based on the initial state and thus to improvise feasible solutions that align with the preferences of the user.

function) $p(\mathbf{s}_{t+1} \mid \mathbf{s}_t, a)$ over the goal state $\mathbf{s}_{t+1}$ at time $t+1$ as a result of applying the action starting in state $\mathbf{s}_t$ at time t, see Figure 2.2-bottom right. In general, we consider multi-modal distributions that capture multiple valid goals of the action with respect to the spatial relations between the objects in $\mathbf{s}_{t+1}$.

2.1.4. Task Goals

As mentioned above, the aim of our work is to enable a robot to solve tasks by manipulating objects in the environment in order to achieve desirable goal states in human-centered environments. In the general case, there is typically more than one valid way of solving a task (e.g., arranging objects on shelves), the choice of which depends on the starting state, the environment, and the preferences of the users. Accordingly, rather than specifying a fixed goal state that the robot should always achieve, we associate each task with an *intention likelihood* function $\Psi : \mathcal{S} \to \mathbb{R}$ that models the intended goal of the task, where $\mathcal{S}$ is the set of all possible states. In other words, $\Psi(\mathbf{s})$ captures how well a goal state $\mathbf{s}$ aligns with or satisfies the goals of the task.

In some cases, a user or an expert could specify Ψ as a formula expressing which conditions (spatial relations) have to be satisfied for solving the task (e.g., all objects on the kitchen table should be moved to the sink). However, in this work, we focus on enabling a robot to learn arbitrary tasks even if it is infeasible to provide it with an explicit goal state. Instead, we enable the robot to leverage demonstrations by a user (or several users) to construct a model of Ψ for the task. For instance, by observing examples of goal states, the robot learns which spatial relations between the objects are (more) important for solving the task, see Figure 2.3. Ideally, this model should enable the robot to generalize previously-seen goal states to new situations while still satisfying the preferences of its users.

2.2. Solving a Task

Given the components defined above, we now formally describe what we mean by solving a task. We consider tasks whose goals can be expressed in terms of the spatial relations between the involved objects. The robot achieves such goals by applying actions from a set of actions associated with the task. Therefore, we define a *task* $\mathcal{T}$ as a tuple $\mathcal{T} = \langle \mathcal{O}, \mathcal{A}, \Psi \rangle$, where $\mathcal{O}$ and $\mathcal{A}$ are respectively the objects and actions associated with the task, and Ψ is the intention likelihood function modeling its desirable goal states.

Given an initial state $\mathbf{s}_0$, *solving the task* is the problem of computing a *plan* Π consisting of a sequence of T actions and intermediate states to achieve a desirable goal state $\mathbf{s}_T$, i.e., $\Pi = \Big(\langle a_0, \mathbf{s}_1 \rangle, \ldots, \langle a_{T-1}, \mathbf{s}_T \rangle\Big)$, see Figure 2.4. Each step $\langle a_t, \mathbf{s}_{t+1} \rangle$ of the plan consists of an action $a_t \in \mathcal{A}$ applied in state $\mathbf{s}_t$ at time t to reach state $\mathbf{s}_{t+1}$.

Given a specific goal state $\mathbf{s}_T$ to achieve, this corresponds to solving a (task and motion) planning problem. However, in this work, we aim to enable a robot to solve tasks without requiring its users to provide an explicit goal state to achieve in each case. Instead, the robot should autonomously infer a desirable goal state $\mathbf{s}_T$ to achieve based on the model Ψ. Specifically, we formulate an optimization problem, the aim of which is to compute a plan Π that trades off $\Psi(\mathbf{s}_T)$ with the cost (length) of the solution:

$$\begin{aligned} \underset{\substack{T \in \mathbb{N}_{\geq 0} \\ a_{0:T-1} \in \mathcal{A} \\ \mathbf{s}_{1:T}}}{\text{maximize}} \quad & \Psi(\mathbf{s}_T) - \sum_{t=0}^{T-1} \text{cost}(a_t), \\ \text{subject to} \quad & \mathbf{s}_t \in \mathcal{S}_\text{f} \qquad\qquad \forall t, \end{aligned} \tag{2.2}$$

where $a_{0:T-1}$ and $\mathbf{s}_{1:T}$ are the plan actions and states, respectively. Note that in general, the length T of the plan is unknown beforehand. Moreover, $\text{cost}(a_t)$ denotes the cost of applying an action, which we incorporate in order to bias the solution towards less expensive plans. Furthermore, $\mathcal{S}_\text{f}$ denotes the set of all physically feasible states. The corresponding constraints ensure that each step of the computed plan is feasible. This

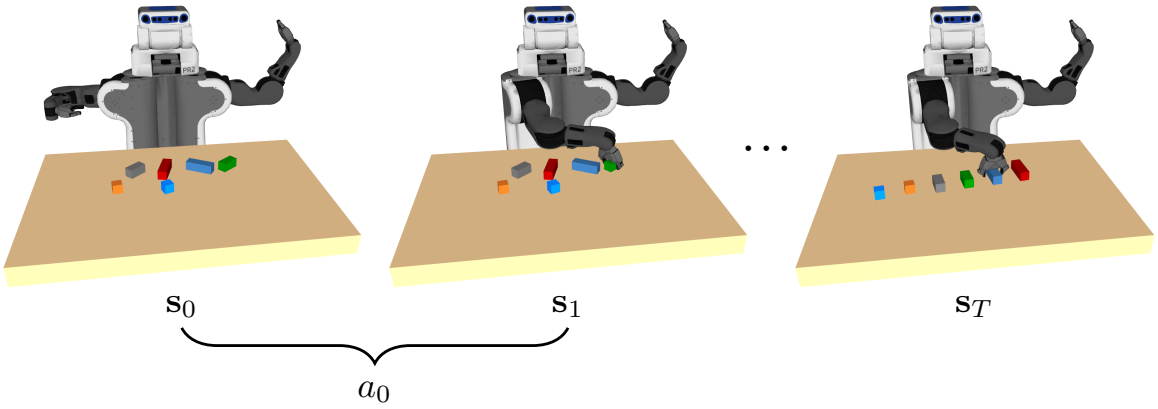


Figure 2.4.: We formulate solving a task starting in an initial state $\mathbf{s}_0$ as an optimization problem, the goal of which is to compute a plan consisting of a sequence of actions $(a_0, \ldots, a_{T-1})$ and the corresponding goal states $(\mathbf{s}_1, \ldots, \mathbf{s}_T)$ to achieve a goal state $\mathbf{s}_T$ that maximizes the intention likelihood $\Psi(\mathbf{s}_T)$ modeling the intention or preference of a user for the task. Rather than querying the user for the best goal state $\mathbf{s}_T$ to achieve in each case, we aim to enable the robot to autonomously infer a valid goal based on $\mathbf{s}_0$.

allows the robot to adapt the solution of the task based on the initial state and what is physically possible in reality. In the next section, we briefly describe these constraints and how we enforce them in our work.

State Feasibility Whereas full-fledged feasibility checks (e.g., based on physics simulations) are outside the scope of this thesis, we consider three types of constraints in our work as shown in Figure 2.5. Firstly, while we allow objects (including the robot's end-effector) to come in contact with each other (e.g., when stacking blocks or grasping a cup), we rely on collision checks to reject solutions in which two objects penetrate each other (e.g., attempting to place two objects in the same place). Secondly, we consider stability constraints that ensure that each object is in contact with a supporting surface beneath it such as a table or shelf, i.e., no object is "floating". We approximate this using a heuristic that ensures an overlap between the 2D convex hulls of objects (when projecting their 3D models to the x-y plane) and by checking the smallest distance between objects in the z (gravity) direction. Thirdly, we consider inverse kinematic constraints to ensure that the robot is able to reach for and place objects based on its degrees of freedom and joint limits. To check all feasibility constraints in our work, we implemented tools that leverage the collision detection capabilities and inverse kinematic solvers of the MoveIt! library (Sucan and Chitta, 2013) using 3D mesh models of all objects and a joint model of the robot. Finally, as we consider point-to-point actions in our work, the above constraints ensure state feasibility for every step of a plan. To

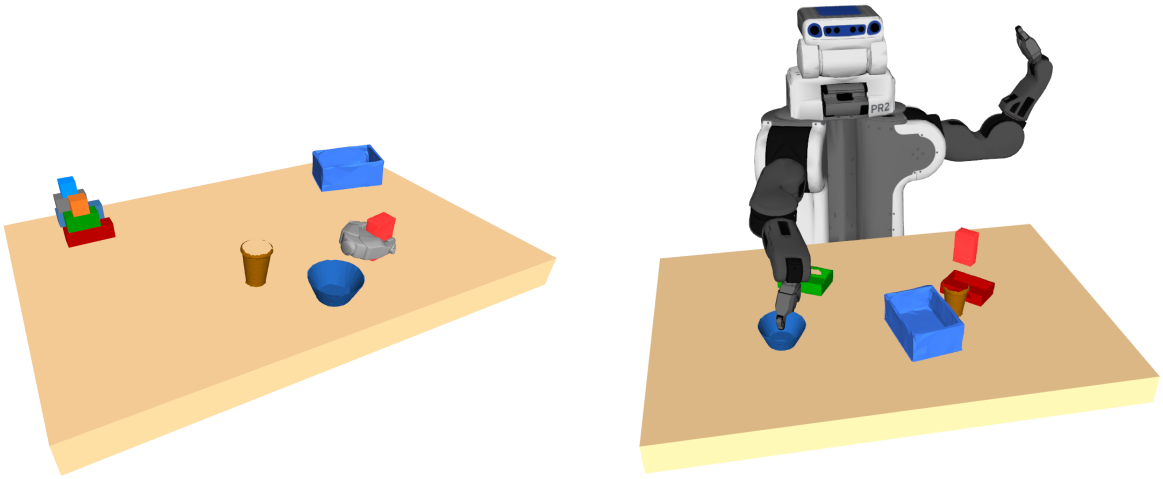


Figure 2.5.: We consider feasibility constraints to allow the robot to reason about its ability to realize task solutions in practice. Left: we allow contacts between objects (including the end-effector), for example when stacking or grasping objects. However, we check collisions to reject states in which two objects penetrate each other (e.g., placing two objects in the same place). Additionally, we consider stability constraints to ensure that each object is supported by another (e.g., a table) to reject states in which an object is "floating". Right: for a desirable end-effector pose, we ensure that there is a feasible inverse kinematics solution based on the robot's degrees of freedom. Additionally, with the exception of the end-effector, we disallow collisions between the robot and the rest of the environment.

compute and execute feasible manipulation trajectories between consecutive plan states, we rely on existing motion planners and controllers.

2.3. Problem Variants

Eq. (2.2) above describes the main problem we address in this thesis in a general form. To solve a task $\mathcal{T}$ from a starting state $\mathbf{s}_0$, we need both action models $\mathcal{A}$ and a model Ψ of desirable goal states for the task. Such models can either be provided by an expert, specified by the user, or learned by the robot from teacher demonstrations or by observing its environment over time. Moreover, we define these models over relations between the objects $\mathcal{O}$ involved in the task. Depending on the context, the robot can leverage different representations of the objects, both on semantic and geometric levels, in order to learn the relevant task models.

Therefore, in this section, we briefly discuss different variants of this problem. Each variant considers a different set of assumptions with respect to which of the above

Table 2.1.: Different variants of the general problem in Section 2.2. Green cells indicate components that are assumed to be known a-priori, blue cells indicate models we aim to learn, and gray cells represent irrelevant components.

object models	**relation models**	**action models**	**task goal model**	
semantic and geometric	symbolic predicates	planning operators	goal formula	**P1**
semantic and geometric	learned from a teacher	learned from a teacher	goal formula	**P2**
semantic and geometric	symbolic predicates	planning operators	lifelong, continual learning	**P3**
geometric	learned from a teacher	learned from a teacher	learned from a teacher	**P4**
geometric	lifelong, continual learning			**P5**

components are known a-priori, and which are learned or inferred by the robot. We identify five such cases **P1**-**P5** in Table 2.1, where green cells indicate components that are assumed to be known a-priori, blue cells indicate that we aim to learn or infer models of the corresponding components, and gray cells correspond to irrelevant components.

We now briefly explain these cases and highlight which of them we focus on and tackle in the course of this thesis.

- **P1:** This corresponds to the case where the robot is provided with action models in the form of planning operators, as a well as an explicit goal state to achieve in the form of a goal formula. Both the planning operators and the goal are typically expressed in terms of logical predicates that model relations between the objects. Additionally, the robot is equipped with object models that describe both their geometric properties (e.g., 3D models) and semantics (e.g., type). This problem is a form of *combined task and motion planning*, in which the robot computes a plan that satisfies the specified high-level goal while ensuring that the intermediate actions are physically feasible, i.e., by planning trajectories on a geometric level (Dornhege et al., 2009; Kaelbling and Lozano-Pérez, 2013; Srivastava et al., 2014; Gharbi et al., 2015).

- **P2:** Whereas **P1** focuses on planning to achieve a specified goal state given existing action models, in this work we are interested in enabling the robot to learn new tasks from its users. One aspect of this is to equip the robot with

techniques to learn new action models and relations between the objects from teacher demonstrations. **P2** describes this problem variant, in which the robot learns actions and relevant relations between the objects on symbolic and geometric levels of abstraction from teacher demonstrations. This then allows the robot to leverage existing state-of-the-art planning techniques in order to compute plans leading to a specified goal state, as in the case of **P1**. There exists a large body of works addressing learning action models from teacher demonstrations on both geometric and symbolic levels of abstraction (Billard et al., 2008; Argall et al., 2009). In general, such techniques focus on learning the action on the motion level to reproduce trajectories demonstrated by a teacher and achieve a specified goal. In contrast to that, in Chapter 4, we tackle learning the relevant spatial constraints between objects before and after applying an action. In this way, we eliminate the need for an expert to provide a suitable model or set of features to use for learning each new action. We propose flexible models that enable a robot to reason about multiple modes in the intention of the teacher with respect to action goals, allowing the robot to generalize an action to new situations.

- **P3:** Rather than relying on the user to specify the goal of the task, this case considers learning a model Ψ of task goals from training data. Examples of this are tasks in which the robot has to arrange objects according to the preferences of its users. Here, it is impractical to expect an expert to pre-program the robot with a fixed goal model that should apply to all users. Instead, we aim at learning *personalized* goal models using training data we collect from different users and environments, e.g., preferred object arrangements. Additionally, the robot should be able to continuously update this model in a lifelong manner given more data. We address this problem in Chapter 3, where we present an approach that leverages collaborative filtering to learn user preferences with respect to object arrangements. Here, we assume that the robot is equipped with prior knowledge about the objects, as well as with action models that enable it to use existing planning techniques to solve the task given the inferred goal state.

- **P4:** Whereas **P2** considered learning action models from a teacher in order to solve the task *given* a specified goal state, here we address the problem of additionally inferring the intended task goal model Ψ from teacher demonstrations. By observing demonstrations of how the non-expert teacher solves the tasks, we aim to simultaneously learn models of the actions as well as the overall goal of the task. As opposed to classical planning given an explicit goal state, this enables the robot to *improvise* desirable goal states $\mathbf{s}_T$ (i.e., maximizing $\Psi(\mathbf{s}_T)$) without requiring the user to explicitly specify them for each starting state $\mathbf{s}_0$. In this problem setting, we leverage the action models introduced in Chapter 4 to tackle the inherent ambiguity in the teacher demonstrations and provide the robot with

great flexibility when solving the task. Finally, we address how the robot can *improve* the learned action models by practicing to solve the task in simulation, such that it is able to better generalize the teacher demonstrations and compute more efficient plans. Accordingly, in Chapter 5 we present a novel method for solving a task by sequencing actions to achieve desirable goal states by learning from a small number of ambiguous demonstrations.

- **P5:** Finally, cases **P1-P4** relied on either known relation models $f_{\mathcal{R}}$ (e.g., predicates defined by an expert), or on modeling relations based on the relative poses between the objects. However, this limits the robot's ability to generalize relations (and hence action and task goals) to new objects of previously unseen shapes and sizes. Therefore, here we focus on learning relation functions $f_{\mathcal{R}}$ that capture the underlying meaning of a pairwise relation irrespective of the object models. One way to address this is to provide the robot with labeled training examples of relations in order to learn a model (e.g., a classifier) that generalizes a relation to different objects (Rosman and Ramamoorthy, 2011; Kulick et al., 2013; Fichtl et al., 2014). However, this typically requires a large number of training examples that need to be labeled (e.g., `on top`, `inside`, etc.) by an expert or a user. More importantly, this requires learning a model for each such new relation we aim to teach the robot, which can be infeasible. Instead, we address this problem in Chapter 6 from the perspective of metric learning, where we propose to learn a feature representation $f_{\mathcal{R}}$ that results in small distances between similar relations and large distances between dissimilar ones. We learn such a model using only a *soft* labeling of training data in the form of how similar two examples are. This enables the robot to reason on a continuous spectrum of spatial relations and to generalize a newly-demonstrated relation to different objects.

3. Organizing Objects by Predicting User Preferences via Collaborative Filtering

In this chapter, we address the problem of predicting user preferences with respect to task goals. Specifically, we consider organizational tasks in domestic environments such as tidying up objects by arranging them on different shelves. In this context, different users typically have their own preferences, many of which depend on a variety of factors including personal taste, cultural background, or common sense. This makes it challenging for an expert to pre-program a robot to accommodate all potential users. At the same time, it is impractical for robots to constantly query users about the preferred way to attend to such tasks. We tackle this challenge by presenting a novel approach based on collaborative filtering, a popular paradigm from the field of recommender systems theory. Our approach enables robots to learn patterns of user preferences with respect to organizing objects in containers such as boxes or shelves. We extract these patterns from a corpus of data from different users, which we gather using crowdsourcing. To deal with novel objects for which we have no data, we propose a method that complements standard collaborative filtering by leveraging information mined from the Web. When solving a tidy-up task, we first predict pairwise object preferences of the user. Then, we subdivide the objects in containers by modeling a spectral clustering problem. Our solution is easy to update, does not require complex modeling, and improves with the amount of user data. We evaluate our approach using crowdsourcing data from over 1,200 users and demonstrate its effectiveness for two tidy-up scenarios. Additionally, we show that a real robot can reliably predict user preferences using our approach.

One of the key goals of robotics is to develop autonomous service robots that assist humans in their everyday life. One envisions smart robots that can undertake a variety of

Figure 3.1.: Left: different ways of organizing a set of grocery objects on shelves according to varying user preferences. Right: our approach enables a service robot to tidy up objects by predicting and following such subjective preferences. We predict pairwise preferences between objects with respect to placing them on the same shelf. We then assign objects to different shelves by maximally satisfying these preferences while considering the number of shelves available for solving the task.

tasks including tidying up, cleaning, and attending to the needs of disabled people. For performing such tasks effectively, each user should teach her robot *how* she likes those tasks to be performed. However, learning user preferences is an intricate problem. In a home scenario, for example, each user has a preferred way of sorting and storing groceries and kitchenware items in different shelves or containers. Many of our preferences stem from factors such as personal taste, cultural background, or common sense, which are hard to formulate or model a priori. At the same time, it is highly impractical for the robot to constantly query users about their preferences.

In this work, we provide a novel solution to the problem of learning user preferences for arranging objects in tidy-up tasks. Our method is based on the framework of collaborative filtering, which is a popular paradigm from the data-mining community. Collaborative filtering is generally used for learning user preferences in a wide variety of practical applications including suggesting movies on Netflix or products on Amazon. Our method predicts user preferences of pairwise object arrangements based on partially-known preferences, and then computes the best subdivision of objects in shelves or boxes. It is able to encode multiple user preferences for each object and does not require that all user preferences are specified for all object-pairs. Our approach is even able to make predictions when novel objects, unknown to previous users, are presented to the robot. For this, we combine collaborative filtering with a mixture of experts that compute similarities between objects by using object hierarchies. These hierarchies consist of product categories downloaded from online shops, supermarkets, etc. Finally, we organize objects in different containers by finding object groups that maximally satisfy the predicted pairwise constraints. For this, we solve a minimum k-cut problem

by efficiently applying self-tuning spectral clustering. Our prediction model is easy to update and simultaneously offers the possibility for lifelong learning and improvement.

To discover patterns in user preferences, we first bootstrap our learning by collecting many user preferences, e.g., through crowdsourcing surveys. Using this data, we learn a model for object-pair preferences for a certain tidy-up task. This enables the robot to infer the preferences of a new user based only on partial knowledge of her preferences, e.g., a few observations of the environment or a small number of interactions with her. Accordingly, the robot is able to predict the unknown object-pair preferences of the new user, and sort the objects (see Figure 3.1).

Our approach is the first to bridge the gap between the worlds of personalized recommendations and service robotics, thereby enabling robots to learn patterns of user preferences with respect to organizing objects in a lifelong manner. Concretely, we make the following contributions in this chapter:

- We model the problem of organizing objects in different containers using the framework of collaborative filtering for predicting personalized preferences.
- We present an approach by which a service robot can easily learn the preferences of a new user using observations from the environment and a model of preferences learned from several previous users.
- We present a novel method to complement standard collaborative filtering techniques by leveraging information from the Web in cases where there is not enough ratings to learn a model.
- We present an extensive experimental evaluation using crowdsourcing data that demonstrates that our approach is suitable for lifelong learning of user preferences with respect to organizing objects.

Our evaluation covers two relevant tidy-up scenarios, arranging toys in different boxes and grocery items on shelves, as well as a real-robot experiment. For training and evaluation, we collected preferences from over 1,200 users through different surveys. Finally, in this chapter, we focus on the problem of inferring the best goal state, i.e., object arrangement, given the preferences of a user and the available containers in the environment. To achieve these goals in our real-robot experiments, we rely on existing actions and solutions for task and motion planning.

3.1. Problem Formulation

In this section, we formalize the problem we address in this chapter. Our goal is to enable a service robot to reason about the preferred way to sort a set of objects into

containers when tidying up in the environment of a specific user. To achieve this, we aim at predicting the preferences of the user with respect to grouping different objects together. As the types of objects (e.g., grocery items) and number of containers (e.g., shelves) typically vary across environments, we aim to learn user preferences for object-object combinations, rather than directly learning an association between an object and a specific container. The problem of predicting an object-object preference for a user closely resembles that of suggesting products to customers based on their tastes. This problem is widely addressed by employing recommender systems, commonly used by websites and online stores such as Amazon and Netflix. The key idea there is to learn to make recommendations based on the purchasing histories of different users collaboratively. In the same spirit of reasoning about products and users, our method relates pairs of objects to users. We predict a *rating* that captures the preference of a user for an object-pair based on two sources of information: *i*) known preferences of the user, e.g., how the user has previously organized other objects, and *ii*) how other users have organized these objects in their environments.

3.1.1. State Representation

More formally, let $\mathcal{O} = \{o_1, o_2, \ldots, o_{|\mathcal{O}|}\}$ be a set of objects, each belonging to a known class, e.g., book, coffee, stapler, etc. In this work, we rely on existing perception techniques to recognize objects, identify their types, and estimate their poses in the scene. Additionally, we define $\mathcal{P} = \{p_1, p_2, \ldots, p_M\}$ as the set of all pairs of objects from $\mathcal{O}$. Moreover, we assume to have a finite number of *containers* $\mathcal{C} = \{c_1, c_2, \ldots, c_{\mathcal{C}}\}$, which the robot can use to organize the objects, e.g., shelves, drawers, boxes, etc. In this work, we assume the robot is provided with a map of the environment such that the poses and geometric models of the containers are known. Additionally, we do not address the problem of identifying sub-containers that the robot could use to organize the objects (e.g., shelves within a cupboard). Instead, we assume that each provided container in itself corresponds to a unique location in the environment that the robot can use, e.g., a specific shelf in a cabinet or a drawer in the kitchen.

Furthermore, we model each container as a set which could be $\emptyset$ or could contain a subset of $\mathcal{O}$. Accordingly, we use $o_1 \in c_2$ to denote the fact that object o_1 is assigned to container c_2. In this chapter, we do not address learning models of spatial relations between objects and containers and instead rely on prior expert knowledge to provide this information. For example, this could be in the form of predicates such as $\texttt{in}(o_1, c_2)$ or $\texttt{on}(o_1, c_2)$ in the case of c_2 being a box or a shelf, respectively. We assume the expert models of such spatial relations enable the robot to map object poses to the truth values of those predicates based on the known poses and geometric properties of the containers.

Accordingly, we express the state $\mathbf{s}$ in terms of the poses of all objects and containers in the environment (see Section 2.1.1). Together with the provided spatial relations

between objects and containers (e.g., $\texttt{on}(o_1, c_2)$), this enables us to reason about which objects are assigned to which containers in the current state. To organize objects, we aim to satisfy user preferences with respect to assigning pairs of objects to the same container. Therefore, for a pair $p_i = \{o_k, o_l\}$, we adopt a binary relation function $f_{\mathcal{R}}(p_i, \mathbf{s}) \in \{0, 1\}$ that is 1 if objects o_k and o_l are assigned to the same container and 0 otherwise, i.e.,

$$f_{\mathcal{R}}(p_i, \mathbf{s}) = \begin{cases} 1 & \text{if } \exists c_m \in \mathcal{C} \text{ s.t. } o_k, o_l \in c_m \\ 0 & \text{if } \exists c_m, c_n \in \mathcal{C} \text{ s.t. } o_k \in c_m, o_l \in c_n, m \neq n. \end{cases} \tag{3.1}$$

3.1.2. Users and Ratings

Given a set of users $\mathcal{U} = \{u_1, \ldots, u_N\}$, we aim to assign a rating r_{ij} to a pair $p_i = \{o_k, o_l\}$ to denote the preference of user u_j for placing o_k and o_l in the same container. Each rating takes a value between 0 and 1, where 0 means that the user prefers to place the objects of the corresponding pair into separate containers, and 1 means that the user prefers placing them together. For convenience, we use $r(o_k, o_l)$ to denote the rating for the pair consisting of objects o_k and o_l when the user is clear from the context. Accordingly, we construct a ratings matrix $\mathbf{R}$ of size $M \times N$, where the rows correspond to the elements in $\mathcal{P}$ and the columns to the users, see Figure 3.2. We use R to denote the number of known ratings in $\mathbf{R}$. Note that typically, $R \ll MN$, i.e., $\mathbf{R}$ is missing most of its entries. This is due to the fact that each user typically "rates" only a small subset of object-pairs. In this work, we denote the set of indices of object-pairs that have been rated by user u_j by $\mathcal{I}_j \subseteq \{1, \ldots, M\}$. Analogously, $\mathcal{J}_i \subseteq \{1, \ldots, N\}$ is the set of indices of users who have rated object-pair p_i.

3.1.3. The Problem

Given a set of objects $\mathcal{O}' \subseteq \mathcal{O}$ that the robot has to sort for a specific user u_j, and the set of containers $\mathcal{C}$ available for the robot to complete this task, our goal is twofold. Firstly, we aim to predict the unknown preference $\hat{r}_{ij}$ of the user for each pair p_i in the set of object-pairs $\mathcal{P}'$ over $\mathcal{O}'$. Secondly, our goal is to assign each object to a specific container such that the predicted user preferences are maximally satisfied. For this, we adopt a function $\Psi : \mathbf{s} \to \mathbb{R}$ that captures the degree to which an object arrangement, modeled by the state $\mathbf{s}$, satisfies the predicted preferences $\hat{r}_{ij}$ of the user:

$$\Psi(\mathbf{s}) = -\sum_{p_i \in \mathcal{P}'} \Big(\hat{r}_{ij} - f_{\mathcal{R}}(p_i, \mathbf{s})\Big)^2. \tag{3.2}$$

Ideally, $\Psi(\mathbf{s})$ is 0 if we are able to compute an arrangement such that all object-pair preferences are satisfied, e.g., $f_{\mathcal{R}}(p_i, \mathbf{s}) = 1$ if $\hat{r}_{ij} = 1$ and $f_{\mathcal{R}}(p_i, \mathbf{s}) = 0$ if $\hat{r}_{ij} = 0$.

$$
\begin{array}{c}
\\
\mathcal{P}\text{ (pairs)}
\end{array}
\begin{array}{c}
\mathcal{U}\text{ (users)} \\
\begin{array}{cccccc}
 & u_1 & u_2 & & u_j & & u_N \\
p_1 & * & r_{12} & & \dots & & * \\
p_2 & r_{21} & r_{22} & & \dots & & r_{2N} \\
 & * & & \ddots & & & * \\
p_i & \vdots & \vdots & & r_{ij} & \ddots & \vdots \\
 & & & & & & \\
p_M & r_{M1} & * & & \dots & & r_{MN}
\end{array}
\end{array}
$$

Figure 3.2.: The ratings matrix $\mathbf{R}$. Each entry r_{ij} corresponds to the rating of a user u_j for an object-pair $p_i = \{o_k, o_l\}$, a value between 0 and 1 denoting whether the two objects should be placed in the same container or not. Our goal is to predict the missing ratings denoted by * and thus compute a partitioning of the objects in different containers that satisfies the user preferences.

However, given the limited number of containers available for solving the task and the often contradictory pairwise preferences, it is typically not possible to satisfy all user preferences. Accordingly, we seek a feasible goal state $\mathbf{s}^*$ that maximizes $\Psi(\mathbf{s})$:

$$\mathbf{s}^* = \arg\max_{\mathbf{s} \in \mathcal{S}_\mathrm{f}} \Psi(\mathbf{s}), \tag{3.3}$$

where $\mathcal{S}_\mathrm{f}$ is the set of all feasible states (see Section 2.2). Note that the solution of Eq. (3.3) is a state modeled in terms of the poses of all objects $\mathcal{O}'$. However, in this chapter, we focus on computing (semantically) preferred task goals in terms of assigning objects to containers, and not on a low level in terms of object poses. Therefore, we first leverage our model of relations (Eq. (3.1)) to abstract away from the object poses and determine which pairs of objects should be placed in the same container. Accordingly, we rely on existing task and motion planning solutions to sample a feasible state $\mathbf{s}^*$ that corresponds to this assignment and to compute a plan that achieves it.

3.2. Collaborative Filtering for Predicting Pairwise Object Preferences

In this section, we describe how we learn models of user preferences from the ratings matrix $\mathbf{R}$. Accordingly, we address the problem of predicting the ratings of a new user for pairs of objects based on partial knowledge of the user's preferences.

3.2.1. Collaborative Learning of User Preferences

We aim to discover latent patterns in the ratings matrix $\mathbf{R}$ that enable us to make predictions about the preferences of users. For this, we leverage a popular technique from the field of factorization-based collaborative filtering (Koren, 2008, 2010). First, we decompose $\mathbf{R}$ into a bias matrix $\mathbf{B}$ and a residual ratings matrix $\overline{\mathbf{R}}$:

$$\mathbf{R} = \mathbf{B} + \overline{\mathbf{R}}. \tag{3.4}$$

Each entry b_{ij} in $\mathbf{B}$ is formulated as follows:

$$b_{ij} = \mu + b_i + b_j, \tag{3.5}$$

where μ is a global bias term, b_i is the bias of the pair p_i, and b_j is the bias of user u_j. We compute μ as the mean rating over all users and object-pairs in $\mathbf{R}$, i.e.,

$$\mu = \frac{1}{R} \sum_{i=1}^{M} \sum_{j \in \mathcal{J}_i} r_{ij}. \tag{3.6}$$

The bias b_j describes how high or low a certain user u_j tends to rate object-pairs compared to the average user. Similarly, b_i captures the tendency of a pair p_i to receive high or low ratings. For example, the pair {*salt*, *pepper*} generally tends to receive high ratings compared to the pair {*candy*, *vinegar*}.

After removing the bias, the residual ratings matrix $\overline{\mathbf{R}}$ captures the fine, subjective user preferences that we aim to learn by factorizing the matrix to uncover latent patterns. Due to the large amount of missing ratings in $\overline{\mathbf{R}}$, it is infeasible to apply classical factorization techniques such as singular value decomposition. Instead, we learn a data-driven factorization based only on the *known entries* in $\overline{\mathbf{R}}$. This approach has been shown to lead to better results in matrix completion or factorization problems compared to imputation of the missing values (Canny, 2002; Koren, 2008). We express $\overline{\mathbf{R}}$ as the product of an object-pair factors matrix $\mathbf{V}^\top$, and a user factors matrix $\mathbf{W}$ of sizes $M \times K$ and $K \times N$, respectively. Each column $\mathbf{v}_i$ of $\mathbf{V}$ is a K-dimensional factors vector corresponding to an object-pair p_i. Similarly, each column $\mathbf{w}_j$ in $\mathbf{W}$ is a K-dimensional factors vector associated with a user u_j. We compute the residual rating $\overline{r}_{ij}$ as the dot product of the factor vectors for object-pair p_i and user u_j, i.e.,

$$\overline{r}_{ij} \;=\; \mathbf{v}_i^\top \cdot \mathbf{w}_j. \tag{3.7}$$

The vectors $\mathbf{v}$ and $\mathbf{w}$ are low-dimensional projections capturing latent characteristics of the pairs and users, respectively. Users who are close to each other in that space are similar with respect to some property. For example, some users could prefer to group objects together based on their shape, whereas others do so based on their function.

Accordingly, our prediction $\hat{r}_{ij}$ for the rating of an object-pair p_i by a user u_j is expressed as

$$\begin{aligned}\hat{r}_{ij} &= b_{ij} + \overline{r}_{ij} \\ &= \mu + b_i + b_j + \mathbf{v}_i^\top \cdot \mathbf{w}_j.\end{aligned} \tag{3.8}$$

We learn the biases and factor vectors from all available ratings in $\mathbf{R}$ by formulating an optimization problem. The goal is to minimize the difference between the observed ratings r_{ij} made by users and the predictions $\hat{r}_{ij}$ of the system over all known ratings. Let the error associated with rating r_{ij} be

$$e_{ij} = r_{ij} - (\mu + b_i + b_j + \mathbf{v}_i^\top \cdot \mathbf{w}_j). \tag{3.9}$$

We jointly learn the biases and factors that minimize the error over all known ratings, i.e.,

$$b_*, \mathbf{V}, \mathbf{W} = \underset{b_*, \mathbf{V}, \mathbf{W}}{\arg\min} \sum_{i=1}^{M} \sum_{j \in \mathcal{J}_i} (e_{ij})^2 + \frac{\lambda}{2}(b_i^2 + b_j^2 + \|\mathbf{v}_i\|^2 + \|\mathbf{w}_j\|^2), \tag{3.10}$$

where b_* denotes all object-pair and user biases and λ is a regularizer. To do so, we use L-BFGS optimization with a random initialization for all variables (Nocedal, 1980). At every step of the optimization, we update the value of each variable based on the error gradient with respect to that variable, which we derive from Eq. (3.10).

3.2.2. Probing and Predicting for New Users

In this section, we describe our approach for predicting the preferences of a new user based on the factorization model learned in the previous section. After learning the biases and factor vectors for all users and object-pairs (Eq. (3.10)), we can use Eq. (3.8) to predict the requested rating $\hat{r}_{ij}$ of a user u_j for an object-pair p_i that she has not rated before. However, this implies that we have already learned the bias b_j and factors vector $\mathbf{w}_j$ associated with that user. In other words, at least one entry in the j-th column of $\mathbf{R}$ should be known. The set of known preferences for a certain user, used for learning her model, are sometimes referred to as *probes* in the recommender system literature. In this work, we use *probing* to refer to the process of eliciting knowledge about a new user.

3.2.2.1. Probing

In the context of a tidy-up service robot, we envision two strategies to do so. In the first probing approach, the robot infers some preferences of the user based on how she has previously sorted objects in the containers $\mathcal{C}$ in the environment. By detecting the objects it encounters there, the robot can infer the probe rating for a certain object-pair based on whether the two objects are in the same container or not in the observed

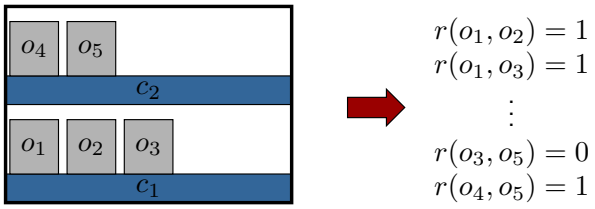


Figure 3.3.: A simple illustration of the probing process by which the robot can infer some preferences for a new user. We set a rating of 0 for a pair of objects that the robot observes to be in different containers, and a rating of 1 for those in the same container. Using these ratings, we learn a model of the user to predict her preferences for other object-pairs.

state $\mathbf{s}$. In other words, we set $r_{ij} = f_{\mathcal{R}}(p_i, \mathbf{s})$ for all object-pairs p_i that the robot observes in the environment (Eq. (3.1)) and fill the corresponding entries in the user's column with the inferred ratings. Figure 3.3 illustrates this with an example.

In the second probing approach, we rely on actively querying the user about her preferences for a set of object-pairs. For this, we rely on simple, out-of-the-box user interface solutions such as a voice- or text-based interface through which the user can provide ratings. For instance, the robot would query the user for her preference with respect to placing *sugar* and *salt* in the same container and set the corresponding rating to 1 if that is what she said. We assume the robot is able to query the user with a maximum of P probe ratings. One naive approach is to acquire probes by randomly querying the user about P object-pairs. However, we aim at making accurate predictions with as few probes as possible. Thus, we propose an efficient strategy based on insights into the factorization of Section 3.2.1. The columns of the matrix $\mathbf{V}$ can be seen as a low dimensional projection of the rating matrix capturing the similarities between object-pairs; object-pairs that are close in that space tend to be treated similarly by users. We therefore propose to cluster the columns of $\mathbf{V}$ in P groups, randomly select one column as a representative from each cluster, and query the user about the associated object-pair. For clustering, we use k-means with P clusters (MacQueen (1967)). In this way, the selected queries to the users capture the complete spectrum of preferences.

The nature of a collaborative filtering system allows us to continuously add probe ratings for a user in the ratings matrix by observing how objects are organized in the environment or by active querying as needed. This results in a flexible, lifelong learning approach that enables the robot to continuously update its knowledge about the user.

3.2.2.2. Inferring a New User's Preferences

After acquiring probes for the new user, we can append her ratings column to the ratings matrix and learn her biases and factors vector along with those of all object-pairs and

other users in the system as in Eq. (3.10). In practice, we can avoid re-learning the model for all users and object-pairs known in the system. Note that the computation of the factorization will require more time as the number of known ratings in $\mathbf{R}$ increases or for higher values of K. Here, we propose a more efficient technique suitable for inferring a new user's preferences given a previously learned factorization. After learning with all object-pairs and users in the database, we assume that all object-pair biases b_i and factor vectors $\mathbf{V}$ are fixed, and can be used to model the preferences of new users. This allows us to formulate a smaller problem to learn the bias b_j and factors vector $\mathbf{w}_j$ of the new user u_j based on the probe ratings we have for this user, i.e.,

$$\begin{aligned} b_j, \mathbf{w}_j &= \arg\min_{b_j, \mathbf{w}_j} \sum_{i \in \mathcal{I}_j} (e_{ij})^2 + \frac{\lambda}{2}(b_j^2 + \|\mathbf{w}_j\|^2), \\ &= \arg\min_{b_j, \mathbf{w}_j} \sum_{i \in \mathcal{I}_j} \Big(r_{ij} - (\mu + b_i + b_j + \mathbf{v}_i^\top \cdot \mathbf{w}_j)\Big)^2 + \frac{\lambda}{2}(b_j^2 + \|\mathbf{w}_j\|^2). \end{aligned} \tag{3.11}$$

Note that, in general, the inclusion of the ratings of a new user in $\mathbf{R}$ will affect the biases and factor vectors of the object-pairs. Whereas Eq. (3.10) represents the batch learning problem to update the model for all users and object-pairs, Eq. (3.11) assumes that the object-pair biases and factor vectors have already been learned from a sufficiently-large set of users that is representative of the new user. This is useful in a lifelong learning scenario where the robot can efficiently make predictions for a new user when solving a tidy-up task. With more knowledge accumulated about the new users, we can update the factorization model and biases for all object-pairs and users in a batch manner.

3.3. Mixture of Experts for Predicting Preferences of Unknown Objects

Thus far, we presented how our approach can make predictions for object-pairs that are known to the robot. In this section, we introduce our approach for computing predictions for an object-pair that no user has rated before, for example when the robot has to handle an object o_* that is not in $\mathcal{O}$. There, we cannot rely on standard collaborative filtering since we have not learned the similarity of the pair (through its factors vector) to others in $\mathcal{P}$.

Our idea is to leverage the known ratings in $\mathbf{R}$ as well as prior information about object similarities that we mine from the internet. The latter consists of object hierarchies provided by popular websites, including online supermarkets, stores, dictionaries, etc. Figure 3.4 illustrates parts of two example experts for a grocery scenario. Formally, rather than relying on one source of information, we adopt a *mixture-of-experts* approach where each expert $\mathcal{E}_i$ makes use of a mined hierarchy that provides information about similarities between different objects. The idea is to query the expert about the unknown

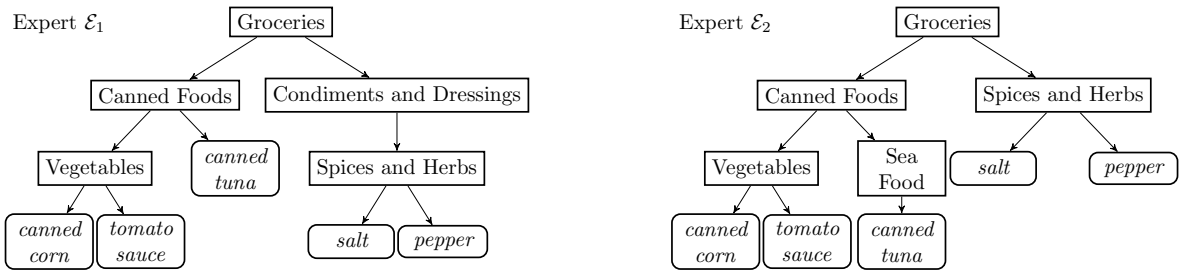


Figure 3.4.: Two examples of expert hierarchies used to compute the semantic similarities between object classes. For example, expert $\mathcal{E}_1$ on the left assigns a similarity ν of 0.4 to the pair $\{$*canned corn*, *canned tuna*$\}$, whereas $\mathcal{E}_2$ on the right assigns a similarity of 0.33 to the same pair, see Eq. (3.12).

object o_* and retrieve all the object-pair preferences related to it. The hierarchy is a graph or a tree where a node is an object and an edge represents an "is-a" relation.

When the robot is presented with a set of objects to organize that includes a new object o_*, we first ignore object-pairs involving o_* and follow our standard collaborative filtering approach to estimate preferences for all other object-pairs, i.e., Eq. (3.8). To make predictions for object-pairs related to the new object, we compute the similarity ν of o_* to other objects using the hierarchy graph of the expert. For that, we employ the *wup* similarity (Wu and Palmer, 1994), a measure between 0 and 1 used to find semantic similarities between concepts

$$\nu_{kl} = \frac{\text{depth}(LCA(o_k, o_l))}{0.5(\text{depth}(o_k) + \text{depth}(o_l))}, \tag{3.12}$$

where depth$(\cdot)$ is the depth of a node, and LCA denotes the lowest common ancestor. In the example of expert $\mathcal{E}_1$ in Figure 3.4-left, the lowest common ancestor of *canned corn* and *canned tuna* is Canned Foods. Their *wup* similarity based on $\mathcal{E}_1$ and $\mathcal{E}_2$ (Figure 3.4-right) is 0.4 and 0.33, respectively. Note that in general, multiple paths could exist between two object classes in the same expert hierarchy. For example, *coffee* could be listed under both Beverages and Breakfast Foods. In such cases, we take the path (LCA) that results in the highest *wup* measure for the queried pair.

Given this similarity measure, our idea is to use the known ratings of objects similar to o_* in order to predict the ratings related to it. For example, if *salt* is the new object, we can predict a rating for $\{$*salt*, *coffee*$\}$ by using the rating of $\{$*pepper*, *coffee*$\}$ and the similarity of *salt* to *pepper*. We compute the expert rating $\hat{r}_{\mathcal{E}_i}(o_*, o_k)$ for the pair $\{o_*, o_k\}$ as the sum of a baseline rating, taken as the similarity ν_{*k}, and a weighted mean of the residual ratings for similar pairs, i.e.,

$$\hat{r}_{\mathcal{E}_i}(o_*, o_k) = \nu_{*k} + \eta_1 \sum_{l \in \mathcal{L}} \nu_{*l} \; (r(o_l, o_k) - \nu_{lk}), \tag{3.13}$$

where $\eta_1 = 1/\sum_{l\in\mathcal{L}} \nu_{*l}$ is a normalizer, and $\mathcal{L}$ is the set of object indices such that the user's rating of pair $\{o_l, o_k\}$ is known. In other words, we rely on previous preferences of the user ($r(o_l, o_k)$) combined with the similarity measure extracted from the expert. The expert hierarchy captures one strategy for organizing the objects by their similarity. If this perfectly matches the preferences of the user, then the sum in Eq. (3.13) will be zero, and we simply take the expert's baseline ν_{*k} when predicting the missing rating. Otherwise, we correct the baseline based on how much the similarity measure deviates from the known ratings of the user.

Accordingly, each of our experts predicts a rating using its associated hierarchy. We compute a final prediction $\hat{r}_{\mathcal{E}_*}$ as a combined estimate of all the expert ratings:

$$\hat{r}_{\mathcal{E}_*}(o_*, o_k) = \eta_2 \sum_i w_i \, \hat{r}_{\mathcal{E}_i}(o_*, o_k), \tag{3.14}$$

where $w_i \in [0, 1]$ represents the confidence of $\mathcal{E}_i$, $\mathcal{E}_*$ denotes the mixture of experts, and $\eta_2 = 1/\sum_i w_i$ is a normalizer. We compute the confidence of expert $\mathcal{E}_i$ as $w_i = \exp(-e_i)$, where e_i is the mean error in the expert predictions when performing a leave-one-out cross-validation on the known ratings of the user as in Eq. (3.13). We set this score to zero if it is below a threshold, which we empirically set to 0.6 in our work. Moreover, we disregard the rating of an expert if o_* cannot be found in its hierarchy, or if all associated similarities ν_{*l} to any relevant object o_l are smaller than a threshold $\nu_{\min}$.

Note that in general, both objects in a new pair could have been previously encountered by the robot separately, but no rating is known for them together. When retrieving similar pairs to the new object-pair, we consider the similarities of both objects in the pair to other objects. For example, we predict the rating of $\{$*sugar*, *coffee*$\}$ by considering the ratings of both $\{$*flour*, *coffee*$\}$ and $\{$*sugar*, *tea*$\}$.

3.4. Grouping Objects Based on Predicted Preferences

The approach we presented so far in this chapter enables the robot to predict the missing preferences of a user with respect to both known (Section 3.2) and novel (Section 3.3) objects. In this section, we describe our approach for organizing the objects into the available containers based on the predicted preferences.

Specifically, we aim to solve the problem in Section 3.1.3 by computing a goal state $\mathbf{s}^*$ that maximizes the preference $\Psi(\mathbf{s}^*)$ of the user (Eq. (3.2)). For this, we first abstract away from computing arrangements in terms of object poses and instead determine which object-pairs p_i should be grouped together ($f_{\mathcal{R}}(p_i, \mathbf{s}^*) = 1$) or separated ($f_{\mathcal{R}}(p_i, \mathbf{s}^*) = 0$). In general, finding a partitioning of the objects such that all pairwise constraints are satisfied is non-trivial. For example, the user may have a high preference for $\{$*pasta*, *rice*$\}$ and for $\{$*pasta*, *tomato sauce*$\}$, but a low preference for $\{$*rice*, *tomato sauce*$\}$. Therefore, we aim at satisfying as many of the preference constraints as possible when grouping

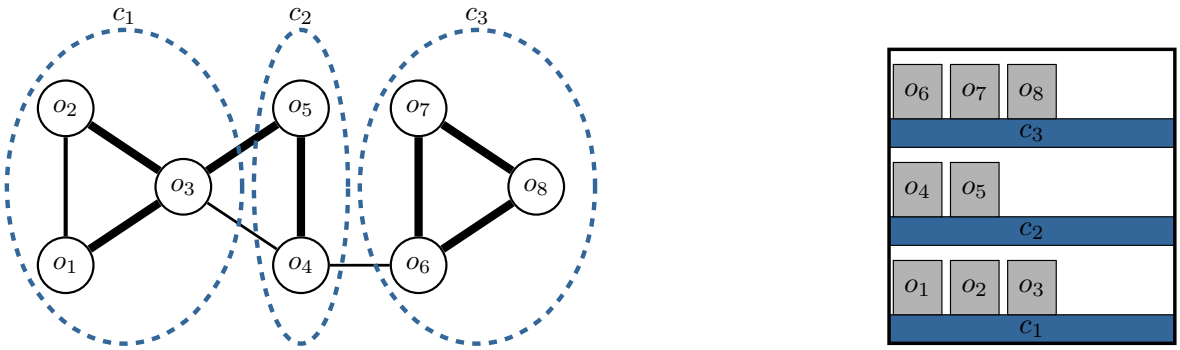


Figure 3.5.: Left: a graph depicting the relations between objects. Nodes depict objects whereas the weights (different edge thickness) correspond to the pairwise ratings. We partition the graph into subgraphs using spectral clustering. Right: we assign objects in the same subgraph to the same container.

the objects into $C' \leq C$ containers, where C is the total number of containers the robot can use.

We propose to tackle this problem from the perspective of graph partitioning. For this, we first construct a weighted graph in which the nodes represent the objects and each edge weight is the rating of the corresponding object-pair, see Figure 3.5. Accordingly, we formulate the subdivision of objects into C' containers as a problem of partitioning the graph into C' subgraphs such that the cut (the sum of the weights between the subgraphs) over all pairs of subgraphs is minimized. This is called the minimum k-cut problem (Goldschmidt and Hochbaum, 1994). Hence, we approximate the problem in Eq. (3.3) by translating it into one that aims to minimize separating pairs of objects for which the predicted rating is high.

However, finding the optimal partitioning of the graph into $C' \leq C$ subgraphs is NP-hard. In practice, we efficiently solve this problem by using a spectral clustering approach (Chung, 1996). The main idea is to partition the graph based on the eigenvectors of its Laplacian matrix, $\mathbf{L}$, as this captures the underlying connectivity of the graph. Let $\mathbf{C}$ be the matrix whose columns are the first C' eigenvectors of $\mathbf{L}$. We represent each object by a row of the matrix $\mathbf{C}$, i.e., a C'-dimensional point, and apply k-means clustering using C' clusters to get a final partitioning of the objects. To estimate the best number of clusters, we implement a self-tuning heuristic which sets the number of clusters C' based on the position of the biggest eigengap from the decomposition of $\mathbf{L}$, which typically indicates a reliable way to partition the graph based on the similarities of its nodes. A good estimate for the eigengap position is the number of eigenvalues of $\mathbf{L}$ that are approximately zero (von Luxburg, 2007; Zelnik-Manor and Perona, 2004). If there exist less containers in the environment than this estimate, we use all C containers for partitioning the objects.

Finally, the computed partitioning of objects represents an object arrangement that captures the preferences of the user on a semantic level. On the low, geometric level, there typically are numerous goal states $\mathbf{s}^*$ that achieve this partitioning. In this chapter, we focus on predicting the semantic preferences of the user with respect to organizing objects and not on computing the pose of each object in its target container. Therefore, we rely on existing task and motion planners to compute a feasible state $\mathbf{s}^*$ as well as a plan for the robot to manipulate the objects and achieve the desired arrangement.

3.5. Experimental Evaluation

In this section, we present the experimental evaluation of our approach by testing it on two tidy-up scenarios. We first demonstrate different aspects of our approach for a simple scenario of organizing toys in boxes based on a small dataset with 15 survey participants. In the second scenario, we address sorting grocery items on shelves, and provide an extensive evaluation based on ratings we collected from over 1,200 users using crowdsourcing. We demonstrate that: *i*) users indeed have different preferences with respect to sorting objects when tidying up, *ii*) our approach can accurately predict personal user preferences for organizing objects (Section 3.2.1), *iii*) we are able to efficiently and accurately learn a model for a new user's preferences based on previous training users (Section 3.2.2), *iv*) our mixture of experts approach enables making reasonable predictions for previously unknown objects (Section 3.3), *v*) our approach is suitable for life-long learning of user preferences, improving with more knowledge about different users, *vi*) our object partitioning approach based on spectral clustering can handle conflicting pairwise preferences and is flexible with respect to the number of available containers (Section 3.4), and *vii*) our approach is applicable on a real tidy-up robot scenario.

In the following experiments, we evaluate our approach using two different methods for acquiring probe ratings, and compare our results to different baselines. For that, we use the following notation:

- CF refers to our collaborative filtering approach for learning user preferences, as described in Section 3.2.1. When selecting probes to learn for a new user, we do so by clustering the object-pairs based on their learned factor vectors in order to query the user for a range of preferences, see Section 3.2.2.1.

- CF-rand selects probes randomly when learning for a new user and then uses our collaborative filtering approach to make predictions as in Section 3.2.1.

- CF-rand$'$ selects probes randomly and learns the preferences of a new user based on the object-pair biases and factor vectors learned from previous users as in Section 3.2.2.2.

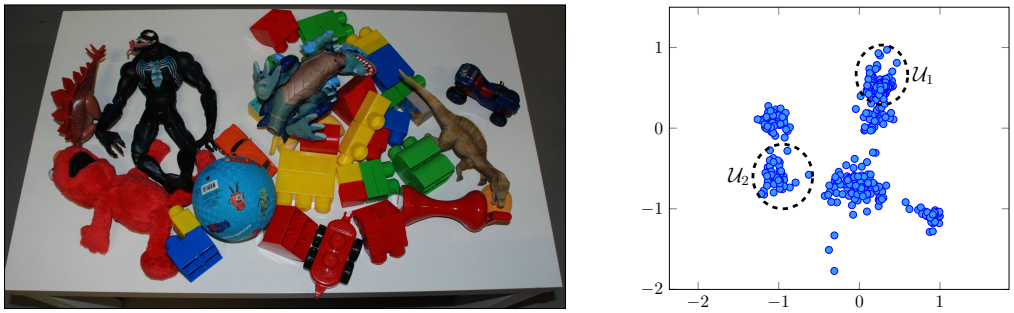


Figure 3.6.: Left: we considered a scenario of organizing toys in boxes. Right: a visualization of user tastes with respect to organizing toys, where we plot the user factor vectors projected to the first two dimensions. For example, the cluster $\mathcal{U}_1$ corresponds to users who grouped all building blocks together in one box. Cluster $\mathcal{U}_2$ corresponds to users who separated building blocks into standard bricks, car-shaped blocks, and miscellaneous.

- Baseline-I uses our probing approach as in CF, and then predicts each unknown pair rating as the mean rating over all users who rated it.

- Baseline-II selects probes randomly and then predicts each unknown pair rating as the mean rating over all users.

In all experiments, unless stated otherwise, we set the number of factor dimensions to $K = 3$ and the regularizer to $\lambda = 0.01$. As part of our implementation of Eq. (3.10) and Eq. (3.11), we rely on the L-BFGS implementation by Okazaki[1]. For our mixture-of-experts approach, unless otherwise stated, we set the *wup* similarity threshold $\nu_{\min}$ to 0.4, see Section 3.3. Finally, in this work, we assume that the robot is equipped with suitable techniques for recognizing the objects of interest. In our experiments, we relied on fiducial markers attached to the objects, and also implemented a classifier that recognizes grocery items by matching visual features extracted from the scene to a database of product images.

3.5.1. Task 1: Organizing Toys

In this experiment, we asked 15 people to sort 26 different toys in boxes, see Figure 3.6-left. This included some plush toys, action figures, a ball, cars, a flashlight, books, as well as different building blocks. Each participant could use *up to* six boxes to sort the toys. Overall, four people used four boxes, seven people used five boxes, and four people used all six available boxes to sort the toys.

[1] `http://www.chokkan.org/software/liblbfgs/`

We collected these results in a ratings matrix with 15 user columns and 325 rows representing all pairs of toys. Each entry in a user's column is based on whether the user placed the corresponding objects in the same box or not, see Section 3.2.2.1. For a fine quantification, we used these ratings to bootstrap a larger ratings matrix representing a noisy version of the preferences with 750 users. For this, we randomly selected 78 ratings out of 325 from each column. We repeated this operation 50 times for each user and constructed a ratings matrix of size 325×750 where 76% of the ratings are missing.

As a first test, we computed a factorization of the ratings matrix as described in Section 3.2.1. Figure 3.6-right shows the user factors $\mathbf{W}$ projected to the first two dimensions, giving a visualization of the user tastes. For example, the cluster of factors labeled $\mathcal{U}_1$ corresponds to users who grouped all building blocks together in one box.

3.5.1.1. Predicting User Preferences for Pairs of Toys

We evaluated our approach for predicting the preferences of the 15 participants by using the partial ratings in the matrix we constructed above. For each of the participants, we queried for the ratings of P probes. We hid all other ratings from the user's column and predicted them using the ratings matrix and our approach. We rounded each prediction to the nearest integer on the rating scale [0,1] and compared it to the ground truth ratings. We evaluated our results by computing the precision, recall, and F-score of our predictions with respect to the two rating classes: *no* ($r = 0$), and *yes* ($r = 1$). We set the number of probes to $P = 50, 100, \ldots, 300$ known ratings, and repeated the experiment 20 times for each value, selecting different probes in each run. The mean F-scores of both rating classes, averaged over all runs are shown in Figure 3.7-left.

Both collaborative filtering techniques outperform baselines I and II. By employing the same strategy for all users, these baselines are only able to make good predictions for object-pairs that have a unimodal rating distribution over all users, and cannot generalize to multiple tastes for the same object-pair, achieveing a mean F-score of 0.89. Using CF and CF-rand significantly improves the mean F-score to around 0.98 (two-sample t-test, p-value < 0.01).

3.5.1.2. Sorting Toys into Boxes

We evaluated our approach for grouping toys into different boxes based on the predicted ratings in the previous experiment. For each user, we partitioned the objects into boxes based on the probed and predicted ratings as described in Section 3.4, and compared that to the original arrangement. We computed the success rate, i.e., the percentage of cases where we achieve the same number and content of boxes, see Figure 3.7-right. Our approach has a success rate of 80% at $P = 300$. As expected, the performance improves with the number of known probe ratings. On the other hand, even with $P = 300$ known ratings, Baseline-I and Baseline-II have a success rate of only 56% and

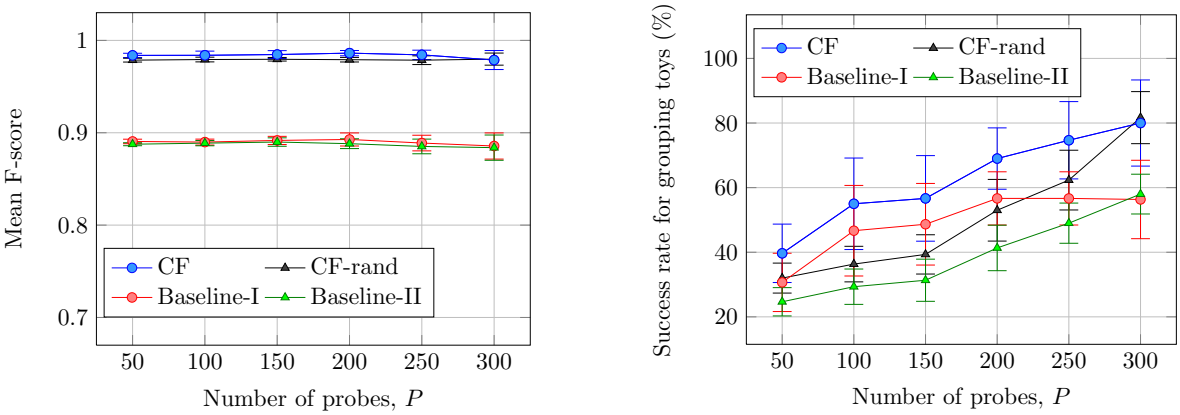


Figure 3.7.: Left: the mean F-score of the predictions of our approach (CF) in the toys scenario for different numbers of known probe ratings. We achieve an F-score of 0.98-0.99 on average over all predicted ratings. CF-rand selects probes randomly and then uses our approach for predicting. It is able to achieve an F-score of 0.98. On the other hand, baselines I and II are unable to adapt to multimodal user preferences. Right: the percentage of times our approach is able to predict the correct arrangement of boxes for sorting different toys. We outperform both baselines and improve with more probe ratings as expected, reaching a success rate of 80%. By selecting probes based on object-pair factor vectors, we are able to achieve higher success rates with less probes compared to CF-rand.

58%, respectively. Although we cannot report any statistically-significant difference in the performance between CF and CF-rand, our results indicate that CF achieves higher success rates with fewer probes. For example, CF-rand requires at least 200 known probe ratings on average to achieve a success rate over 50%, compared to only 100 known probe ratings using CF. The probes chosen by our approach capture a more useful range of object-pairs based on the distribution of their factor vectors, which is precious information to distinguish a user's taste.

3.5.1.3. Predicting Preferences for New Objects

We evaluated the ability of our approach to make predictions for object-pairs that no user has rated before (Section 3.3). For each of the 26 toys, we removed all ratings related to that toy from the ratings of the 15 participants. We predicted those pairs using a mixture of three experts and the known ratings for the remaining toys. We evaluated the F-scores of our predictions as before by averaging over both *no* and *yes* ratings. We based our experts on the hierarchy of an online toy store (toysrus.com), appended with three different hierarchies for sorting the building blocks (by size, color,

or function). The expert hierarchies contained between 165-178 nodes. For one of the toys (flash light), our approach failed to make predictions since the experts found no similarities to other toys in their hierarchy. For all other toys, we achieved an average F-score of 0.91 and predicted the correct box to place a new toy 83% of the time.

3.5.2. Task 2: Organizing Groceries

In this scenario, we considered the problem of organizing different grocery items on shelves. We collected data from over 1,200 users using a crowdsourcing service[2], where we considered a set of 22 common grocery item types, e.g., cans of beans, flour, tea, etc. We asked each user about her preferences for a subset of pairs related to these objects. For each pair, we asked the user if she would place the two objects together on the same shelf. Each user could answer with *no*, *maybe*, or *yes*, which we translated to ratings of 0, 0.5, and 1, respectively. We aggregated the answers into a ratings matrix $\mathbf{R}$ of size 179×1,284. Each of the user columns contains between 28 and 36 known ratings, and each of the 179 object-pairs was rated between 81 to 526 times. Overall, only around 16% of the matrix is filled with ratings, with the ratings distributed as in Table 3.1. Due to the three possible ratings and the noise inherent to crowdsourcing surveys, the ratings we obtained were largely multi-modal, see Figure 3.8 for some examples.

Table 3.1.: The distribution of ratings for the groceries scenario obtained through crowdsourcing. Overall, we gathered 37,597 ratings about 179 object-pairs from 1,284 users. For each object-pair, users indicated whether they would place the two objects on the same shelf or not.

	Rating Classes		
	no ($r = 0$)	*maybe* ($r = 0.5$)	*yes* ($r = 1$)
Rating Percentage	47.9%	29.2%	22.9%

3.5.2.1. Predicting User Preferences for Pairs of Grocery Items

We show that our approach is able to accurately predict user ratings of object-pairs using the data we gathered from crowdsourcing. For this, we tested our approach through 50 runs of cross-validation. In each run, we selected 50 user columns from $\mathbf{R}$ uniformly at random, and queried them with P of their known ratings. We hid the remaining ratings from the matrix and predicted them using our approach. We rounded each prediction to the closest rating (*no*, *maybe*, *yes*) and evaluated our results by computing the precision, recall, and F-score. Additionally, we compared the predictions of our approach (CF) to

[2]http://crowdflower.com/

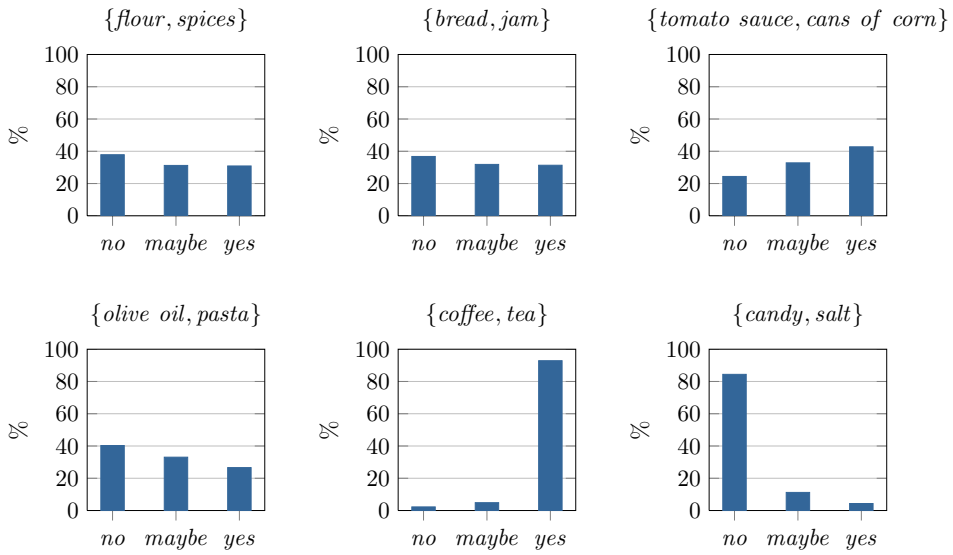


Figure 3.8.: Example distributions of the ratings given by users for different object-pairs. Each user could answer with *no* ($r = 0$), *maybe* ($r = 0.5$), or *yes* ($r = 1$) to indicate the preference for placing the two objects on the same shelf. The three possible rating classes, as well as the noise inherent to crowdsourcing surveys, resulted in multi-modal taste distributions. This highlights the difficulty of manually designing rules to guide the robot when sorting objects into different containers.

CF-rand, Baseline-I, and Baseline-II described above. The average F-scores over all runs and rating classes are shown in Figure 3.9-left for $P = 4, 8, \ldots, 20$. Both collaborative filtering approaches significantly outperform the baseline approaches for all values of P (two-sample t-test, p-value < 0.01), reaching a mean F-score of 0.63 at $P = 20$ known probe ratings. Baseline-I and Baseline-II are only able to achieve an F-score of 0.45 by using the same rating of a pair for all users. Note that, by employing our probing strategy, our technique is able to achieve an F-score of 0.6 with only 8 known probe ratings. On the other hand, CF-rand needs to query a user for the ratings of at least 12 object-pairs on average to achieve the same performance.

For a closer look at the performance with respect to the three rating classes, we select the results at $P = 12$ and show the per-class precision, recall, and F-score values for both CF and Baseline-I in Figure 3.10-left. Note that the baseline achieves its highest recall value for the *maybe* class since it uses the mean rating received by a specific object-pair to predict its rating for new users. On the other hand, we are able to achieve a similar recall (0.63) for the *maybe* class, as well as higher recall values for the *no* and *yes* classes despite the large degree of noise and the variance in people preferences in the training data. Our approach is able to achieve higher F-scores over all rating classes compared to the baseline. Out of the three classes, we typically achieved better

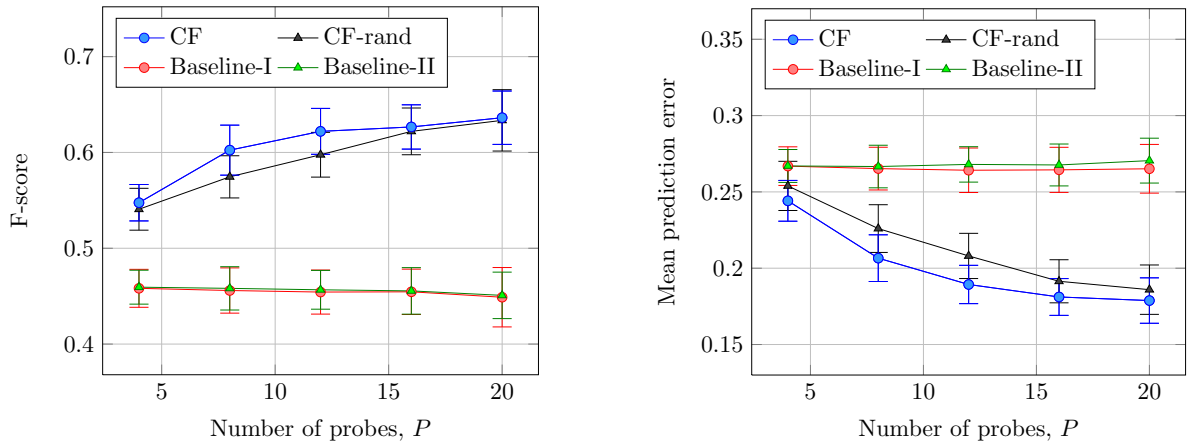


Figure 3.9.: Results for the scenario of organizing grocery items on different shelves. Left: the mean F-score of our predictions averaged over all rating classes *no*, *maybe*, and *yes*. Despite the large degree of multi-modality and noise in the user preferences we collected through crowdsourcing, our approach (CF) is able to achieve an F-score of 0.63 with 20 known probes and to outperform the baselines. Moreover, our performance improves with more knowledge about user preferences as expected. Right: the mean prediction error for different numbers of probes, P. The baselines are unable to cope with different modes of user preferences. They consistently result in a prediction error of around 0.27 irrespective of the number of probes. On the other hand, the mean prediction error using CF drops from 0.24 to 0.18 as P increases from 4 to 20. Using our probing technique, we are able to achieve a lower error with fewer probes compared to CF-rand.

scores for predicting the *no* class compared to *maybe* or *yes*. This is expected due to the distribution of the training ratings we gathered from the crowdsourcing data, see Table 3.1.

Additionally, we computed the prediction error (Eq. (3.9)) averaged over all experimental runs for each value of P, see Figure 3.9-right. The baselines are unable to cope with the different modes of user preferences, and consistently result in a prediction error of around 0.27 irrespective of the number of probes. On the other hand, the mean prediction error using CF and CF-rand drops from 0.24 to 0.18 and from 0.25 to 0.19 as P increases from 4 to 20, respectively. Note that, using our probing technique, we are able to achieve a lower error with fewer probes compared to CF-rand. This illustrates the importance of selecting more intelligent queries for users to learn their preferences. For a closer inspection of the prediction error, Figure 3.10-right shows the distribution of the error for our approach and Baseline-I given $P = 12$ probes. Our approach achieves an error of 0 for 64.62% of the predictions we make, compared to 49.78% only for

		no	*maybe*	*yes*
Baseline-I	Precision	0.71	0.34	0.79
	Recall	0.52	0.69	0.19
	F-score	0.60	0.46	0.31
CF	Precision	0.80	0.45	0.72
	Recall	0.72	0.63	0.49
	F-score	0.76	0.53	0.58

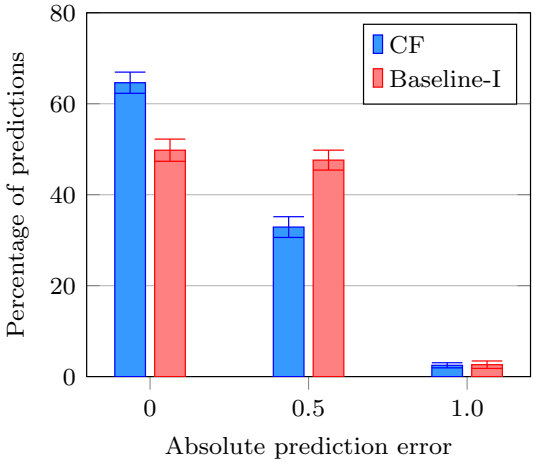


Figure 3.10.: Left: the detailed evaluation for the groceries scenario with $P = 12$ probes. Our approach results in higher F-scores across all rating classes compared to the baseline. Figure 3.9-left shows the mean F-score for different values of P. Right: the detailed distribution of the prediction errors using $P = 12$ probes, see Figure 3.9-right for the mean error for different values of P.

Baseline-I. Moreover, Baseline-I results in an absolute error of 0.5 (confusing *no*/*yes* with *maybe*) for 47.60% of the predictions, compared to 32.88% only for our approach. Finally, our approach and the baseline result in a prediction error of 1.0 (misclassifying *no* as *yes* or vice versa) for only 2.49% and 2.62% of the predictions, respectively.

3.5.2.2. The Effect of the Number of Latent Dimensions

In this experiment, we investigated the effect of varying the number of latent dimensions K used when learning the factorization of $\mathbf{R}$ on the quality of the learned model. We repeated the experiment in Section 3.5.2.1 for $K = 3, 6, 9, 15$. For each setting of K, we conducted 50 runs where, in each run, we selected 50 random user columns, queried them for P random probe ratings, and learned the factorization in Section 3.2.1 to predict the remaining ratings. As in the previous experiment, we evaluated the quality of predicting the unknown ratings by computing the average F-score for the *no*, *maybe*, and *yes* classes. Additionally, we computed the root mean square error (RMSE) for reconstructing the *known* ratings in $\mathbf{R}$ used in training, i.e.,

$$\text{RMSE} = \sqrt{\frac{1}{R}\sum_i \sum_{j \in \mathcal{J}_i} \left(r_{ij} - (\mu + b_i + b_j + \mathbf{v}_i^\top \cdot \mathbf{w}_j) \right)^2}.$$

The results are shown in Figure 3.11. When using larger values of K, we are able to reconstruct the known ratings in $\mathbf{R}$ with lower RMSE values. This is expected since we are computing a more accurate approximation of $\mathbf{R}$ when factorizing it into

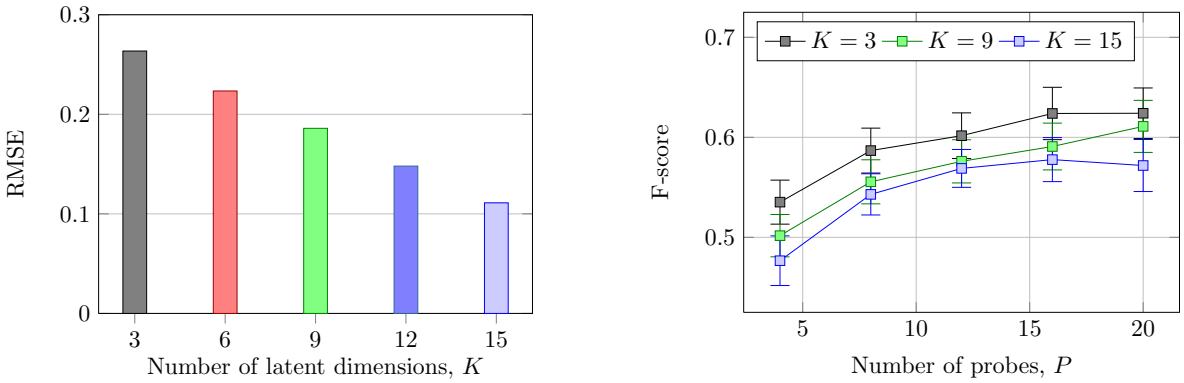


Figure 3.11.: We learned different factorizations of the ratings matrix $\mathbf{R}$ by varying the number of latent dimensions, K. For each learned model, we evaluated the RMSE when reconstructing the known ratings in $\mathbf{R}$ (left), and the F-score for predicting unknown ratings in $\mathbf{R}$ given different numbers of probes P for randomly selected user columns (right). Learning factors ($\mathbf{V}$ and $\mathbf{W}$) with larger dimensionality leads to reconstructing the known ratings in $\mathbf{R}$ with a higher fidelity (lower RMSE). However, this comes at the expense of over-fitting to the known ratings for users, leading to lower F-scores with larger K when predicting new ratings given the same number of probes, P.

higher-dimensional matrices ($\mathbf{V}$ and $\mathbf{W}$), thus capturing finer details in user preferences. However, this results in over-fitting (lower F-scores) when predicting unknown ratings, especially for lower values of P. In other words, for higher values of K, we need more probes per user when predicting unknown ratings, since we need to learn more factors for each user and object-pair. In general, the more known ratings we have in the user columns, the more sophisticated are the models that we can afford to learn.

Furthermore, we found interesting similarities between object-pairs when inspecting their learned biases and factor vectors. For example (for $K = 3$), users tend to rate $\{coffee, honey\}$ similarly to $\{tea, sugar\}$ based on the similarity of their factor vectors. Also, the closest pairs to $\{pasta, tomato\ sauce\}$ included $\{pancakes, maple\ syrup\}$ and $\{cereal, honey\}$, suggesting that people often consider whether objects can be used together or not. With respect to the biases (b_i) learned, object-pairs with the largest biases (rated above average) included $\{pepper, spices\}$, $\{pasta, rice\}$ and $\{cans\ of\ corn, cans\ of\ beans\}$. Examples of object-pairs with the lowest biases (rated below average) included $\{candy, olive\ oil\}$, $\{cereal, vinegar\}$, and $\{cans\ of\ beans, cereals\}$. On the other hand, object-pairs like $\{cans\ of\ corn, pasta\}$ and $\{pancakes, honey\}$ had a bias of almost 0.

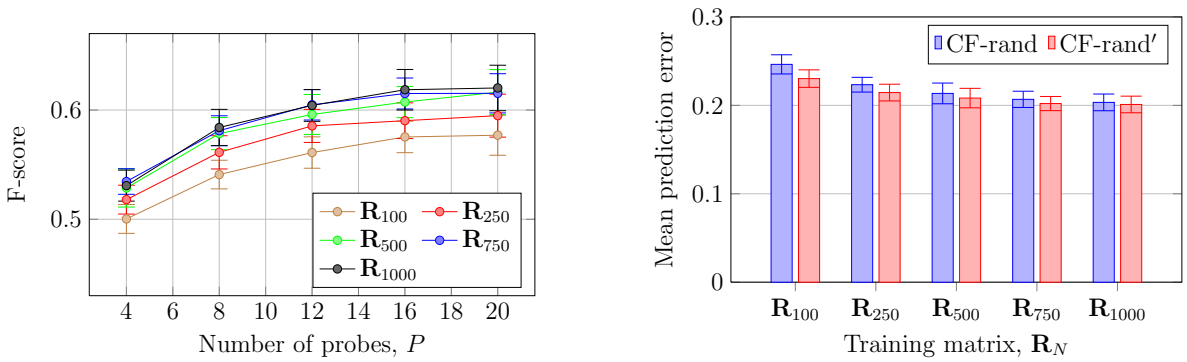


Figure 3.12.: Left: we tested our approach for learning the preferences of new users based on the object-pair biases and factors vectors learned from rating matrices $\mathbf{R}_N$ of different sizes. The results are shown for predicting with a set of 100 test users based on P random probe ratings each, averaged over 50 runs. The performance improves with more training users as expected, approaching the performance when training with all users, see Figure 3.9-left for comparison. Given sufficient users in the robot's database (≥ 750), we can infer the preferences of new users by assuming fixed object-pair biases and factor vectors without loss in prediction accuracy. Right: the prediction error given $P = 12$ probe ratings when inferring the preferences of test users given a previously-learned factorization (CF-rand′) compared to batch learning with the training and test users combined (CF-rand). As expected, the error for both approaches drops given more training users, converging to 0.20 for $\mathbf{R}_{750}$ and $\mathbf{R}_{1000}$, i.e., approaching the performance when training with the full $\mathbf{R}$, see Figure 3.9-right.

3.5.2.3. Learning of New User Preferences

In this experiment, we show that our approach is able to learn the preferences of new users based on the object-pair biases and factor vectors learned from previous users, see Section 3.2.2.2. We conducted an experiment similar to that in Section 3.5.2.1 using a random probing strategy. However, we first learned the biases b_i and factor vectors $\mathbf{V}$ using rating matrices $\mathbf{R}_{100}$, $\mathbf{R}_{250}$, ..., $\mathbf{R}_{1000}$, corresponding to 100, 250, ..., 1000 training users, respectively (Eq. (3.10)). We then used this model to compute the biases b_j and factor vectors $\mathbf{W}$ for a set of 100 (different) test users (Eq. (3.11)) and predict their missing ratings. As before, we repeated this experiment for different values P of known probe ratings for the test users. The prediction F-score averaged over 50 runs is shown in Figure 3.12-left. As expected, the performance improves given more training users for learning the b_i's and $\mathbf{V}$, converging to the performance when training with all user columns (compare to CF-rand in Figure 3.9-left). This validates that, given enough users in the robot's database, we can decouple learning a projection for the object-pairs

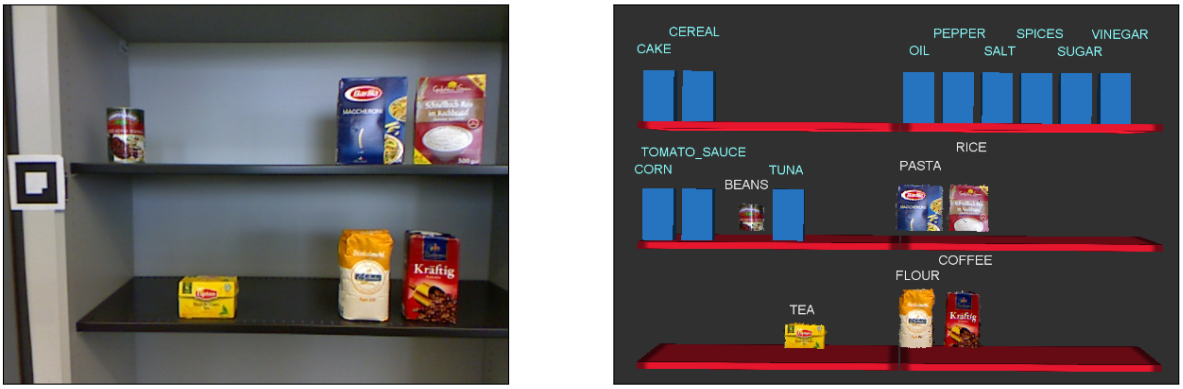


Figure 3.13.: An application of our approach that demonstrates how our predictions for a new user change based on how (probing) objects are arranged on the shelves. Left: the camera image of the scene. To label the objects, we relied on matching SIFT features from a database of images for the used products. Right: a visualization of the predicted preferred arrangement of other objects based on the corresponding learned model. Our method is able to infer the user's preferences by adapting to the perceived arrangement. For example, by moving *coffee* from the shelf containing *tea* to the one containing *flour*, the predicted arrangement separates *cake mix* and *sugar* and moves them to different shelves.

from the problem of learning the new users' biases and factor vectors.

Moreover, we compared the predictions using this approach (CF-rand′) to the standard batch approach that first appends the 100 new user columns to the training matrix and learns all biases and factor vectors collaboratively (CF-rand). The prediction error, averaged over the 50 runs, is shown in Figure 3.12-right for $P = 12$ probe ratings. As expected, the error for both approaches drops given more training users, converging to 0.20 for $\mathbf{R}_{750}$ and $\mathbf{R}_{1000}$, i.e., approaching the performance when training with the full $\mathbf{R}$ (compare to CF-rand in Figure 3.9-right for $P = 12$). Furthermore, with smaller training matrices, we observed a slight advantage in performance for CF-rand′. In other words, given fewer ratings, it might be advantageous to solve the smaller optimization problem in Eq. (3.11).

Probing and Learning for New Users Using our method, the time for computing the model for one new user (based on a previously-learned factorization) on a consumer-grade notebook was 10-20 ms on average, compared to about 4 s for batch learning with all 1,248 user columns ($K = 3$). To demonstrate the applicability of our approach (Section 3.2.2) in a real-world scenario, we conducted an experiment where we used a Kinect camera to identify a set of objects that we placed on shelves and used the

perceived pairwise ratings as probes for inferring a user's preference. For perception, we clustered the perceived point cloud to segment the objects, and relied on SIFT features matching using a database of product images to label each object. We learned the bias and factors vector for the user associated with this scene using the object-pairs model that we learned with our crowdsourcing data. Accordingly, we predicted the pairwise ratings related to objects that are not in the scene and computed the preferred shelves to place them on. Figure 3.13 shows an example where the left image shows the camera image of the scene, and the right image shows the corresponding computed shelf arrangement in the rviz visualization environment. Due to physical space constraints, we assume that each shelf is actually equally divided into two shelves.

3.5.2.4. Using a Mixture of Experts to Predict User Preferences

The purpose of this experiment is to evaluate our mixture-of-experts approach for predicting user preferences. For this, we defined three experts by mining the hierarchies of the groceries section of three large online stores (amazon.com, walmart.com, target.com). This includes up to 550 different nodes in the object hierarchy.

Predicting Preferences for New Objects In this experiment, we demonstrate that our mixture of experts approach is able to make reasonable predictions for previously unrated objects. For each of the 22 grocery objects, we removed ratings related to all of its pairs from $\mathbf{R}$, such that the typical collaborative filtering approach cannot make predictions related to that object. We used the mixture of experts to predict those ratings using the remaining ratings in each column and the expert hierarchies as explained in Section 3.3. The mean F-score over all users for three grocery objects is shown in Figure 3.14-left, where the mixture of experts is denoted by $\mathcal{E}_*$. We also show the individual expert results ($\mathcal{E}_1$-$\mathcal{E}_3$) and their corresponding baseline predictions ($\mathcal{E}'_1$-$\mathcal{E}'_3$). The baselines take only the *wup* similarity of two objects as the rating of the pair but do not consider the ratings of similar pairs made by the same user as our approach does. As we can see, the results of each individual expert outperform the baseline predictions. Note that $\mathcal{E}_*$ is able to overcome the shortcomings of the individual experts, as in the case of *rice*. There, $\mathcal{E}_1$ is unable to find similarities between *rice* and any of the rated objects, whereas $\mathcal{E}_2$ and $\mathcal{E}_3$ are able to relate it to *pasta* in their hierarchies. For two of the objects (*bread* and *candy*), we were unable to make any predictions, as none of the experts found similarities between them and other rated objects. For all other objects, we achieve an average F-score of 0.61.

Predicting Ratings for New Object-Pairs In this experiment, we demonstrate how we can use the mixture of experts above to extend our ratings matrix $\mathbf{R}$ with rows for object-pairs that no user rated in our crowdsourcing surveys. We created a new

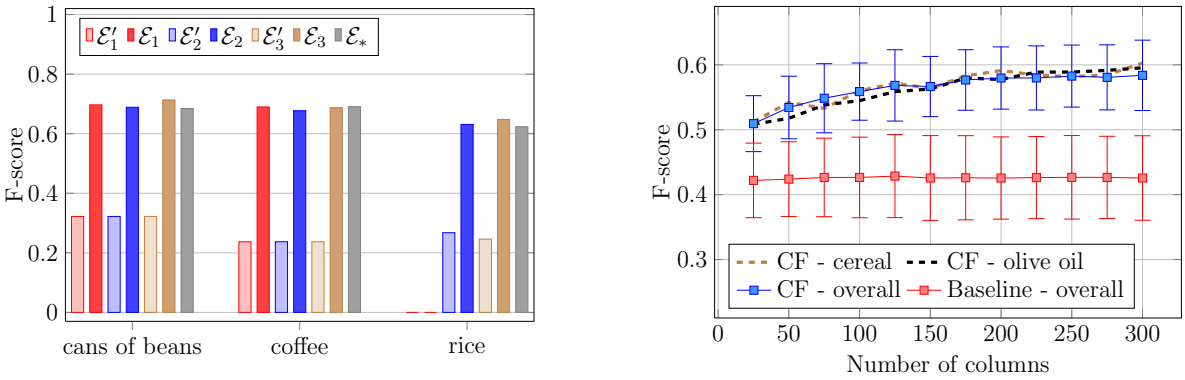


Figure 3.14.: Left: we predict preferences related to new objects by using a mixture of experts approach. The experts $\mathcal{E}_1$-$\mathcal{E}_3$ are based on the hierarchies of three online grocery stores. The mixture of experts $\mathcal{E}_*$ is a merged prediction of all three experts based on their confidence for a specific user. Therefore, it is able to recover if a certain expert cannot find similarities for a new object, as in the case of *rice*. The baselines $\mathcal{E}'_1$-$\mathcal{E}'_3$ make predictions based only on the semantic *wup* similarity of two objects without considering the ratings of similar pairs rated by the user, see Section 3.3. Right: the mean F-score for predicting the ratings for a new object vs. the number of training user columns who have rated pairs related to it. As soon as some users have rated pairs related to a new object, our collaborative filtering approach is able to make predictions about it. The performance improves with more users rating pairs related to the object.

ratings matrix $\mathbf{R}'$ of size 214×1284, i.e., with 35 additional object-pair rows. These included pairs related to the new object *cake mix*, as well as other object combinations. For each user column in the original $\mathbf{R}$, the mixture of experts used already-rated object-pairs to infer ratings for the new pairs. Figure 3.15 shows examples of rating distributions in the resulting ratings matrix for two object-pairs: {*cans of beans*, *sugar*} and {*cake mix*, *flour*}. Additionally, for each of them, we show the rating distributions of two of their most similar object-pairs that the experts used when making their predictions. In a later experiment, we use the resulting $\mathbf{R}'$ with this combination of crowdsourcing and expert-generated ratings to train a factorization model for predicting preferred arrangements of survey participants, see Section 3.5.2.6.

3.5.2.5. The Advantage of Learning from Different Users

In this experiment, we compare the performance of collaborative filtering, which leverages the ratings of different users when predicting for a specific user, to that of the mixture of experts $\mathcal{E}_*$, which only considers the known ratings of the user when making predictions

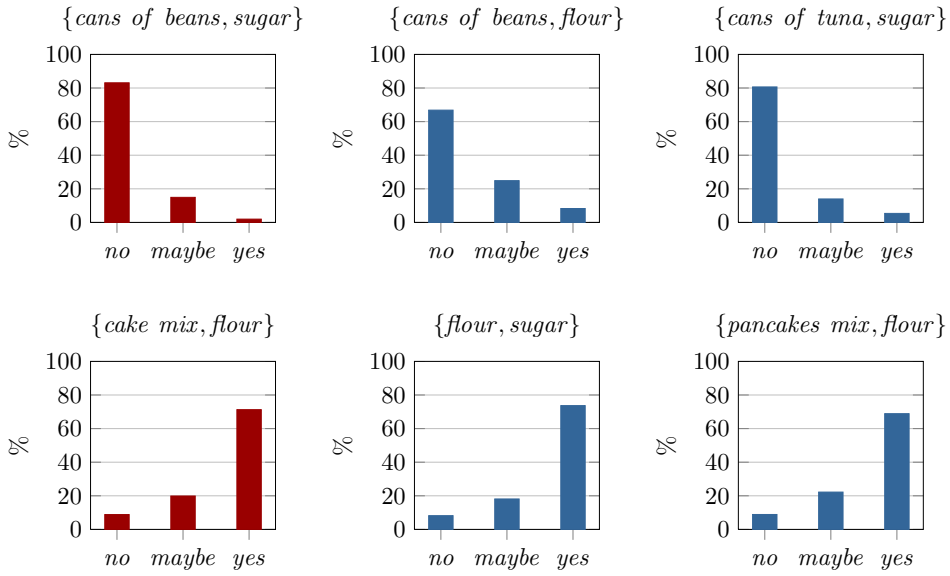


Figure 3.15.: The rating distributions depicted in red correspond to example object-pairs that no user had rated in the original crowdsourcing data. We generated those ratings using our mixture of experts approach based on the hierarchies of three online stores. In the case of $\{cans\ of\ beans, sugar\}$, the experts relied on how each user rated similar object-pairs such as $\{cans\ of\ beans, flour\}$ and $\{cans\ of\ tuna, sugar\}$. The rating distributions of those pairs (over all user columns who rated them) are depicted in blue on the same row. Similarly, in the case of $\{cake\ mix, flour\}$, the experts relied on the known ratings of $\{flour, sugar\}$ and $\{pancake\ mix, flour\}$.

for her. We performed the experiment described in Section 3.5.2.1 using the mixture of three experts. In each run of the experiment, we selected 50 random users and queried them randomly for P of their ratings. We then used the mixture of experts to predict the remaining ratings for each user by considering the similarities between object-pairs with unknown ratings and those with known ratings as described in Section 3.3. We repeated this 50 times for each value of P. Since the experts consider two object-pairs similar if their *wup* similarity ν is at least a threshold $\nu_{\min}$ according to their hierarchies, we repeated this experiment for three different values of $\nu_{\min}$ (0.2, 0.4, and 0.6). For each setting, we computed the percentage of cases in which the mixture of experts was able to make predictions by relating object-pairs using their similarities. For those cases, we computed the mean F-score over all rating categories as before. The results are shown in Figure 3.16, in which we also show the results of CF-rand from Section 3.5.2.1 for convenience.

The mixture of experts is able to make predictions for only a subset of the required ratings for each user, see Figure 3.16-left. As P increases, the experts are able to

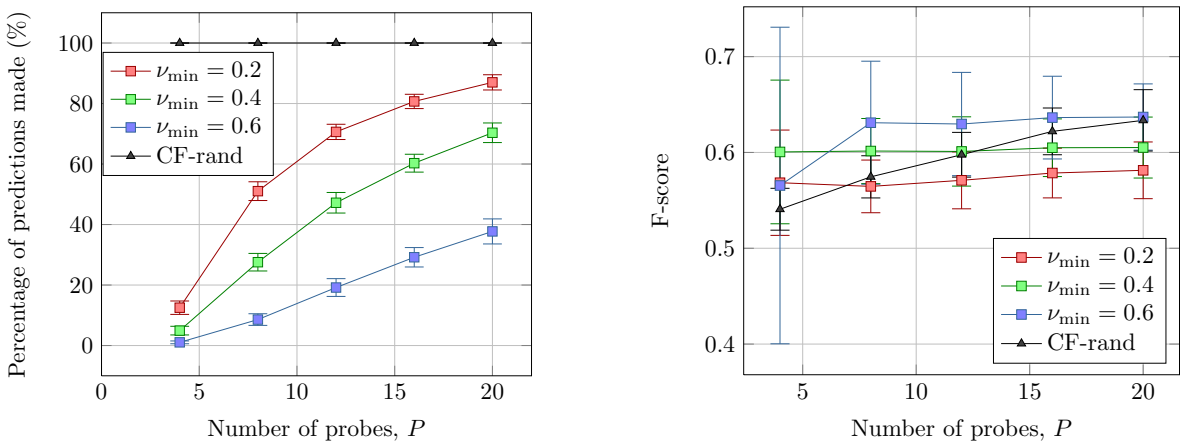


Figure 3.16.: We compared the predictions made by CF-rand and the mixture of experts $\mathcal{E}_*$ given different numbers P of known probe ratings. We tested $\mathcal{E}_*$ for three different settings of $\nu_{\min}$, the minimal *wup* similarity of any of its expert hierarchies to consider two object-pairs similar. Left: The mixture of experts does not consider the ratings made by other users, and is only able to make predictions for a subset of the desired object-pair ratings, i.e., those where it finds at least one similar object-pair where the user indicated a known preference. On the other hand, CF-rand is able to make predictions for all unknown ratings by learning a model from all users. Right: whereas $\mathcal{E}_*$ is able to make more predictions for lower values of $\nu_{\min}$, this is associated with a lower mean F-score for the predictions made. CF-rand is able to make predictions for all desired-object pairs and its performance improves with more probe ratings known for the user.

make more predictions as expected by finding more similar object-pairs to use for making predictions. However, for higher values of $\nu_{\min}$, it is more challenging for the experts to find object-pairs with known ratings that are considered similar enough to the queried object-pairs. Given four probes only, $\mathcal{E}_*$ is able to make predictions for less than 20% of the cases for all settings of $\nu_{\min}$. At $P = 20$, this percentage increases to 38% for $\nu_{\min} = 0.6$ and to 87% for $\nu_{\min} = 0.2$. In contrast to that, collaborative filtering (CF-rand) is able to make predictions for all queried object-pairs even with lower numbers of probes, since it leverages the ratings made by all other users in the training data.

Moreover, whereas the mixture of experts is able to make more predictions for lower threshold values of $\nu_{\min}$, this comes at a cost in terms of the quality of the predictions, see Figure 3.16-right. $\mathcal{E}_*$ achieves a mean F-score of 0.58 with $\nu_{\min} = 0.2$. On the other hand, when considering only the ratings of object-pairs with a similarity of at least $\nu_{\min} = 0.6$, $\mathcal{E}_*$ achieves a mean F-score of 0.64. Whereas CF-rand achieves lower F-score

values for small values of P, its performance increases steadily with the number of probes reaching 0.63 at $P = 20$, and it is consistently able to make predictions for all missing ratings as opposed to $\mathcal{E}_*$.

Finally, we conducted an experiment to show that the performance of our approach improves with more users in the system. For each object, we removed from $\mathbf{R}$ all columns with ratings related to that object. Over 20 runs, we randomly sampled ten different user columns (test users) from these and hid their ratings for pairs related to the object. We predicted those ratings using our approach (Section 3.2.1) by incrementally adding more columns of other (training) users who rated that object to the ratings matrix in increments of 25. We evaluated the mean F-score for the predictions for the test users. The results (CF - overall) are shown in Figure 3.14-right averaged over 20 different types of objects (those where we had at least 300 user ratings). We also show the improvement with respect to two of the objects individually. The performance of our approach improves steadily with the number of users who rate pairs related to a new object, as opposed to a baseline that updates the mean rating over all users and uses that for predicting. This shows that collaborative filtering is suitable for lifelong and continual learning of user preferences.

3.5.2.6. Assigning Objects to Shelves Based on Pairwise Preferences

In this section, we show that our approach is able to group objects into containers to satisfy pairwise preferences, see Section 3.4. We evaluated our approach in two settings. In the first, we compute the object groupings given ground truth pairwise ratings from users. In the second, we predict the pairwise ratings according to our approach and use those when grouping the objects on different shelves.

Arrangements Based on Ground Truth Ratings We conducted a qualitative evaluation of our approach for grouping objects into different containers based on *known* object-pair preferences. We asked a group of 16 people to provide their ratings (0 or 1) for 55 object-pairs, corresponding to all pairs for a set of 11 objects. For each participant, we then computed an object arrangement allowing our spectral clustering approach to use up to 6 shelves. We showed each participant the shelf arrangement we computed for them in the rviz visualization environment. We asked each of them to indicate whether they think the arrangement represents their preferences or not. They then had the choice to make changes to the arrangement by indicating which objects they would move from one shelf to another.

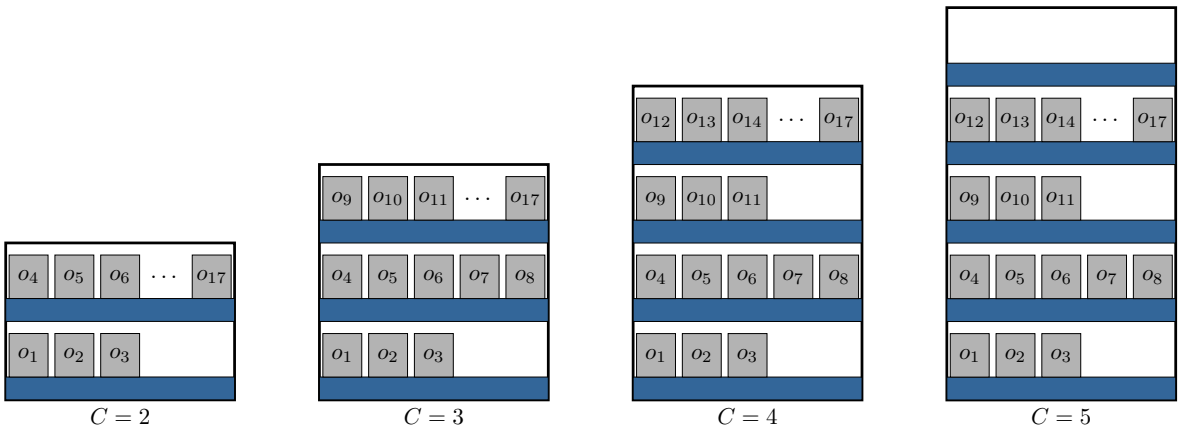
The results are shown in Figure 3.17. In five out of 16 cases, the participants accepted the arrangements without any modifications. Overall, the participants modified only two objects on average. Even given ground truth object-pair ratings from the users, there are often several inconsistencies in the preferences that can make it challenging for

Figure 3.17.: In a small survey with 16 participants, we computed groupings of 11 objects based on each participant's ground truth object-pair ratings. We presented the arrangement to each participant by visualizing it in a 3D visualization environment. Each participant then indicated which objects they would like to move in order to achieve a better arrangement. Despite inconsistencies in the pairwise preferences of the participants, we are able to compute reasonable object groupings that correspond to their preferences. On average, the participants modified the locations of only two objects.

Figure 3.18.: We asked survey participants to organize different types of grocery objects to test our approach for predicting their preferences, see Section 3.5.2.6.

the eigengap heuristic to estimate the best number of shelves to use. Nonetheless, we are able to compute reasonable groupings of objects. Moreover, the nature of our approach allows the robot to observe arrangements given by users to modify its knowledge about their preferences and use this when making new predictions in the future.

Arrangements Based on Predicted Ratings The goal of this experiment is to evaluate the performance of our approach for grouping objects based on *predicted* object-pair ratings. To collect ground truth data for arrangements, we asked 15 people to organize 17 different grocery items according to their preferences, using *up to* six shelves, see Figure 3.18. Four people grouped the items on four shelves, three people used five shelves, and eight people used all six shelves. Figure 3.1-left shows examples of arrangements



o_1 : *cake mix*	o_4 : *olive oil*	o_7 : *spices*	o_{10} : *coffee*	o_{13} : *corn*	o_{16} : *tomato sauce*
o_2 : *flour*	o_5 : *pepper*	o_8 : *vinegar*	o_{11} : *tea*	o_{14} : *pasta*	o_{17} : *tuna*
o_3 : *sugar*	o_6 : *salt*	o_9 : *cereal*	o_{12} : *beans*	o_{15} : *rice*	

Figure 3.19.: Our approach is able to adapt to the number of containers C available for organizing the objects. In this example, applying the self-tuning heuristic correctly estimates the best number of shelves to use ($C' = 4$), matching the original user preferences. Given more shelves in the scene ($C = 5$), our approach still prefers grouping the objects on four shelves, as this maximally satisfies the user's preferences. With only two or three shelves, our method attempts to satisfy the preferences of the user as much as possible by grouping the objects differently.

produced by the survey participants. We translated these arrangements to user rating columns with 0 or 1 ratings as described in Section 3.2.2.1. Figure 3.19 shows the arrangements our method computes for one of the participants (who used four shelves) when given all ground truth object-pair ratings of that participant. Given four or more shelves to use, we are able to reproduce the original object grouping of this user with $C' = 4$ shelves. The figure also shows how our approach adapts by merging some object groups together when given only two or three shelves to sort the objects.

Our goal is to evaluate the arrangements we compute for the 15 participants given only partial knowledge of their ratings, and based on a previously-learned model of object-pair biases b_i and factor vectors $\mathbf{V}$ from training users. To learn the model, we used the ratings matrix $\mathbf{R}'$ described in Section 3.5.2.4 above, as this covers all object-pairs relevant for the objects in this experiment. For each of the 15 participants, we simulated removing O random objects from their arrangement, and hid all ratings related to those objects. Using the remaining ratings as probes, we learned the bias b_j and factors vector $\mathbf{w}_j$ of each participant (Section 3.2.2.2), and predicted the missing

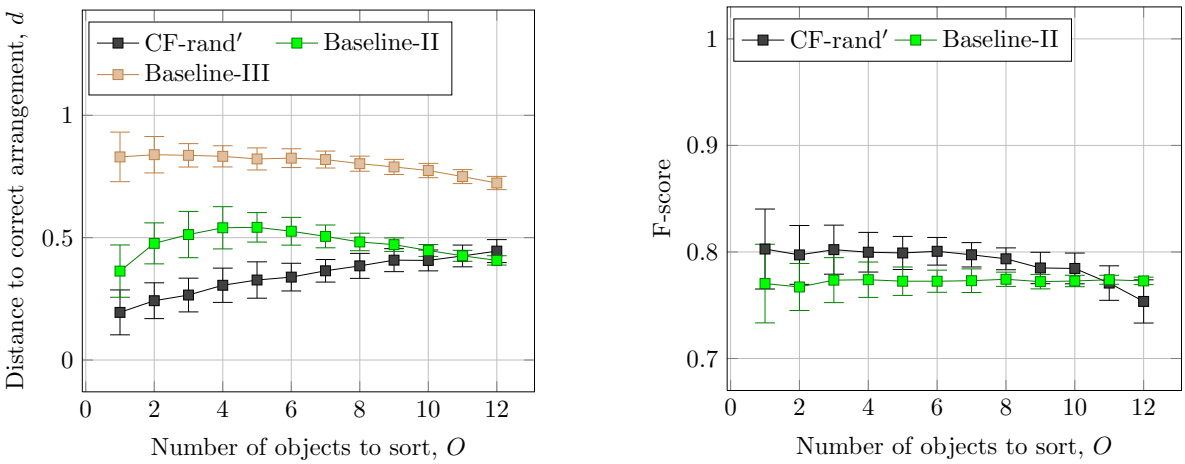


Figure 3.20.: We evaluated our approach (CF-rand′) for predicting the correct shelves for O randomly-selected objects (out of 17) given the shelves that the remaining objects have been assigned to. CF-rand′ predicts the missing object-pair ratings using our approach and then partitions the objects using our spectral clustering method. Baseline-II predicts the missing ratings as the mean rating over all training users, and uses our method for partitioning the objects. Baseline-III makes no predictions, but randomly assigns each object to one of the six shelves. Left: the mean error in the computed arrangements (ratio of misplaced objects). Right: the prediction F-score of the object-pair ratings averaged over the 0 and 1 categories. Our approach outperforms both baselines for $O \leq 10$. As O increases, the error in our predictions increases since there are less probe ratings based on the remaining objects on the shelves to infer the preferences of a user.

ratings accordingly. Finally, we used those ratings to compute an object arrangement for each participant, and compared it to their ground truth arrangement. We conducted this experiment for $O = 1, 2, 3, \ldots, 12$, repeating it 100 times for each setting.

We evaluated the results by computing an "edit distance" d between the computed and ground truth arrangements. We compute d as the minimum number of objects we need to move from one shelf to another to achieve the correct arrangement, divided by O. This gives a normalized error measure between 0 and 1 capturing the ratio of misplaced objects. The results (averaged over all runs) are shown in Figure 3.20-left, where we denote our method by CF-rand′. We compared our results to two baselines. The first is Baseline-II, which predicts the missing object-pair preferences using the mean ratings over all users in $\mathbf{R}'$ and then computes the object groupings using our approach. The second is Baseline-III, which makes no predictions but simply assigns each object to a random shelf. Figure 3.20-right also shows the mean F-score of our approach and Baseline-II averaged over the 0 and 1 rating categories.

Our approach outperforms both baselines for $O \leq 10$, and results in a mean error from 0.19 to 0.41, and an F-score from 0.80 to 0.78, as O changes from 1 to 10. The model we learned from $\mathbf{R}'$, the noisy crowdsourcing data from 1284 users augmented by ratings from a mixture of experts, is able to accurately predict the preferences of the new 15 survey participants. For the same values of O, Baseline-II achieves a mean error ranging between 0.36 and 0.54, and an F-score of 0.77. As O increases, the error in our predictions increases as expected, since we have less probe ratings based on the remaining objects on the shelves to infer the preferences of a new user. For $O > 11$, Baseline-II results in less error than our approach when computing arrangements as this baseline does not depend on the remaining objects to make predictions. On the other hand, using a random strategy for assigning objects to shelves (Baseline-III) resulted in an error above 0.72 for all values of O.

Predictions by Humans We conducted a qualitative evaluation to gauge the difficulty of the above task for humans. We asked 15 new participants (who did not take part in the previous surveys) to complete partial arrangements by predicting the preferences of the 15 users above, whom they do not know. Each participant solved six tests. In each test, we manually reconstructed an arrangement from the surveys on the six shelves, then we removed O objects randomly and placed them on a nearby table. We asked each participant to predict the preference of another user by inspecting the arrangement in front of them, and to finish sorting the remaining O objects accordingly. Each participant solved six such cases ($O = \{2, 4, \ldots, 12\}$), each corresponding to a different user. As before, we computed the error d, the ratio of objects that were placed on wrong shelves. The results are shown in Table 3.2. When presented with two objects to place only, most participants were able to predict the correct shelf for them based on the arrangement of the remaining objects. However, given four to twelve objects, the participants found the task increasingly difficult and misplaced between one forth to one third of the objects on average.

Table 3.2.: The error d in the arrangements produced by 15 participants whom we asked to sort O objects by predicting the preferences of users they do not know.

# objects, O	2	4	6	8	10	12
Error, d	0.07 ± 0.17	0.27 ± 0.25	0.24 ± 0.22	0.27 ± 0.18	0.33 ± 0.18	0.34 ± 0.17

3.5.2.7. Real Robot Experiments

We conducted an experiment to illustrate the applicability of our approach (Section 3.2.1) on a real tidy-up robot scenario using our PR2 robot platform, see Figure 3.21-left. We executed 25 experimental runs in which the task of the robot was to fetch two

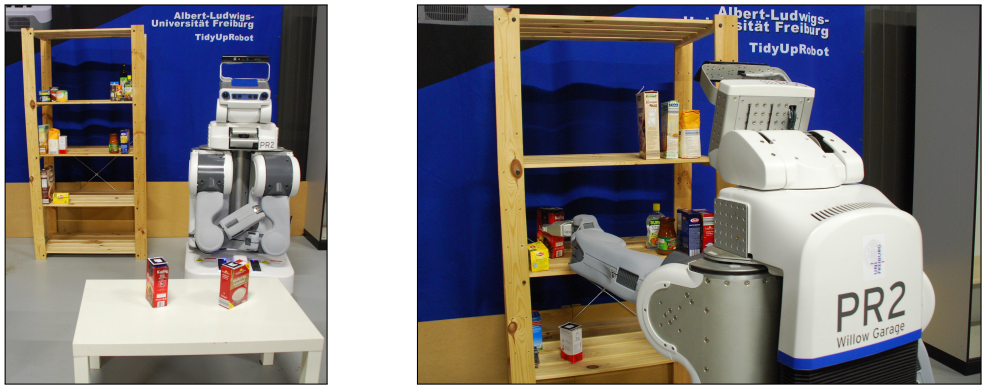


Figure 3.21.: Left: the robot has to assign the two objects that are on the table to shelves according to predicted user preferences. In this example, the robot places *coffee* on the same shelf as *tea*, and *rice* next to *pasta*. Right: an example where the robot places *coffee* next to *tea* and *sugar* next to *salt*.

objects from the table and return them to their preferred shelves as predicted by our approach. In each run, we arranged 15 random objects on the shelves according to the preferences of a random user from the survey we conducted in Section 3.5.2.6, and provided this information as probes for the robot to learn a model of that user (Section 3.2.2.1). The robot used this to predict the pairwise preferences related to the two objects on the table, which it recognized with its Kinect camera using unique fiducial markers we attached to the objects. It then computed an assignment of the objects to one of the shelves (Section 3.4), and expressed those assignments as planning goals in the form of logical predicates in PDDL (e.g., $\texttt{on}(\mathit{coffee}, \mathit{shelf}_2)$). To achieve those goals, we used a state-of-the-art planner (Dornhege and Hertle, 2013) to generate a plan for the robot to navigate to the table, grasp the detected objects, navigate to the shelves, and place the objects on their corresponding shelves (that may be empty or have objects on them) after detecting free space on them. For manipulation, we relied on an out-of-the-box motion planner. Additionally, we provided the robot with a 3D map of the environment for localizing itself, where we labeled the table and the six shelves. Overall, the robot predicted the correct shelf assignment for 82% of the objects using our approach, see Figure 3.21-right for an example where the robot successfully placed *coffee* on the same shelf as *tea*.

3.5.3. Summary and Discussion of Results

In this section, we present a brief summary of our results, highlighting the characteristics of our approach. Our experimental evaluation for two different tidy-up scenarios confirms the claim that users have varying preferences with respect to organizing objects, making it hard to provide the robot with a fixed strategy for all users. Our experiments demonstrate

that collaborative filtering overcomes this challenge by learning a model of preferences from all users collaboratively. They also supported the claim that our method is able to make accurate predictions for a user given partial knowledge of her preferences, and that the performance of our method improves with more probe ratings (Section 3.5.1.1 and Section 3.5.2.1), and with more users in the system (Section 3.5.2.5). This shows the suitability of our method for lifelong learning applications. Moreover, the results indicate that our method is able to leverage models learned from previous users in order to infer the preferences of a new user in an efficient manner (Section 3.5.2.3). Furthermore, our spectral clustering method is able to handle conflicting user preferences and take into account the number of containers available for organizing the objects (Section 3.5.1.2 and Section 3.5.2.6). We demonstrated the applicability of our techniques in practice through experiments with a real robot, see Section 3.5.2.7. We also highlighted the difficulty of this task through a qualitative survey in which we asked human volunteers to predict the preferences of other users, see Section 3.5.2.6.

In cases in which there are no ratings related to a specific object-pair in the training data, our mixture-of-experts method is able to make reasonable predictions using prior knowledge of object similarities that we mined from the Web. This allowed us to make predictions for new objects (Section 3.5.1.3 and Section 3.5.2.4), as well as to extend our ratings matrix with training ratings for new object-pairs (Section 3.5.2.4).

In this work, we focused on learning latent patterns for organizing objects based on their known semantic categories (e.g., book, coffee, pencil). One of the main advantages of our method is that it does not require an expert to define features to describe objects in order to learn how to organize them. However, this limits the degree to which the robot can exploit other aspects of the objects or the environment that could be relevant for the task. While outside the scope of this work, it would be interesting to extend our method in the future to take such features into consideration. For example, considering the geometry and physical properties (e.g., weight, size, color, etc.) of objects could lead to computing a more fine-grained prediction of organizational preferences. It would be interesting to combine content-based recommender system approaches that take such features into consideration with our collaborative filtering method, as in the case of the so-called hybrid recommender systems (Adomavicius and Tuzhilin, 2005).

Finally, there are cases in which the mixture of experts is unable to make predictions for a new object-pair if none of the experts is able to find sufficient similarities between the relevant objects in their hierarchies (Section 3.5.2.4). In practice, the robot could address such cases by querying the user for her preference related to such pairs using the probing techniques we presented in Section 3.2.2.1. As soon as ratings for a new object-pair have been appended to the ratings matrix, the robot can learn an updated model and use it when making predictions in the future.

3.6. Related Work

In this chapter, we presented a novel approach that enables a service robot to organize objects in a way that aligns with the preferences of its users. To the best of our knowledge, our contribution is the first in the literature to bridge the gap between the worlds of recommender systems and service robotics to allow a robot to tailor its behavior to a specific user. Indeed, equipping robots with the knowledge and skills needed to attend to complex chores in domestic environments has been the aim of researchers for years. Recent advances in perception, manipulation, planning, and control have enabled robots to perform a variety of chores that range from cleaning and tidying up to folding laundry (Saxena et al., 2008; Hess et al., 2012; Miller et al., 2012; Doumanoglou et al., 2014). However, as highlighted by a number of researchers, service robots should also be able to attend to such tasks in a manner that corresponds to the personal preferences of end users (Cakmak and Takayama, 2013; Forlizzi and DiSalvo, 2006; Dautenhahn et al., 2005; Pantofaru et al., 2012; Ray et al., 2008; Smarr et al., 2014; Cha et al., 2015). For example, the results of Pantofaru et al. (2012) show that people exhibit strong feelings with respect to robots organizing personal items, suggesting the need for the robot to ask humans to make decisions about where to store them (Pantofaru et al., 2012).

By leveraging the paradigm of collaborative filtering, our approach enables a robot to discover organizational patterns from a corpus of user preferences to compute preferred object arrangements when tidying up for a specific user. Accordingly, the robot is able to predict the preferred location (e.g., a specific shelf) to store an object by observing how the user has previously arranged other objects in the same environment. Several researchers have also leveraged the fact that our environments are rich with cues that can assist robots in various tasks that require reasoning about objects and their locations. For example, different works have addressed object classification or predicting the locations of objects using typical 3D structures in indoor environments or object-object relations such as co-occurrences in a scene (Joho et al., 2011; Aydemir and Jensfelt, 2012; Lorbach et al., 2014; Nyga et al., 2014; Kunze et al., 2014). However, in contrast to our work, these approaches do not consider the personal preferences of end-users. In the remainder of this section, we discuss prior work in the literature that is most relevant to the problem we addressed and the techniques we presented in this chapter.

Learning Object Arrangements and Placements Recently, Schuster et al. (2010) presented an approach for distinguishing clutter from clutter-free areas in domestic environments so that a robot can reason about suitable surfaces for placing objects. Related to that, the work of Jiang et al. (2012b) targets learning physically stable and semantically preferred poses for placing objects given the 3D geometry of the scene. Joho et al. (2012) developed a novel hierarchical nonparametric Bayesian model for

learning scenes consisting of different object constellations. Their method can be used to sample missing objects and their poses to complete partial scenes based on previously seen constellations. Other approaches have targeted synthesising artificial 3D object arrangements that respect constraints like physical stability or that are semantically plausible (Xu et al., 2002; Fisher et al., 2012). We view such works as complementary to ours, as we address the problem of learning preferred groupings of objects in different containers (e.g., shelves) for the purpose of tidying up. After predicting the preferred container for a specific object, our approach assumes that the robot is equipped with a suitable technique to compute a valid placement or pose of the object in that location. Moreover, as we explicitly consider organizing objects when tidying up, we do not reason about object affordances associated with various human poses and activities in the scene (e.g., cooking, working at a desk, etc.) when computing arrangements, as in the work of Jiang et al. (2012a) and Savva et al. (2014).

Related to our work, previous approaches have addressed learning organizational patterns from surveys conducted with different users. Schuster et al. (2012) presented an approach for predicting the location for storing different objects (e.g., cupboard, drawer, fridge, etc.) based on other objects observed in the environment. They consider different features that capture object-related properties (e.g., the purpose of an object or its semantic similarity to other objects) and train classifiers that predict the location at which an object should be stored. Similarly, Cha et al. (2015) explored using different features describing both objects and users to train classifiers for predicting object locations in user homes. Similar to Schuster et al. (2012), we also make use of a similarity measure based on hierarchies mined from the Web and use it for making predictions for objects for which we have no training data. However, in contrast to these works, our approach learns latent organizational patterns across different users in a collaborative manner and without the need for designing features that describe objects or users. Recently, Toris et al. (2015) presented an approach to learn placing locations of objects based on crowdsourcing data from many users. Their approach allows for learning multiple hypotheses for placing the same object, and for reasoning about the most likely frame of reference when learning the target poses. They consider different pick-and-place tasks such as setting a table or putting away dirty dishes, where the aim is to infer the final object configuration at the goal. Our approach is also able to capture multiple modes with respect to the preferred location for placing a certain object. In contrast to Toris et al. (2015), we target learning patterns in user preferences with respect to sorting objects in different containers.

Moreover, Brawner and Littman (2016) recently presented an approach that extends our work to incorporate contextual information about a kitchen environment to enable predicting item-location preferences for a user. In contrast to our work, they use features describing items (e.g., edible) and locations (e.g., low cabinet) whereas our approach assumes no such prior knowledge. Instead of partitioning the objects based on the

available number of containers, they combine item-item and item-location ratings to select the best location for storing an item.

Similar to the above works, we assume the robot is equipped with a map of the environment where all locations relevant to the task are already identified (e.g., shelves, drawers, etc.), which we refer to as containers. Previous work has focused on constructing semantic maps that are useful for robots when planning to solve complex household tasks (Vasudevan et al., 2007; Zender et al., 2008; Pangercic et al., 2012). However, in contrast to the above works, our method allows the robot to adapt to the available number of containers in a certain environment to sort the objects while satisfying the user's preferences as much as possible.

Service Robots Leveraging the Web Recently, several researches have leveraged the Web as a useful source of information for assisting service robots in different tasks (Tenorth et al., 2011; Kehoe et al., 2013). To cope with objects that are not in the robot's database, our method combines collaborative filtering with a mixture-of-experts approach based on object hierarchies we mine from online stores. This allows us to compute the semantic similarity of a new object to previously known objects to compensate for missing user ratings. The work by Schuster et al. (2012) has also utilized such similarity measures as features when training classifiers for predicting locations for storing objects. Pangercic et al. (2011) also leverage information from online stores but in the context of object detection. Kaiser et al. (2014) recently presented an approach for mining texts obtained from the Web to extract common sense knowledge and object locations for planning tasks in domestic environments. Moreover, Nyga et al. (2014) presented an ensemble approach where different perception techniques are combined in the context of detecting everyday objects.

Collaborative Filtering We predict user preferences for organizing objects based on the framework of collaborative filtering, a successful paradigm from the data mining community for making personalized user recommendations of products (Canny, 2002; Bennett and Lanning, 2007; Koren, 2008, 2010; Sarwar et al., 2001). Such techniques are known for their scalability and suitability for life-long learning settings, where the quality of the predictions made by the recommender system improves with more users providing their ratings. Outside the realm of customers and products, factorization-based collaborative filtering has recently been successfully applied to other domains including action-recognition in videos (Matikainen et al., 2012) and predicting drug-target interactions (Temerinac-Ott et al., 2015). Recently, Matikainen et al. (2013) combined a recommender system with a multi-armed bandit formulation for suggesting good floor coverage strategies to a vacuum-cleaning robot by modeling different room layouts as users. To the best of our knowledge, our work (Abdo et al., 2014b, 2015, 2016) was the first to leverage collaborative filtering for predicting personalized user

preferences in the context of service robotics. Recently, Brawner and Littman (2016) presented an approach that builds on our work to enable a robot to predict the preferred locations for organizing items in a kitchen. In addition to predicting ratings with respect to pairs of items, they present a context-aware recommender system that incorporates information about the environment such as location categories to enable predicting item-location preferences directly.

Crowdsourcing for Robotics To learn different user preferences, we collected data from many non-expert users using a crowdsourcing platform. Prior work has also leveraged crowdsourcing for data labeling or as an efficient platform for transferring human knowledge to robots (Deng et al., 2013; Kent et al., 2014). For example, Sorokin et al. (2010) utilized crowdsourcing to teach robots how to grasp new objects. Moreover, several researchers have used crowdsourcing to facilitate learning manipulation tasks from large numbers of human demonstrations (Chung et al., 2014; Toris et al., 2014, 2015; Ratner et al., 2015). In the context of learning user preferences, Jain et al. (2015) recently presented a new crowdsourcing platform where non-experts can label segments in robot trajectories as desirable or not. This is then used to learn a cost function for planning preferred robot trajectories in different indoor environments. Also in the context of user preferences, their prior work enables a robot to learn preferred manipulation trajectories by interacting with a teacher (Jain et al., 2013).

3.7. Conclusion

In this chapter, we presented a novel approach built on recommender system theory that enables robots to predict user preferences with respect to tidying up objects in containers such as shelves or boxes. To do so, we first predict pairwise object preferences of the user by formulating a collaborative filtering problem. This allows us to subdivide the objects in containers by modeling and solving a spectral clustering problem. Our approach is able to make predictions for new users based on partial knowledge of their preferences and a model that we learn collaboratively from several users. Furthermore, our technique allows for easily updating knowledge about user preferences, does not require complex modeling of objects or users, and improves with the amount of data, allowing for lifelong learning of user preferences. To deal with novel objects for which we have no data, we presented an approach that complements collaborative filtering with a mixture of experts based on object hierarchies mined from the Web. This enables the robot to relate these objects to similar ones it knows about. We trained our system using data we collected from over 1,200 users through crowdsourcing, and thoroughly evaluated the effectiveness of our approach for two tidy-up scenarios: sorting toys in boxes and arranging groceries on shelves. Additionally, we demonstrated the

applicability of our approach in a real service robot scenario. Our results show that our technique is accurate and is able to sort objects into different containers according to user preferences.

The approach presented in this chapter focused on computing object arrangements that align with user preferences while adapting to the number of containers in the environment. This enables the robot to predict a desirable goal state to achieve without constantly querying its users for the best way to handle each new situation. To manipulate the objects and achieve the predicted goal state of the task, we leveraged predefined spatial relations between objects and containers and out-of-the-box solutions for task and motion planning. However, this assumes prior semantic knowledge about the involved objects and relations as well as suitable action models for manipulation. This limits the robot to handling objects that are supported by its prior knowledge. In the next chapters, we address this and tackle the problems of learning to manipulate objects by acquiring new action models from teacher demonstrations (Chapter 4), and using them to compute plans that achieve desirable task goals based on the intention of the teacher (Chapter 5). Additionally, in Chapter 6, we tackle the problem of learning arbitrary spatial relations between pairs of objects from non-expert teachers and generalizing them to new objects.

4. Learning Action Models from Teacher Demonstrations

The previous chapter dealt with enabling a robot to predict the preferences of its user by inferring the best way to solve a task such as arranging objects. By extracting patterns in user preferences from a corpus of data related to different users, the robot is able to tailor the task to a specific user and infer the best goal state for her. To compute a plan to achieve this goal in practice, we relied on existing actions and planners. However, for a robot to operate effectively alongside humans, it should also be able to continuously acquire new action models in order to handle new situations such as grasping and moving novel objects to achieve arbitrary desirable configurations. In this chapter, we tackle the problem of learning models of actions from a small number of non-expert teacher demonstrations. Specifically, we tackle learning the relevant spatial constraints between the involved objects before and after applying an action. This is crucial for successfully generalizing the demonstrations and for sequencing several actions when planning. Due to the small number of training examples, a key challenge in this context is the ambiguity in the intention of the teacher with respect to the relevant constraints. We first address this problem from a model-selection perspective by leveraging prior expert knowledge about actions in general to infer a model of the new action. Our experiments demonstrate the effectiveness of this method and its ability to reproduce an action successfully in new starting states. Additionally, we propose a novel approach that overcomes the ambiguity in the demonstrations in a data-driven manner without requiring prior expert knowledge. Rather than committing to a single model, our techniques provide the robot with the flexibility to reason about multiple valid interpretations of the learned action.

In Chapter 3, we proposed a method to enable a robot to predict the preferences of its user with respect to the goal of a task. This allows the robot to infer preferred goal

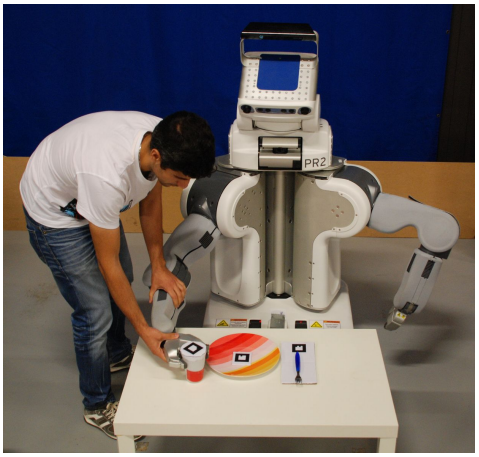


Figure 4.1.: In this chapter, we tackle the problem of teaching a robot action models from user demonstrations. This enables the robot to deal with novel situations (e.g., new objects) and adapt to the continuous needs of its users. In this example, the teacher demonstrates to the robot how to move a cup in order to achieve desirable spatial relations between the cup and the other objects on the table.

states (object arrangements) by relating its knowledge about the user to data collected from different users and environments. To achieve this goal state, Chapter 3 relied on previous action models and planners for solving the task by computing a plan to move objects to their desirable locations. Typically, these action models are provided beforehand by an expert with domain knowledge.

In practice, however, it is infeasible to pre-program the robot with action models for handling all potential situations it can face in a domestic environment. On the one hand, the robot may encounter new objects that it does not know how to handle or grasp yet. On the other hand, new tasks might require the robot to manipulate objects to achieve arbitrary arrangements that are not accounted for in the robot's prior knowledge of actions and their goals. To address these limitations, robots should have the ability to continuously acquire new action models by learning from their users, which is the problem we address in this chapter.

In this context, programming by demonstration is a promising paradigm that enables non-expert users to intuitively transfer their knowledge about actions to robots simply by demonstrating them, see Figure 4.1. This typically involves addressing two main challenges, namely the questions of *what* to imitate and *how* to imitate (Billard et al., 2008). The former question aims at identifying the features, constraints, or symbols that are relevant for reproducing an action, whereas the latter addresses learning the motion associated with the action, i.e., generating feasible trajectories of the robot's manipulators when imitating the action. In this thesis, we consider point-to-point

actions that are modeled in terms of constraints before and after applying an action. Therefore, we focus on addressing the first challenge, i.e., *what* to imitate, and not on learning the motion associated with the action. Given a set of demonstrations of an action, we seek to infer the relevant aspects of the action before and after applying it so that a robot can reproduce it in starting states that differ from the ones demonstrated by the teacher. To execute the action on the trajectory level based on the learned spatial constraints, we rely on existing motion planners or controllers.

In general, learning the relevant spatial constraints for the action from a small number of teacher demonstrations is hard due to the ambiguity in the demonstrations. This is because the feature space defining the geometric constraints between the objects is large. Imitating an action naively based on a few training points in this space will considerably limit the robot's ability to generalize the action to new settings. The most direct solution to this problem is to identify relevant features by making use of large sets of training data. However, from a robotics standpoint, it is highly impractical to generate large amounts of data by non-expert users for every action. Therefore, in this chapter, we consider learning actions from a small number of examples and address the challenge of inferring the best way to model an action despite the ambiguity in the teacher demonstrations. We do so from two perspectives.

First, we propose a method to identify the relevant features of the new action by leveraging expert knowledge about actions in general. We encode this knowledge by borrowing ideas from the theory of recommender systems. As discussed in Chapter 3, these systems typically identify patterns in user preferences either through identifying similarities between users or by analyzing a user's purchase history (Goldberg et al., 1992; Koren, 2008; Basu et al., 1998; Pazzani and Billsus, 1997). Analogously, we consider multiple experts who provide recommendations with respect to which sets of features are relevant for an action. Our approach considers these recommendations based on the state perceived by the robot and the training demonstrations. The experts are users that have *in-depth* knowledge of robot manipulation. Their rules and their recommendations are collected offline, before training, and without knowledge of which specific action will be demonstrated. Based on their recommendations, our system builds multiple probabilistic models of the action. Thus, we formulate a model-selection problem to select the model with the best explanation of the action each time the robot has to reproduce it. We implemented and tested our approach in simulation using real-world data obtained via kinesthetic teaching on a PR2 robot. We demonstrate that we are able to leverage expert knowledge about actions in order to reproduce new actions based on a small number of demonstrations.

The second approach we present enables the robot to reproduce an action in the absence of such prior expert knowledge about relevant features. Rather than learning or inferring an explicit model of the action, we tackle the ambiguity in the demonstrations by constructing an *implicit action model* in a data-driven manner. The idea of our

approach is to let the starting state "vote" for the most likely goal state of an action based on the spatial relations demonstrated by the teacher. In this way, we consider multiple modes in the intention of the teacher by combining several interpretations of the action at the same time. The robot thus selects the most likely goal state of the action based on the state in which it has to reproduce it. Even in completely new situations, our method enables the robot to "improvise" by reproducing as many aspects of the demonstrations as possible. Accordingly, we build on these action models in Chapter 5 to compute desirable task solutions that require sequencing several actions.

The remainder of this chapter is organized as follows. We formalize the problem of learning point-to-point action models in Section 4.1. In Section 4.2, we present our model-selection approach for reproducing a new action based on prior expert knowledge. We demonstrate this approach through experiments with real-world data recorded using a PR2 robot. In Section 4.3, we present our approach for learning implicit action models without prior expert knowledge. Finally, we discuss related work in Section 4.4 before concluding the chapter in Section 4.5.

4.1. Problem Formulation

In this section, we formalize the problem we address in this chapter: learning an action a from teacher demonstrations such that the robot can generalize it to new situations.

4.1.1. State Representation

Let $\mathcal{O}$ be the set of objects involved in the action, which includes the end-effector of the robot, denoted by o_{ee}. We follow Section 2.1.1 and model the state $\mathbf{s}_t$ at time t using the poses of all objects expressed relative to a world frame. We describe the pose of an object $o_k \in \mathcal{O}$ in $SE(3)$ as a homogeneous transformation matrix $\mathbf{T}_k$, composed of a 3D translation vector $\mathbf{t}_k$ and 3D rotation matrix $\mathbf{R}_k$. We use ${}^l\mathbf{T}_k$ to denote the pose of object o_k relative to o_l. Moreover, $\mathbf{T}_k(t)$ and ${}^l\mathbf{T}_k(t)$ respectively denote the pose of object o_k and the pose of o_k relative to o_l at time t. Additionally, our state model considers the joint configuration $\mathbf{o}_{ee}(t)$ of the end-effector at time t, e.g., the degree of opening of the gripper.

4.1.2. Action Demonstrations

To learn a model of an action a, we rely on a teacher who provides the robot with a set of N_a demonstrations $\mathcal{D}^{(a)} = \{\mathbf{d}^{(1)}, \ldots, \mathbf{d}^{(N_a)}\}$ of how to apply the action from different starting states. We consider different methods of demonstrating an action to the robot, see Figure 4.2.

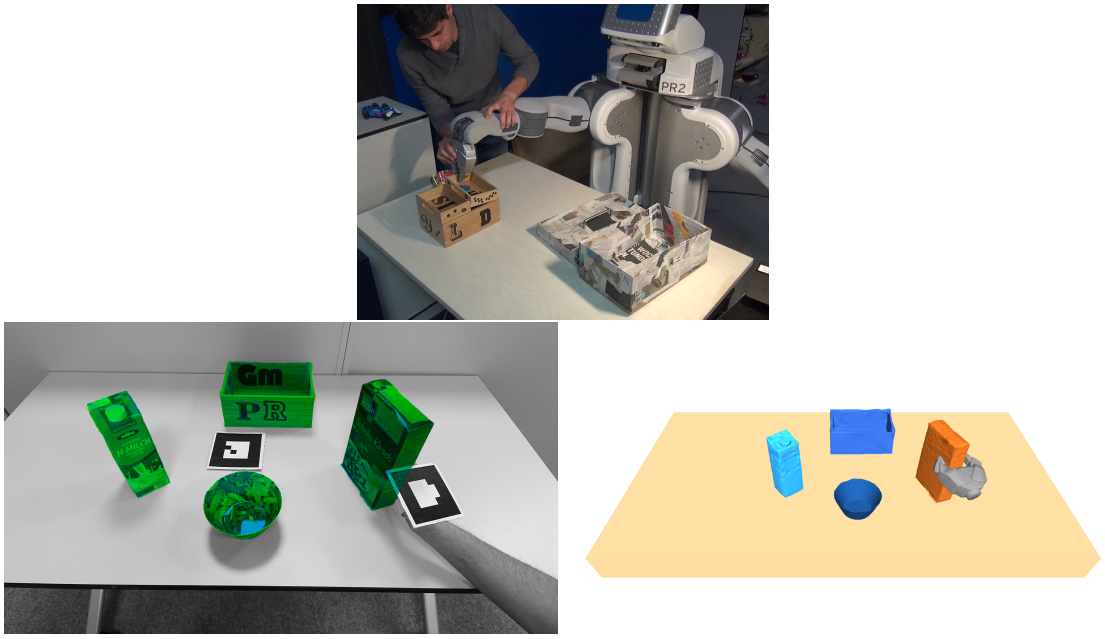


Figure 4.2.: We considered different methods of providing the robot with demonstrations. Top: the teacher provides kinesthetic demonstrations by directly moving the robot's manipulator to achieve the goal of an action. Bottom: we additionally relied on fiducial markers to emulate the end-effector of the robot and to facilitate detecting the table when providing demonstrations. The left scene shows the camera image of the scene with the detected objects using SimTrack (Pauwels and Kragic (2015)). The right side shows a visualization of the corresponding scene with the robot's end-effector.

In this work, we assume complete observability during the demonstrations, i.e., the robot can observe the state (the poses of all objects) using existing perception techniques. Moreover, as we consider point-to-point actions (Section 2.1.3), we assume that the demonstrations are segmented by the teacher such that each demonstration starts from an arbitrary state $\mathbf{s}_{t_s}$ and ends with a desired goal state $\mathbf{s}_{t_e}$ that reflects the intention of the teacher for this action. Accordingly, we express each demonstration $\mathbf{d}^{(n)} \in \mathcal{D}^{(a)}$ as a sequence of two states such that $\mathcal{D}^{(a)} = \{(\mathbf{s}_{t_s}, \mathbf{s}_{t_e})^{(1)}, \ldots, (\mathbf{s}_{t_s}, \mathbf{s}_{t_e})^{(N_a)}\}$, where t_s and t_e are the start and end time of a segment, respectively.

Furthermore, the teacher demonstrates the intention of the action by moving a subset of the objects in each case. Accordingly, we associate an action a with two sets of objects. The first, denoted by $\mathcal{O}_M^a$, consists of the objects that move when applying the action. The second, denoted by $\mathcal{O}_S^a$, is the set of objects that remain stationary when applying the action. These sets are non-empty and partition the set of objects $\mathcal{O}$, i.e., $\mathcal{O}_M^a \neq \emptyset$, $\mathcal{O}_S^a \neq \emptyset$, $\mathcal{O}_M^a \cap \mathcal{O}_S^a = \emptyset$, and $\mathcal{O}_M^a \cup \mathcal{O}_S^a = \mathcal{O}$. Finally, we assume that no object can move unless it is manipulated by the robot. Therefore, the set $\mathcal{O}_M^a$ includes the

end-effector o_{ee} for all actions.

4.1.3. State Affordability

Using the set of demonstrations $\mathcal{D}^{(a)}$, we aim to learn a model of the action. One aspect of that is learning a model $p(\mathbf{s}_{t_s} \mid a)$ of states $\mathbf{s}_{t_s}$ that afford applying the action a. We refer to this as the *state affordability* (see Section 2.1.3). We assume the teacher demonstrations implicitly encode the conditions necessary for the action to succeed based on the starting state $\mathbf{s}_{t_s}$. By ensuring that the robot applies the action in states with a high affordability, we therefore increase the likelihood of the action succeeding and achieving a goal state that aligns with the intention of the teacher.

4.1.4. Action Transition Model

Moreover, we aim to learn a *transition model* $p(\mathbf{s}_{t_e} \mid \mathbf{s}_{t_s}, a)$, capturing the likelihood of a state $\mathbf{s}_{t_e}$ being an intended goal of the action when applied in state $\mathbf{s}_{t_s}$. This is potentially a multi-modal distribution reflecting different valid ways of applying the action as demonstrated by the teacher, e.g., placing a cup on either side of a plate. This is analogous to the transition probabilities of stochastic actions in Markov Decision Processes (Bellman, 1957a,b; Sutton and Barto, 1998). However, in our case, we assume that the only source of stochasticity is the hidden intention of the teacher and not in the dynamics of executing the action. In other words, if the robot infers a likely and feasible goal state $\mathbf{s}_{t_e}$, then it is able to achieve it deterministically using an existing motion planner or controller.

4.1.5. Action Templates

The main challenge in learning the above models lies in the inherent ambiguity in the demonstrations of the teacher. Given only a small number of demonstrations and the large feature space of relations between the objects, the intention of the teacher can be "explained" in different ways. Depending on the degree of variation in the demonstrated starting states, some spatial relations between the objects can seem to be relevant or irrelevant for correctly reproducing the action. This ambiguity renders learning the action challenging, as different spatial constraints can conflict with each other, see Figure 4.3 Without a model specifying the relevant features for the action, the robot will be limited in its ability to generalize the action in new starting states not demonstrated by the teacher.

To address this, we rely on so-called action *templates* for modeling the action. A template γ provides an interpretation of the action by specifying the subset of features

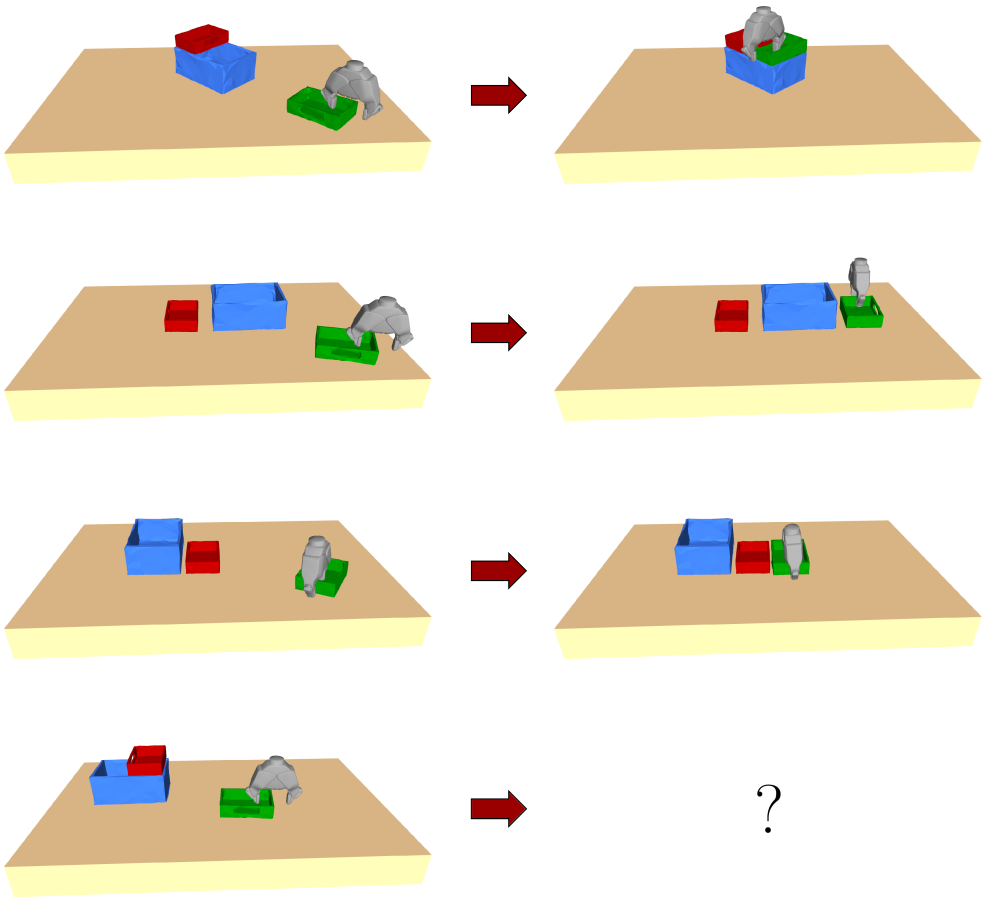


Figure 4.3.: An illustration of the inherent ambiguity in learning actions from a small number of teacher demonstrations. Each of the top three rows shows a demonstration for the action of moving the green box. The left side shows the starting state and the right side shows the corresponding goal state according to the latent intention of the teacher. Given such demonstrations, we tackle the problem of generalizing the action by applying it in a previously unseen starting state (bottom row). In such a case, the robot has to infer a likely goal state to generalize the action as it cannot satisfy all demonstrated spatial constraints involving the green box at the same time, e.g., with respect to the table and the blue box.

(spatial constraints) that are relevant for learning it, thereby reducing the dimensionality of the problem.

4.1.6. The Problem

Using these models, we now formulate the problem we address in this chapter. Given an arbitrary starting state $\mathbf{s}_{t_s}$, the robot should *apply* the action in order to achieve a goal state $\mathbf{s}^*_{t_e}$ that aligns with the intention of the teacher. Concretely, our goal is twofold: *i)* to use the teacher demonstrations to construct a distribution $p(\mathbf{s}_{t_e} \mid \mathbf{s}_{t_s}, a)$ of goal states for the action given the starting state, and *ii)* to use this model to compute a feasible goal state $\mathbf{s}^*_{t_e}$ maximizing the likelihood of the intended action goals, i.e.,

$$\mathbf{s}^*_{t_e} = \arg\max_{\mathbf{s}_{t_e}} \quad p(\mathbf{s}_{t_e} \mid \mathbf{s}_{t_s}, a), \tag{4.1}$$

$$\text{subject to} \quad \mathbf{s}_{t_e} \in \mathcal{S}_\text{f}, \tag{4.2}$$

where $\mathcal{S}_\text{f}$ denotes the set of feasible states. The constraint in Eq. (4.2) ensures that the robot is able to achieve $\mathbf{s}^*_{t_e}$ in practice, for example with respect to the inverse kinematics constraints of its manipulator, see Section 2.2.

In this chapter, we investigate two approaches to solving this problem, each with different assumptions regarding prior knowledge. In the first (Section 4.2), we assume the existence of a set of candidate models $\Gamma = \{\gamma_1, \ldots, \gamma_{|\Gamma|}\}$ designed beforehand by experts in order to interpret different manipulation actions. Accordingly, we formulate a model selection problem to infer the best model $\gamma^* \in \Gamma$ to apply the new action based on the affordability of $\mathbf{s}_{t_s}$ given γ^*. We then use γ^* to compute a goal $\mathbf{s}^*_{t_e}$ that maximizes $p(\mathbf{s}_{t_e} \mid \mathbf{s}_{t_s}, a, \gamma^*)$. In the second (Section 4.3), we propose a method that does not assume such prior knowledge in the form of expert models. Instead we introduce an *implicit action model* corresponding to a flexible, multi-modal distribution of goals based on a mixture of possible interpretations of the teacher demonstrations. Note that in this work, we focus on inferring the best goal state $\mathbf{s}^*_{t_e}$ that aligns with the intention of the teacher, and not on computing a feasible trajectory to achieve this state starting in $\mathbf{s}_{t_s}$. To execute the motion associated with the action, we rely on existing motion planners and controllers.

4.2. Action Models Based on Expert-Designed Templates

In this section, we propose an approach for learning an action by leveraging prior expert knowledge about manipulation actions in general. Specifically, we aim to learn the spatial constraints between the involved objects that are relevant for reproducing the

action in arbitrary initial states. We represent the spatial relations between the objects in terms of features that we use to model the action. It is clearly infeasible to expect an expert to pre-program a robot with knowledge about the relevant features for all potential actions and objects it can encounter in the real world. At the same time, learning a new action from a small number of demonstrations is challenging due to the large dimensionality of the feature space capturing relations between the involved objects. Therefore, we propose to leverage knowledge about previously-known manipulation actions that are similar with respect to patterns in the feature changes associated with applying the action. This allows us to significantly reduce the dimensionality of the problem by considering the smaller feature spaces defined by such previous action models. Furthermore, this results in a flexible approach in which we select the best model for applying the action depending on the starting state rather than committing to the same model for all cases.

4.2.1. Feature Representation

We propose to model point-to-point actions in terms of changes in feature values before and after applying an action. Therefore, we first seek a feature representation to encode the teacher demonstrations. For this, we consider two types of feature functions. The first is a relation function $f_{\mathcal{R}}(p_{(k,l)}, \mathbf{s}) \rightarrow \mathbb{R}$ that describes a certain relation between a pair of objects $p_{(k,l)} = \{o_k, o_l\}$ in state $\mathbf{s}$ such as the distance between them or their relative position with respect to the x-axis. The second $g_o(o_k, \mathbf{s}) \rightarrow \mathbb{R}$ is an object-specific function that describes a property of an object o_k such as its size or the degree of opening in the case of the end-effector object o_{ee}. Accordingly, we represent a state $\mathbf{s}_t$ at time t as a feature vector $\mathbf{f}_t \in \mathbb{R}^F$ such that

$$\mathbf{f}_t = [f_1, \ldots, f_{N_1}, g_1, \ldots, g_{N_2}]^\top, \tag{4.3}$$

where $F = N_1 + N_2$ and $f_1, \ldots, f_{N_1}$ and $g_1, \ldots, g_{N_2}$ are the feature values computed using functions $f^1_{\mathcal{R}}, \ldots, f^{N_1}_{\mathcal{R}}$, and $g^1_o, \ldots, g^{N_2}_o$, respectively. In general, $\mathbf{f}_t$ has a large dimensionality based on all possible relation and object functions for the set of objects $\mathcal{O}(\mathbf{s}_t)$ in state $\mathbf{s}_t$.

Based on this representation, we can express a teacher demonstration $\mathbf{d} = (\mathbf{s}_{t_s}, \mathbf{s}_{t_e})$ in terms of the corresponding feature vectors at t_s and t_e, i.e., $\mathbf{d} := (\mathbf{f}_{t_s}, \mathbf{f}_{t_e})$. This is an abstraction of the trajectory $(\mathbf{f}_{t_s}, \ldots, \mathbf{f}_{t_e})$ consisting of the sequence of features computed at each time step of the demonstration (assuming a discrete-time system).

4.2.2. Modeling an Action

Learning an action a corresponds to building a model for it based on the teacher demonstrations $\mathcal{D}^{(a)}$. Specifically, in the context of point-to-point actions, we are

interested in learning a probability density function that models the interaction between features at the start and the end of the action. In this work, we assume that all dimensions of a feature vector $\mathbf{f}$ are independent of each other, and therefore aim to model these interactions for each feature individually.

Let s_i and e_i be the i-th dimension of $\mathbf{f}_{t_s}$ and $\mathbf{f}_{t_e}$, respectively. We construct a bivariate density function $\phi_i(s_i, e_i)$ capturing the joint likelihood of s_i and e_i for this feature dimension. We adopt a discretized model that approximates the continuous feature space by dividing it into bins. Accordingly, we define ϕ_i in terms of a 2-dimensional histogram $\mathcal{H}_i$ as follows:

$$\phi_i(s_i, e_i) := \eta_i \mathcal{H}_i(\lfloor s_i \rfloor, \lfloor e_i \rfloor), \tag{4.4}$$

where η_i is a normalizer and $\lfloor \cdot \rfloor$ is a quantization operator that returns the histogram bin for the given feature value. We build ϕ_i by populating the corresponding histogram with the training data (feature values) across all demonstrations.

To reproduce the action from a new starting state $\mathbf{s}^*_{t_s}$, our idea is to use the corresponding feature values $\mathbf{f}^*_{t_s}$ to reason about goal values $\mathbf{f}^*_{t_e}$ defined by the distribution associated with each feature dimension. In the general case, the robot has not seen the teacher execute the action in $\mathbf{s}^*_{t_s}$ before. However, our discretized model allows us to use all demonstrations with starting features $\mathbf{f}_{t_s}$ that fall in the same s_i bins as $\mathbf{f}^*_{t_s}$ in each feature histogram $\mathcal{H}_i$. The possible goals $\mathbf{f}^*_{t_e}$ are therefore modeled by the conditional distribution $\phi_i(e_i \mid s_i = s^*_i)$, which we construct by taking the (normalized) 1-dimensional histogram over e_i for bin $\lfloor s^*_i \rfloor$.

The shape of $\phi_i(e_i \mid s_i = s^*_i)$ is important as it models the distribution of possible goals observed in $\mathcal{D}^{(a)}$ with respect to the i-th feature dimension given the starting state. We assume that aspects considered by the teacher to be relevant for the action are associated with consistent demonstrations, i.e., low variance in the corresponding feature values. Accordingly, if the goals are concentrated in a few possible values, we interpret this as an indication of goodness. In other words, based on the training data, the goal state with respect to that feature dimension is known with high certainty. We illustrate this idea with an example in Figure 4.4.

We compute this confidence using the entropy H_i of e_i based on $\phi_i(e_i \mid s_i = s^*_i)$

$$H_i(e_i) = -\sum_{e_i} P(e_i) \log\left(P(e_i)\right), \tag{4.5}$$

where $P(e_i) = \phi_i(e_i \mid s_i = s^*_i)$, i.e., the probability of e_i being a goal with respect to the i-th feature when starting in s^*_i.

Let $\mathbf{\Phi}^a$ be the set of all bivariate distributions ϕ_i modeling the action. Based on the entropy H_i associated with each feature dimension, we aim to compute a score $q(\mathbf{f}^*_{t_s} \mid \mathbf{\Phi}^a)$ that captures the likelihood of starting state $\mathbf{s}^*_{t_s}$ given the model, i.e., the likelihood of $\mathbf{s}^*_{t_s}$ leading to goal states that align with the teacher demonstrations. This

Figure 4.4.: An example of two feature distributions constructed based on ten demonstrations for the action of placing a grasped object A on top of another object B. The left side shows feature f_1, which describes the pose of A relative to B in the x direction. The right side depicts feature f_2 describing the pose of the end-effector relative to the robot's torso frame in the x direction. Given a new starting point s_i^*, the conditional distribution of goal values for f_1 exhibits less entropy compared to that of f_2. This results in a higher likelihood (Eq. (4.6)) with respect to f_1 compared to f_2, indicating a higher relevance to the action compared to f_2.

is similar to the affordability of $\mathbf{s}_{t_s}^*$, see Section 4.1.3. We define this affordability score as follows

$$q(\mathbf{f}_{t_s}^* \mid \mathbf{\Phi}^a) := \prod_{i=1}^{F} e^{-H_i}. \tag{4.6}$$

Lower entropy values for each feature dimension given the starting state correspond to a higher confidence about the goal feature values. Note that if the new starting state for some feature dimension corresponds to a bin where no training data points are available, we use the marginalized goal distribution over all starting feature values observed in the demonstrations for computing the entropy.

As mentioned before, the main issue with this model is the dimensionality of the feature space. A large number of features that might be irrelevant for the action will render the model unlikely. Moreover, different feature dimensions with a high confidence about the goal might impose contradictory constraints with respect to the goal states. This is especially true when data is scarce, resulting in ambiguity regarding the intetion of the teacher. Therefore, we address this issue using a feature selection method to reduce the dimensionality of the feature space based on pre-existing action models.

4.2.3. Feature Selection Using a Template Recommender System

From a robotics standpoint, it is impractical to generate large sets of training examples for each action. Instead of collecting a large amount of data, we seek to reduce the

dimensionality of the problem. The key idea of our approach is to perform feature selection by means of a "recommender system" based on expert knowledge. This system proposes a portfolio of various low-dimensional feature spaces that are able to explain *some* actions. The spaces are based on a set of feature functions (see Section 4.2.1 above) that are provided by domain experts with in-depth knowledge of robotic manipulation. We collect this information offline, before training, and without knowledge of the demonstrated action a beforehand.

We represent this expert knowledge in the form of *action templates* introduced in Section 4.1.5. Note that here, we consider templates that are not tailored to a specific action. Instead, each template specifies, on an abstract level, a set of feature functions that are typically relevant for modeling a variety of actions. An example of an expert template γ_1 is one that considers only the spatial relations between the end-effector of the robot, a grasped object, and a second object in the scene to be relevant. Features related to any other object are considered irrelevant by γ_1. The robot can leverage such a template to learn different actions, e.g., one where the teacher demonstrates how to place a grasped cup next to a plate, and one where the teacher demonstrates placing a grasped toy inside a box. In other words, we consider expert templates that are generic or abstract with respect to the specific object instances used by the teacher, and which can therefore be grounded in different ways based on the demonstrations. We use Γ to denote the set of all templates provided by potentially multiple experts.

Moreover, each template $\gamma \in \Gamma$ is associated with a *template relevance function* $p(\gamma \mid \mathbf{s}^*_{t_s}, a)$ designed by the expert. This function specifies whether γ should be considered for modeling the action given the teacher demonstrations $\mathcal{D}^{(a)}$ and the starting state $\mathbf{s}^*_{t_s}$. In this work, we consider this function to be boolean, i.e., taking a value of either 0 or 1. For example, in the case of γ_1 above, $p(\gamma_1 \mid \mathbf{s}^*_{t_s}, a)$ is 1 if the teacher demonstrations involved a grasped object moving with the end-effector, as well as consistent relative poses between the grasped object and a stationary object o_s in $\mathcal{O}^a_S$. On the other hand, $p(\gamma_1 \mid \mathbf{s}^*_{t_s}, a)$ is 0 if the object o_s is not in the scene when applying the action, i.e., $o_s \notin \mathcal{O}^a_S(\mathbf{s}^*_{t_s})$. The expert designs such template relevance functions beforehand based on domain knowledge and independent of the new actions or demonstrations that the robot could encounter.

Accordingly, we compute a subset of recommended templates Γ^* using all templates for which $p(\gamma \mid \mathbf{s}^*_{t_s}, a) = 1$. As the same template can be grounded in different ways (e.g., for several objects in $\mathcal{O}^a_S(\mathbf{s}^*_{t_s})$), we consider all possible interpretations individually when constructing Γ^*. This can be interpreted as a form of content-based recommender systems (Pazzani and Billsus, 1997; Basu et al., 1998). Such systems recommend different items or products to a user by analyzing her profile or purchase history. In our context, we aim to predict which templates to consider for modeling the action by analyzing the teacher demonstrations and the new starting state in which the robot has to reproduce the action.

Note that we recommend the set of templates Γ^* *each time* the robot has to reproduce the action, as this depends on which objects exist in $\mathbf{s}^*_{t_s}$. Rather than committing to a specific action model all the time, we are therefore able to generalize to changes in conditions between the teacher demonstrations and the new starting state. Finally, each template $\gamma_g \in \Gamma^*$ addresses the dimensionality problem in Section 4.2.2 by defining a low dimensional feature space of size $F_g \ll F$ for modeling the action and evaluating the state affordability likelihood in Eq. (4.6). In the next section, we discuss how we use this to select the best template for applying the action in $\mathbf{s}^*_{t_s}$.

4.2.4. Model Selection for Reproducing the Action

Given the recommended set of templates Γ^*, the robot has to select a template $\gamma_g \in \Gamma^*$ corresponding to the best subset of features for applying the action in $\mathbf{s}^*_{t_s}$.

4.2.4.1. Selecting the Best Template

We address the problem of selecting a template from a model-selection perspective and compute a score β_g for each template $\gamma_g \in \Gamma^*$ based on two criteria. The first favors templates with a high confidence about the intended goal of the action given $\mathbf{s}^*_{t_s}$, i.e., a high state affordability, see Eq. (4.6). Such templates are able to *explain* the demonstrations by relating them to the new starting state. The second encourages using templates that consider more features when reproducing the action, as these imitate more aspects of the action with respect to the objects and the relations between them. This regularizes the first criterion and avoids selecting "trivial", over-confident templates, e.g., one that considers only the distance between two objects.

We compute the affordability score $q(\mathbf{f}^*_{t_s} \mid \mathbf{\Phi}^a, \gamma_g)$ given template γ_g by only considering the populated bivariate distributions ϕ of features specified by the template. In other words, we compute $q(\mathbf{f}^*_{t_s} \mid \hat{\mathbf{\Phi}}^a_g)$, where $\hat{\mathbf{\Phi}}^a_g$ are the distributions of the template features, i.e., $|\mathbf{\Phi}^a_g| = F_g$. Accordingly, we compute the score β_g as follows:

$$\beta_g = -\ln\left(q(\mathbf{f}^*_{t_s} \mid \hat{\mathbf{\Phi}}^a_g)\right) - \alpha F_g, \tag{4.7}$$

where α is a weight controlling the trade-off between the state affordability of the template and its complexity F_g. Thus, we select the best template $\gamma^* \in \Gamma^*$ as

$$\gamma^* = \arg\min_{\gamma_g} \beta_g. \tag{4.8}$$

We perform this selection each time the robot has to apply the action in a new state as the recommended templates and their scores depend on the starting state.

4.2.4.2. Applying the Action

After selecting γ^*, we solve the problem in Section 4.1.6 by computing a goal state $\mathbf{s}^*_{t_e}$ that maximizes the likelihood of the intended action goals $p(\mathbf{s}_{t_e} \mid \mathbf{s}^*_{t_s}, a, \gamma^*)$. Here, we model action goals in terms of constraints describing desirable goal feature values $\mathbf{f}_{t_e}$ demonstrated by the teacher. We assume that each teacher demonstration $\mathbf{d}^{(n)} = (\mathbf{s}_{t_s}, \mathbf{s}_{t_e})^{(n)}$ encodes the preferred way of applying the action given the corresponding starting state $\mathbf{s}_{t_s}$. For example, the teacher could place a grasped cup on a specific side of a plate depending on the positions of other objects relative to the plate. Similarly, the teacher would demonstrate to the robot how to move its end-effector to achieve the best grasp for a specific object depending on how it is placed on the table beforehand.

Therefore, given the new starting state $\mathbf{s}^*_{t_s}$, we seek to compute $\mathbf{s}^*_{t_e}$ such that the goal feature values $\mathbf{f}^*_{t_e}$ correspond to those of the demonstration $\mathbf{d}^* \in \mathcal{D}^{(a)}$ with the closest resemblance with respect to the starting conditions, i.e.,

$$\mathbf{d}^* = \underset{\mathbf{d}^{(n)} \in \mathcal{D}^{(a)}}{\arg\min} \|\mathbf{f}^{(n)}_{t_s} - \mathbf{f}^*_{t_s}\|_{\gamma^*}, \tag{4.9}$$

where $\mathbf{f}^{(n)}_{t_s}$ is the starting feature vector of the n-th demonstrations, and $\|\cdot\|_{\gamma^*}$ is the distance computed using only the feature dimensions defined by γ^*. We compute this as a weighted sum of the distances with respect to each feature scaled by the entropy of starting values in that dimension over all demonstrations. In this way, we give more importance to dimensions with consistent feature values.

Finally, given the selected demonstration $\mathbf{d}^*$, we compute the goal state $\mathbf{s}^*_{t_e}$ by sampling goal poses of the end-effector that satisfy feature values $\mathbf{f}^*_{t_e}$ observed in $\mathbf{d}^*$. For this, we only consider features in the selected template γ^* that describe spatial relations between the end-effector and stationary objects in the scene. We construct the goal state $\mathbf{s}^*_{t_e}$ by computing a goal end-effector pose satisfying all such constraints, and assuming that all other moving objects will maintain their current poses relative to the end-effector after applying the action, i.e., no grasped object slips during the motion. Note that in this work, we focus on computing the goal of an action based on a small set of ambiguous demonstrations. Our approach enables the robot to select the best set of spatial constraints for modeling the action and to compute a goal state accordingly. This eliminates the need for an expert to provide this information for each new action. To achieve the computed goal state, we assume the robot is equipped with a suitable technique for computing a feasible trajectory. This can either be done by adapting the trajectory demonstrated by the teacher in $\mathbf{d}^*$ (transformed based on the new goal pose of the end-effector), or by relying on a motion planner to compute a new trajectory.

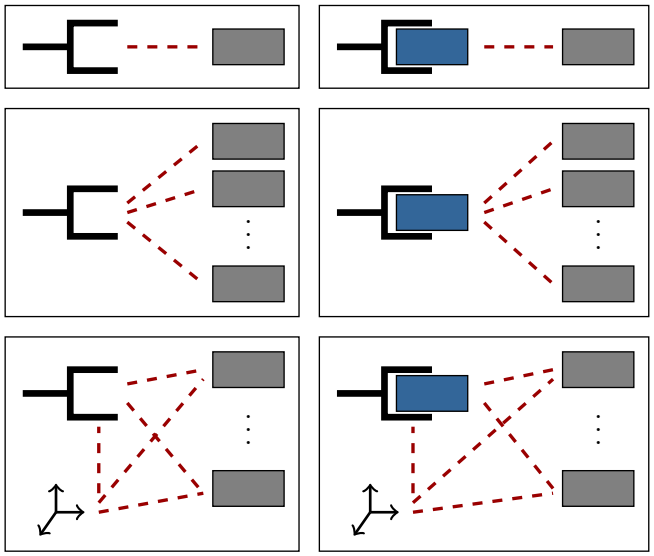


Figure 4.5.: Six examples of abstract expert templates we considered in this work. Blue objects indicate objects that move with the end-effector, whereas gray objects indicate stationary objects. The coordinate systems in the lower row indicate the world frame. We use the dashed red edges to indicate that the template considers spatial relations between the corresponding objects as relevant for the action.

4.2.5. Approach Evaluation

In this section, we evaluate our approach for learning an action based on expert-designed templates. We consider three actions that we demonstrated on a real PR2 robot kinesthetically. For perceiving the state during the demonstrations, we relied on fiducial markers that we attached to the objects and measured their poses using a camera mounted on the robot's head, see Figure 4.1.

4.2.5.1. Expert Templates

To evaluate our method, we designed a set Γ of generic action templates along with their relevance functions $p(\gamma \mid \mathbf{s}^*_{t_s}, a)$ that determine when they are recommended for applying an action a based on the demonstrations $\mathcal{D}^{(a)}$ and a new starting state $\mathbf{s}^*_{t_s}$, as described in Section 4.2.3.

Each template considers only a subset of the potential spatial relations between the objects that seem relevant during the demonstrations. We based our templates on typical pick-and-place actions that achieve desirable spatial arrangements of objects. For instance, one template considers the relation between the end-effector and one stationary object $o_s \in \mathcal{O}^a_S$ if the teacher demonstrations involved a "consistent" goal pose of the

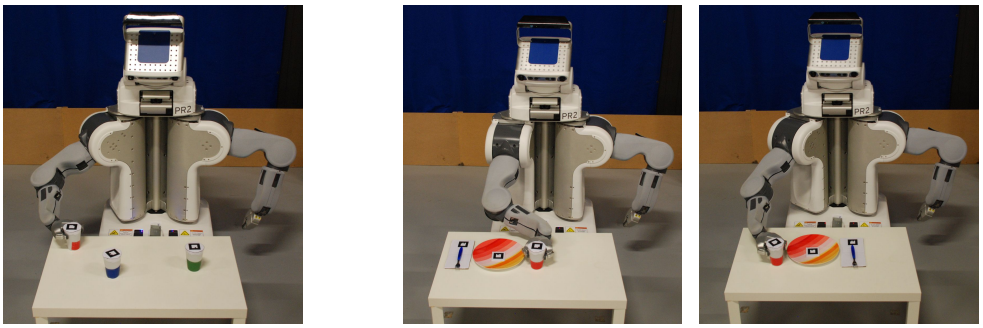


Figure 4.6.: Left: we demonstrated to the robot how to reach for a specific object (red cup). The positions of the other objects on the table do not change in the demonstrations, adding to the ambiguity of the intended action goals. Right: we demonstrated to the robot how to place a grasped cup next to a plate on the table. The cup is always placed on the side opposite to that of the fork.

end-effector relative to o_s. We defined consistency based on the compactness of the corresponding feature values according to the approach of Abdo et al. (2013). This method considers a feature dimension to be relevant for the action if its demonstrated values can be modeled using a small number of compact clusters. Similarly, other templates considered features describing the relation between the end-effector and all stationary objects.

Additionally, we designed templates that consider spatial relations between stationary objects or the end-effector and the global reference frame (e.g., the table or the torso of the robot). Our system recommends such templates if the involved objects had consistent poses relative to the global frame across the demonstrations.

Moreover, we designed templates that additionally consider spatial relations between other moving (grasped) objects $o_m \in \mathcal{O}_M^a$ and the end-effector or stationary objects, and which we recommend if there are consistent feature values between o_m and those objects across the demonstrations. Figure 4.5 illustrates some of the templates we used in our experiments. When recommending a set of templates Γ^* to consider for applying an action, we include all valid ways of grounding those abstract templates based on the objects in the starting state. Finally, for computing the distributions ϕ_i, we used histograms with a bin size of 0.2 m for translation and distance features, and 45 deg for angular features.

4.2.5.2. Demonstrated Actions

We demonstrated three different actions to the robot as follows:

- Action $a^{(1)}$: placing an object on another. We provided the robot with ten

demonstrations of how to place a grasped cup on top of a coaster starting from arbitrary initial poses of the coaster on the table and of the end-effector with the grasped cup.

- Action $a^{(2)}$: reaching for a specific object. We provided the robot with ten demonstrations of how to reach for a red cup placed on the table in order to grasp it. Each demonstration started with the red cup placed at an arbitrary position on the table. All demonstrations involved two other cups (green and blue) in fixed positions on the table, see the two left-most images in Figure 4.6.

- Action $a^{(3)}$: setting a table. We provided the robot with twelve demonstrations of how to place a grasped cup next to a plate and a fork on the table. The teacher always placed the cup on the opposite side of the plate with respect to the fork, providing six training demonstrations for each setting, see the two right-most images in Figure 4.6.

4.2.5.3. Experiments and Results

We tested our approach on each of the three actions in simulation. In each case, we used our approach to apply the action by computing a goal state from a random starting state that was not demonstrated by the teacher. For $a^{(1)}$ and $a^{(3)}$, we sampled a random grasping pose of the grasped cup relative to the end-effector in the starting state. We repeated this 500 times for each action. We used our method to reason about the best template to apply the action in each case and compute a goal state accordingly. We considered a run successful only if we were able to compute a feasible goal state satisfying all features considered by the best template, and if this goal state aligns with the true intention of the teacher, which we did not provide to the robot.

Placing a Cup on a Coaster For $a^{(1)}$, we considered a goal state to be correct if it corresponded to the grasped cup being placed on the coaster. Figure 4.7-right depicts the success rate given different numbers of demonstrations for training. As expected, the performance improves with the number of demonstrations. With more training data, we are able to better disambiguate the demonstrations since feature dimensions that are more relevant for the action will contribute to higher template scores. Figure 4.7-left depicts the ratio of times a specific template was selected as the best template for applying the action given five and ten demonstrations. Overall, our method selected two templates, γ_1 and γ_2, for explaining the action. We achieved a success rate of 75.8% when given five demonstrations for learning. With ten demonstrations, our method solved 93.6% of the cases using γ_1, which includes features describing the poses of the end-effector and the cup relative to the coaster. On the other hand, γ_2 contains features

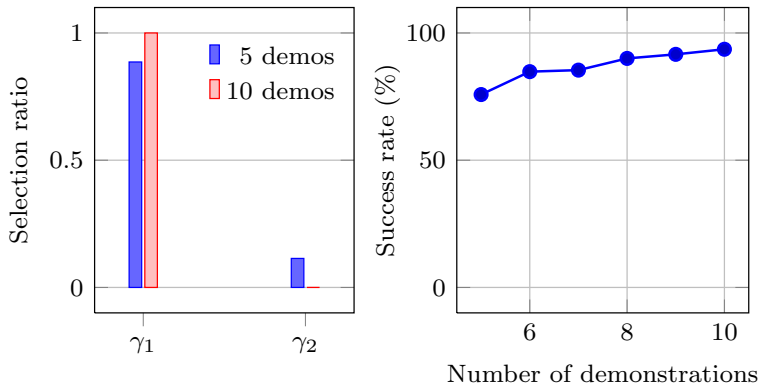


Figure 4.7.: Results for action $a^{(1)}$: placing a grasped object onto another object on the table. Left: the selection of templates given five and ten demonstrations for learning the action. Out of all available templates, the recommender system suggested only two. Our system selected γ_1 most of the time, as this template considers features describing the pose of the gripper and the grasped object relative to the target object on the table. Right: the success rate increases up to 93.6% with ten demonstrations.

describing the pose of the coaster relative to the global reference frame, which are less relevant for the action.

Reaching for a Specific Object In each test case for this action, we started with all three cups placed in random positions on the table. We considered a run successful if the goal state corresponded to correctly grasping the red cup. Figure 4.8-right depicts the success rate as we varied the number of demonstrations from five to ten. When using the full training dataset, we achieve a success rate of 88.8%. As expected, the success rate increases with the size of the training set. Additionally, we analyzed which templates were more influential for applying the action in given five and ten demonstrations, see Figure 4.8-left. Our approach frequently selected three templates, γ_3 - γ_5, when reproducing the action from varying starting poses. γ_3 contains features describing the pose of the end-effector relative to the red cup. On the other hand, γ_4 and γ_5 contain features involving the poses of the other two cups and the end-effector relative to the robot reference frame. Combined, they provided successful interpretations of the action in 74% of the 500 trials given five demonstrations. With ten demonstrations, γ_3 successfully explained 88.8% of all the initial configurations. In other words, changes in the poses of the green and blue cups in the starting state rendered the templates considering irrelevant features to become unlikely with more data, enabling the robot to generalize the demonstrations of the teacher.

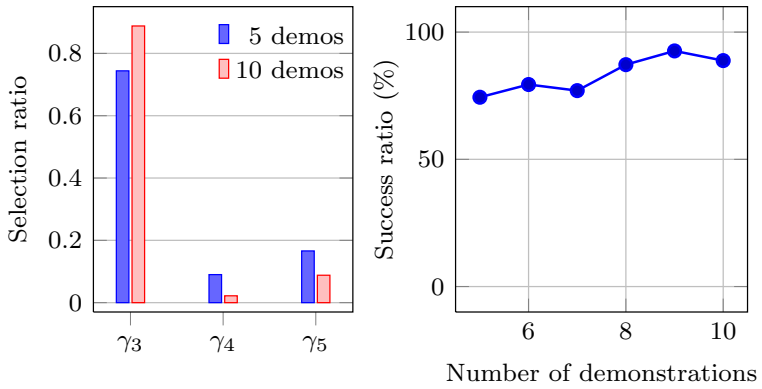


Figure 4.8.: Results for action $a^{(2)}$: reaching for a specific object. Left: with more demonstrations, the system prefers γ_3 to the other models. This template contains features describing the pose of the gripper relative to the target object. Right: the success rate increases with the number of training demonstrations, reaching up to 92.6%.

Setting a Table In all starting configurations for this action, we placed the fork to the left of the plate. We recorded the success rate as the number of times the robot placed the cup on the right side of the plate (i.e., opposite to the fork). The results are shown in Figure 4.9-right. We achieved a success rate of 96% when using all twelve training examples. Additionally, we analyzed the influence of the number of training data on the success rate. As expected, the success rate increased with the quantity of the learning examples. We also analyzed the template selection over the varying number of training demonstrations, see Figure 4.9-left. With five training demonstrations, four templates were selected approximately equally during the 500 trials. Even with such a small number of demonstrations, our method succeeded in solving 65% by leveraging the different templates based on the starting state. The most selected templates included features describing the pose of the grasped cup relative to the fork and plate. With twelve training demonstrations, there was an increasing preference for explaining the action using template γ_7 which considers features describing the pose of the end-effector and the grasped cup relative to the fork. Using the recommended templates, our system achieved 96% success rate given twelve demonstrations.

To further illustrate the flexibility of our method in reproducing the action under different starting conditions, we ran the experiment again after removing the fork from the scene. In this case, we consider a run successful if our method placed the cup on either side of the plate. Since our method recommends templates based on the objects in the scene, we are able to cope with such changes relative to the teacher demonstrations. In this case, we often selected a template that considers the pose of the cup relative to the plate, and achieved a success rate of 82% given twelve demonstrations.

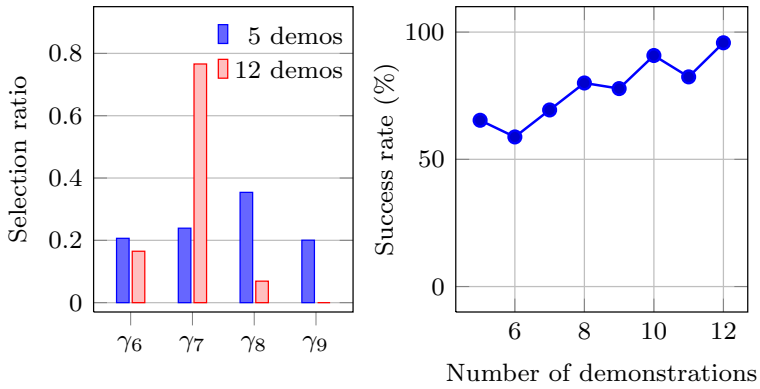


Figure 4.9.: Results for action $a^{(3)}$: placing a grasped cup next to a plate and fork. Left: template selection given five and twelve demonstrations. The system prefers templates that describe the pose of the cup relative to the other objects on the table. Right: the success rate as a function of the number of training demonstrations, reaching up to 96% given twelve demonstrations.

4.2.5.4. Discussion

So far in this chapter, we presented an approach for learning manipulation actions from a small number of demonstrations by leveraging expert knowledge. The inherent ambiguity in the demonstrations can lead to conflicting effects and less generalization to new conditions if one commits to a specific action model. Instead, our method uses techniques inspired from recommender system theory to reason about multiple interpretations of the action and select which features are relevant for reproducing it given a new state. We generate these features based on action templates designed beforehand by experts to model a variety of actions. We use these interpretations to build multiple probabilistic models that we evaluate for each new initial condition in which the robot has to apply the action. In this way, we are able to account for several interpretations of the demonstrations and select the best one depending on the scene.

4.3. Implicit Action Models

The approach we presented in Section 4.2 addressed inferring a suitable action model to use in a new situation based on a set of existing templates designed by experts. This enables us to tackle the ambiguity in the teacher demonstrations by considering only a subset of the potential features (spatial relations) as relevant for applying the action based on the expert recommendations. However, this assumes that the expert models are suited for modeling any arbitrary action that the teacher could demonstrate to the robot. Moreover, after selecting such a model, we assume that the robot is able

to compute a goal state that satisfies all constraints specified by the corresponding template in order to apply the action.

To address this, in this section, we present an approach for modeling an action that does not assume the existence of prior expert knowledge about actions. At the same time, we do not aim to commit to an explicit model of the action goals, as this limits the robot's ability to apply the action in states that differ from the demonstrations. Instead, we present an *implicit* action model that takes into consideration a mixture of interpretations of the intention of the teacher simultaneously.

The main idea of our approach is to model an action a in terms of spatial relations between the set of moving objects $\mathcal{O}_M^a$ and the stationary objects $\mathcal{O}_S^a$ in the scene. Accordingly, each stationary object o_s implicitly represents one potential interpretation γ that "votes" for desirable poses of the moving objects. By combining these interpretations into one flexible model, we construct a multi-modal distribution of action goals that adapts to the starting state.

This is analogous to the concept of implicit shape models in computer vision, a technique for learning flexible models of object shapes in order to recognize them in 2D images (Leibe and Schiele, 2003; Leibe et al., 2004). The idea there is to construct a codebook of features from training image patches based on co-occurrences between those features and the center of the target object being learned. This results in learning an implicit model of what the target object looks like. Recognizing a new instance of the object is done by matching parts of an image to learned codebook entries and having them vote for the position of the object. The method is thus able to reason about the likelihood of the object being in the scene based only on partial observations of features commonly associated with it. In the same manner, our method represents action goals in a flexible manner based on the poses of the stationary objects in the scene.

In Section 4.3.3, we present a continuous model for this distribution based on kernel density estimation. Subsequently, in Section 4.3.4, we present a method for discretizing this distribution to approximate it using a finite set of goal states, each associated with a probability of being an intended goal of the action. This enables us to efficiently sample a likely goal state for the action while still considering multiple modes in the intention of the teacher.

4.3.1. Pairwise Relations

In Section 4.2, we presented action models that use feature functions $f_{\mathcal{R}}$ defined by expert templates. These features describe different spatial relations between the objects such as their distances and relative positions. Similarly, to construct our implicit action model, we rely on a feature representation describing the relation between a pair of objects $p_{(k,l)} = \{o_k, o_l\}$ in terms of the 6D pose ${}^l\mathbf{T}_k$ of o_k relative to o_l.

We use this representation to model distributions $p({}^l\mathbf{T}_k)$ of relative poses for an object

pair given the set of demonstrated poses $\{{}^l\mathbf{T}_k^{(1)}, \ldots, {}^l\mathbf{T}_k^{(N_a)}\}$, where ${}^l\mathbf{T}_k^{(n)}$ is the relative pose in the n-th demonstration (either at time $t = t_s$ or $t = t_e$). This distribution captures the likelihood of a certain state satisfying an intended pairwise constraint with respect to the two objects.

On the one hand, we seek a model that is based on a small number of demonstrations. On the other hand, we aim to capture fine details in the intention of the teacher. Therefore, we adopt a non-parametric, data-driven model and use kernel density estimation to compute the likelihood of a relative pose ${}^l\mathbf{T}_k$:

$$p({}^l\mathbf{T}_k) = \frac{1}{N_a} \sum_{n=1}^{N_a} \mathbf{k}\Big({}^l\mathbf{T}_k^{(n)}, {}^l\mathbf{T}_k\Big). \tag{4.10}$$

Here, the function $\mathbf{k}(\cdot, \cdot)$ is the Gaussian kernel

$$\mathbf{k}\Big(\mathbf{T}^{(n)}, \mathbf{T}\Big) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{\mathrm{dist}(\mathbf{T}^{(n)}, \mathbf{T})^2}{2\sigma^2}\right), \tag{4.11}$$

where σ is the kernel bandwidth. In this work, we compute the distance dist between two poses as a weighted sum of the distances between the translational and rotational components of the corresponding transforms, i.e.,

$$\mathrm{dist}(\mathbf{T}^{(n)}, \mathbf{T}) = \alpha_t \|\mathbf{t}^{(n)} - \mathbf{t}\|_2 + \alpha_r d_g(\mathbf{R}^{(n)}, \mathbf{R}), \tag{4.12}$$

where d_g is the (scaled) geodesic distance between the two rotations, i.e.,

$$d_g(\mathbf{R}^{(n)}, \mathbf{R}) = \frac{1}{\pi\sqrt{2}} \left\| \log\left((\mathbf{R}^{(n)})^\top \mathbf{R}\right) \right\|_F. \tag{4.13}$$

Here, log is the principal logarithm of the relative rotation matrix and $\|\cdot\|_F$ is the Frobenius norm. The scaling by $\frac{1}{\pi}$ serves to restrict the rotational distances to be between 0 and 1. The constants α_t and α_r correspond to weighting coefficients for the translational and rotational distances, respectively. In this work, we set both to 0.5.

In the following sections, we build on such distributions to model the state affordability and goal distributions of the action.

4.3.2. Modeling State Affordability

Recall that the state affordability (Section 4.1.3) encodes the likelihood that a starting state $\mathbf{s}_{t_s}$ is suitable for applying the action as intended by the teacher. In Section 4.2, we modeled the affordability of state $\mathbf{s}_{t_s}$ based on the feature distributions specified by an expert template γ. In this section, we aim for a model that does not rely on such expert knowledge.

Since we associate the action with the set of objects $\mathcal{O}_M^a$ that move when applying it, we propose to model the state affordability $p(\mathbf{s}_{t_s} \mid a)$ based on the pairwise relations

over objects in $\mathcal{O}_M^a$, which we assume to be independent of each other. To achieve this, we select a representative object $o_* \in \mathcal{O}_M^a$, and construct a distribution over the pose of o_* relative to other objects in $\mathcal{O}_M^a$, i.e.,

$$p(\mathbf{s}_{t_s} \mid a) = \prod_{\substack{o_k \in \mathcal{O}_M^a \\ o_k \neq o_*}} \sum_{n=1}^{N_a} \frac{1}{N_a} \mathbf{k}\left({}^k\mathbf{T}_*^{(n)}(t_s),\, {}^k\mathbf{T}_*(t_s)\right), \tag{4.14}$$

where ${}^k\mathbf{T}_*^{(n)}(t_s)$ is the pose of o_* relative to o_k at the start of the n-th action demonstration. In other words, each pair of objects in $\mathcal{O}_M^a$ is associated with a multi-modal distribution, with each training example representing a desirable starting pose from which the action can be applied. In this work, we select o_* to be the end-effector object o_{ee} for all actions.

Note that this distribution models pairwise relations, and is therefore only defined for actions for which $|\mathcal{O}_M^a| \geq 2$. This excludes the case in which $\mathcal{O}_M^a$ includes only the end-effector of the robot, e.g., moving the end-effector to achieve a desirable object grasp. In this case, we consider all states to be equally likely for applying the action, corresponding to a uniform distribution over $\mathbf{s}_{t_s}$

By applying the action starting in states $\mathbf{s}_{t_s}$ with a high affordability, we increase the likelihood of the action succeeding based on the knowledge that the teacher implicitly included in the demonstrations, e.g., achieving a grasping pose of the end-effector relative to a cup before moving the cup. In this chapter, we do not address inferring starting states with a high affordability and focus on inferring a goal state $\mathbf{s}_{t_e}$ of the action assuming the action is applicable in the given starting state $\mathbf{s}_{t_s}$. In Chapter 5, we present an approach that enables the robot to compute a sequence of actions such that each action is applied in a state satisfying a minimal desirable affordability constraint.

4.3.3. Implicit Action Goals

We aim to model a distribution $p(\mathbf{s}_{t_e} \mid \mathbf{s}_{t_s}, a)$ of goal states $\mathbf{s}_{t_e}$ when applying action a in state $\mathbf{s}_{t_s}$. The approach we presented in Section 4.2 relied on selecting an interpretation γ for the action based on an expert template. Accordingly, we modeled the goal states $p(\mathbf{s}_{t_e} \mid \mathbf{s}_{t_s}, a, \gamma)$ based on goal features $\mathbf{f}_{t_e}$ defined by the template γ. By specifying the set of relevant features for the action, γ enabled us to select one specific interpretation of the action based on the starting state $\mathbf{s}_{t_s}$.

Without such expert knowledge, we treat the true intention of the teacher γ as a latent random variable. Accordingly, we compute the distribution of action goals $p(\mathbf{s}_{t_e} \mid \mathbf{s}_{t_s}, a)$ by integrating over all potential interpretations Γ, i.e.,

$$p(\mathbf{s}_{t_e} \mid \mathbf{s}_{t_s}, a) = \int_\Gamma p(\mathbf{s}_{t_e} \mid \mathbf{s}_{t_s}, a, \gamma)\, p(\gamma \mid \mathbf{s}_{t_s}, a) d\gamma, \tag{4.15}$$

where $p(\gamma \mid \mathbf{s}_{t_s}, a)$ is the template-relevance probability capturing the importance of template γ for modeling the action in $\mathbf{s}_{t_s}$, see Section 4.2.3. In practice, however,

considering all possible action templates is infeasible as there are numerous ways of selecting subsets of spatial relations for modeling the action. Moreover, we do not have prior knowledge about the relevance $p(\gamma \mid \mathbf{s}_{t_s}, a)$ of each template.

Therefore, and rather than relying on an expert to explicitly specify these models, we seek an implicit model of the action goals that encodes different ways of interpreting the teacher demonstrations at the same time. We assume that the intention of the teacher is to move objects in $\mathcal{O}_M^a$ to achieve desirable spatial relations relative to stationary objects $\mathcal{O}_S^a$. Accordingly, we approximate Eq. (4.15) using a finite set of templates $\Gamma = \{\gamma_1, \ldots, \gamma_{|\mathcal{O}_S^a|}\}$ that we construct automatically based on the stationary objects $\mathcal{O}_S^a$ of the action. Specifically, each object $o_s \in \mathcal{O}_S^a$ represents one interpretation of the action by "voting" for goal poses of the moving objects relative to it.

In the following, we break down computing the goal distribution in Eq. (4.15) by first considering the distribution $p(\mathbf{s}_{t_e} \mid \mathbf{s}_{t_s}, a, \gamma_s)$ of goal states with respect to one stationary object modeled by γ_s (Section 4.3.3.1) before constructing the overall goal distribution as a mixture of all such templates based on their relevance $p(\gamma_s \mid \mathbf{s}_{t_s}, a)$ (Section 4.3.3.2).

4.3.3.1. Action Goals Relative to One Stationary Object

We assume that applying an action maintains all rigid transforms between the moving objects in $\mathcal{O}_M^a$. For example, moving the end-effector along with a grasped cup results in the same relative pose between those two objects in both $\mathbf{s}_{t_s}$ and $\mathbf{s}_{t_e}$. Therefore, we model the distribution of goal states $\mathbf{s}_{t_e}$ based on the pose of one representative object $o_* \in \mathcal{O}_M^a$ relative to all other objects in $\mathcal{O}_S^a$.

Furthermore, we consider a finite set of templates Γ, each interpreting the action using relations between o_* and one stationary object $o_s \in \mathcal{O}_S^a$ only. Accordingly, for each template, we define the distribution $p(\mathbf{s}_{t_e} \mid \mathbf{s}_{t_s}, a, \gamma_s)$ of goal states to reflect the intended poses $\mathbf{T}_*(t_e)$ of o_* in the goal state $\mathbf{s}_{t_e}$. Here, γ_s denotes the template that considers only object o_s to be relevant for the action. As in Section 4.3.1, we adopt a non-parametric model for this distribution based on the training poses for the relation ${}^s\mathbf{T}_*$ as observed at the end of each demonstration.

Let ${}^sV_*$ denote the set of these poses (votes), i.e., ${}^sV_* = \{{}^s\mathbf{T}_*^{(1)}(t_e), \ldots, {}^s\mathbf{T}_*^{(N_s)}(t_e)\}$, where $N_s \leq N_a$ is the number of demonstrations involving o_s and ${}^s\mathbf{T}_*^{(v)}(t_e) \in {}^sV_*$ is the pose of o_* relative to o_s at the end of the v-th demonstration. To use these votes in a new starting state $\mathbf{s}_{t_s}$, we transform them to the world frame using the pose $\mathbf{T}_s(t_s)$ of the voting object in the scene. We use the resulting transformed poses $\mathbf{T}_* = \mathbf{T}_s(t)\, {}^s\mathbf{T}_*^{(v)}(t_e)$ for modeling a multi-modal goal distribution using kernel density estimation:

$$p(\mathbf{s}_{t_e} \mid \mathbf{s}_{t_s}, a, \gamma_s) = \sum_{v=1}^{N_s} \frac{1}{N_s} \mathbf{k}\left(\mathbf{T}_s(t_s)\, {}^s\mathbf{T}_*^{(v)}(t_e), \mathbf{T}_*(t_e)\right), \tag{4.16}$$

where $\mathbf{k}(\cdot, \cdot)$ is the Gaussian kernel as in Eq. (4.11).

This continuous distribution models the state $\mathbf{s}_{t_s}$ in terms of the goal pose $\mathbf{T}_*(t_e)$ of o_*. Computing the pose for all other moving objects $o_m \in \mathcal{O}_M^a$ in $\mathbf{s}_{t_e}$ is straightforward since we assume all transforms ${}^*\mathbf{T}_m$ between moving objects to stay the same before and after applying the action, i.e., $\mathbf{T}_m(t_e) = \mathbf{T}_*(t_e)\,{}^*\mathbf{T}_m(t_s)$. Moreover, the pose of each stationary object $o_s \in \mathcal{O}_S^a(\mathbf{s}_{t_s})$ is simply its pose before applying the action, i.e., $\mathbf{T}_s(t_e) = \mathbf{T}_s(t_s)$. Finally, this distribution captures multiple modes of the action goal with respect to object o_s as demonstrated by the teacher. As it depends on the starting pose $\mathbf{T}_s(t_s)$ of o_s in the starting state, we are able to generalize the goal poses ${}^sV_*$ demonstrated by teacher to new starting states.

4.3.3.2. Combined Action Goals Distribution

Eq. (4.16) models the action goals by assuming that the intention of the teacher is to move object o_* relative to only one stationary object o_s. Given a small number of demonstrations, it is typically unclear which other interpretations are also relevant for the action based on the stationary objects in the scene. To tackle this ambiguity, we construct a distribution $p(\mathbf{s}_{t_e} \mid \mathbf{s}_{t_s}, a)$ of goal states as a mixture of all such interpretations. In other words, we allow all $\mathcal{O}_S^a(\mathbf{s}_{t_s})$ in the scene to contribute by voting for the goal pose of o_* as in Eq. (4.16). This approximates Eq. (4.15) using the finite set of templates Γ, each considering one stationary object, i.e.,

$$\begin{aligned} p(\mathbf{s}_{t_e} \mid \mathbf{s}_{t_s}, a) &= \int_\Gamma p(\mathbf{s}_{t_e} \mid \mathbf{s}_{t_s}, a, \gamma_s)\, p(\gamma_s \mid \mathbf{s}_{t_s}, a) d\gamma \\ &\approx \sum_{\gamma_s \in \Gamma} p(\mathbf{s}_{t_e} \mid \mathbf{s}_{t_s}, a, \gamma_s)\, p(\gamma_s \mid \mathbf{s}_{t_s}, a) \\ &\approx \sum_{\gamma_s \in \Gamma} p(\gamma_s \mid \mathbf{s}_{t_s}, a) \sum_{v=1}^{N_s} \frac{1}{N_s} \mathbf{k}\left(\mathbf{T}_s(t_s)\, {}^s\mathbf{T}_*^{(v)}(t_e), \mathbf{T}_*(t_e)\right). \end{aligned} \tag{4.17}$$

Without prior knowledge about the relevance of each template γ_s (i.e., stationary object o_s), we assume the relevance probability $p(\gamma_s \mid \mathbf{s}_{t_s}, a)$ to be uniformly distributed over all templates, i.e., all stationary objects contribute votes with equal weight to the goal distribution. At the same time, a template γ_s is irrelevant for applying the action in state $\mathbf{s}_{t_s}$ if the corresponding stationary object is not in the scene, i.e.,

$$p(\gamma_s \mid \mathbf{s}_{t_s}, a) := \begin{cases} \frac{1}{|\mathcal{O}_S^a|} & \text{if } o_s \in \mathcal{O}_S^a(\mathbf{s}_{t_s}) \\ 0 & \text{if } o_s \notin \mathcal{O}_S^a(\mathbf{s}_{t_s}). \end{cases} \tag{4.18}$$

Therefore, we express Eq. (4.17) in terms of the stationary objects $\mathcal{O}_S^a(\mathbf{s}_{t_s})$ that exist in the starting state $\mathbf{s}_{t_s}$ and their votes:

$$p(\mathbf{s}_{t_e} \mid \mathbf{s}_{t_s}, a) = \eta \sum_{o_s \in \mathcal{O}_S^a(\mathbf{s}_{t_s})} \sum_{v=1}^{N_s} \mathbf{k}\left(\mathbf{T}_s(t_s)\, {}^s\mathbf{T}_*^{(v)}(t_e), \mathbf{T}_*(t_e)\right), \tag{4.19}$$

where η is a normalizer. Note that in this way, we model the intention of the teacher in a flexible, data-driven way. We are able to generalize the demonstrations and apply the action even if some of the stationary objects are missing in the scene. Moreover, we do not commit to a particular model of the goals beforehand, but change the shape of the distribution "on the fly" based on the starting state. By aggregating the transformed votes from all stationary objects, the peaks of the resulting distribution will correspond to likely modes of the intention of the teacher based on the current state.

Note that Eq. (4.19) is only defined if there is at least one stationary object in the scene. Otherwise, the action cannot be applied. Finally, our model is generic with respect to which representative moving object o_* is selected to construct the distribution. In this work, we randomly select an object other than the end-effector if $|\mathcal{O}_M^a| \geq 2$ to model the goal of the action. Otherwise, we select the end-effector as o_*.

4.3.4. Discretized Action Models

The above section described how we construct a continuous distribution of action goals. Given a new starting state $\mathbf{s}_{t_s}^*$, we use the teacher demonstrations to construct this distribution as in Eq. (4.19). To apply the action, we seek a goal state $\mathbf{s}_{t_e}^*$ that maximizes $p(\mathbf{s}_{t_e} \mid \mathbf{s}_{t_s}^*, a)$ subject to $\mathbf{s}_{t_e}^*$ being feasible, see Section 4.1.6. In general, this is challenging due to the large state space and the highly non-convex nature of the distribution. In principle, we are interested in goal states corresponding to regions of high (voting) density in the distribution. One way to address this is in practice is to rely on a sampling-based approach. For instance, we draw samples from Eq. (4.19) by viewing it as a mixture of Gaussians distribution over $\mathbf{T}_*$ with each Gaussian component centered at one transformed vote for the pose of o_*. However, such a method could require a large number of samples before finding a feasible state that maximizes the goal likelihood.

Therefore, in this section, we propose a more efficient solution by approximating the above distribution with a finite set of discrete goal states and their probabilities. Specifically, we seek a finite set of S goal states $\hat{\mathcal{S}}_{t_e} = \{\hat{\mathbf{s}}_{t_e}^{(1)}, \ldots, \hat{\mathbf{s}}_{t_e}^{(S)}\}$, along with their probabilities $P(\mathbf{s}_{t_e} = \hat{\mathbf{s}}_{t_e}^{(s)} \mid \mathbf{s}_{t_s}, a)$ for $s = 1, \ldots, S$, to approximate the distribution in Eq. (4.19). As this distribution depends on the initial state, the number S of goal states is unknown beforehand.

One approach to achieve this would be to discretize the whole state space in a regular grid and consider each cell as one potential (discrete) state. Although correct, this would require many unnecessary computations for exploring states with low probabilities, since only a small part of the state space is occupied with points corresponding to the teacher demonstrations. Instead, we rely on clustering to achieve this discretization by determining regions of the state space with a high probability density in Eq. (4.19). In this way, we are able to efficiently select the best goal state for a given situation while

still reasoning about the complete spectrum of intended goals.

In the following, we first describe how we construct a discretized voting distribution for each stationary object in order to efficiently draw votes for the goal poses of the moving objects (Section 4.3.4.1). Accordingly, we describe how we use these votes to compute a discretized version of the distribution in Eq. (4.19) (Section 4.3.4.2), which allows us to efficiently select the most likely goal state for applying the action (Section 4.3.4.3).

4.3.4.1. Pairwise Voting Distributions

We constructed the distribution of goal states in Eq. (4.19) by relying on all votes ${}^{s}V_{*} = \{{}^{s}\mathbf{T}_{*}^{(1)}(t_e), \ldots, {}^{s}\mathbf{T}_{*}^{(N_s)}(t_e)\}$ representing poses of o_* relative to $o_s \in \mathcal{O}_S^a$ at the end of each teacher demonstration. Each (transformed) vote corresponds to one component in the final distribution of goal poses for o_* in the world frame. We build on this motivation to achieve our approximate discretized model.

However, rather than using all N_s votes each time we apply the action, we associate each stationary object o_s with a discrete voting distribution $\hat{p}_s$ over relative poses ${}^{s}\mathbf{T}_{*}$, such that a relative pose can take one of $\hat{N}_s \leq N_s$ possible values ${}^{s}\hat{V}_{*} = \{{}^{s}\hat{\mathbf{T}}_{*}^{(1)}, \ldots, {}^{s}\hat{\mathbf{T}}_{*}^{(\hat{N}_s)}\}$. Each possible pose ${}^{s}\hat{\mathbf{T}}_{*}^{(n)} \in {}^{s}\hat{V}_{*}$ represents a vote for the relative transform between o_* and o_s, and is associated with a weight $\hat{w}_s^{(n)}$ equal to its probability $\hat{p}_s({}^{s}\mathbf{T}_{*} = {}^{s}\hat{\mathbf{T}}_{*}^{(n)})$. To construct the voting distribution, we aim to "compress" the demonstrations ${}^{s}V_{*}$ by merging poses that are close to each other into one discrete vote ${}^{s}\hat{\mathbf{T}}_{*}^{(n)}$. The distribution $\hat{p}_s$ therefore reflects the importance of each vote ${}^{s}\hat{\mathbf{T}}_{*}^{(n)}$ based on how many original poses ${}^{s}\mathbf{T}_{*}^{(v)}(t_e)$ were used to compute it. The intuition behind this is that pairwise relations that are important for the action are typically demonstrated more often by the teacher, resulting in multiple relative poses that are close in distance.

We therefore rely on clustering to merge similar poses in ${}^{s}V_{*}$. Instead of specifying the number of clusters, as is required by techniques such as k-means clustering, we adopt a hierarchical clustering approach and impose a compactness $\delta_{\max}$ specifying the maximum allowed distance between any two poses in one cluster. Specifically, we use the complete-linkage version of agglomerative hierarchical clustering. Starting with each of the N_s poses as its own cluster, we iteratively merge clusters together until there are no two clusters left such that the maximum distance (Eq. (4.12)) between any of their poses is less than or equal to $\delta_{\max}$. The result is $\hat{N}_s$ clusters $\hat{C}_s^{(1)}, \ldots, \hat{C}_s^{(\hat{N}_s)}$, where $\hat{N}_s \leq N_s$. Moreover, each demonstrated pose ${}^{s}\mathbf{T}_{*}^{(v)}(t_e)$ from ${}^{s}V_{*}$ belongs to one of the resulting clusters, which we denote by ${}^{s}\mathbf{T}_{*}^{(v)}(t_e) \in \hat{C}_s^{(n)}$.

Based on this, we construct the set of discretized votes ${}^{s}\hat{V}_{*} = \{{}^{s}\hat{\mathbf{T}}_{*}^{(1)}, \ldots, {}^{s}\hat{\mathbf{T}}_{*}^{(\hat{N}_s)}\}$ by computing the mean pose ${}^{s}\hat{\mathbf{T}}_{*}^{(n)}$ of each cluster $\hat{C}_s^{(n)}$. We compute the mean of a set of poses by considering the translation and rotational components independently and computing the mean on the corresponding topological space ($\mathbb{R}^3$ and $SO(3)$).

By setting the compactness $\delta_{\max}$ of each cluster, we have an upper bound on the

maximum distance between any two poses inside it. Therefore, clusters with more poses in them correspond to regions of space with a higher density of desirable relative poses. Accordingly, we compute the weight $\hat{w}_s^{(n)}$ associated with each cluster $\hat{C}_s^{(n)}$ as the ratio of poses that belong to it, i.e.,

$$\hat{w}_s^{(n)} = \frac{1}{|{}^sV_*|} \sum_{v=1}^{N_s} \mathbf{1}_{\hat{C}_s^{(n)}}({}^s\mathbf{T}_*^{(v)}(t_e)), \tag{4.20}$$

where $\mathbf{1}_{\hat{C}_s^{(n)}}(\cdot)$ is the indicator function, which is equal to 1 if ${}^s\mathbf{T}_*^{(v)}(t_e) \in \hat{C}_s^{(n)}$ and to 0 otherwise. In fact, by viewing $\hat{p}_s$ as a multinomial distribution over ${}^s\mathbf{T}_*$ with $\hat{N}_s$ possible outcomes, Eq. (4.20) corresponds to the maximum likelihood estimate for the probability $\hat{w}_s^{(n)}$ of outcome ${}^s\hat{\mathbf{T}}_*^{(n)}$ (i.e., ${}^s\mathbf{T}_*$ falling in $\hat{C}_s^{(n)}$).

4.3.4.2. Discretized Action Goal States

We now use the discrete voting distribution $\hat{p}_s$ associated with each each stationary object to compute a discretized version of the goal distribution in Eq. (4.19). As before, we aim to construct a multi-modal distribution capturing different ways of applying the action by mixing the votes by all stationary objects in the scene. However, instead of using all N_s votes ${}^s\mathbf{T}_*^{(v)}(t_e)$ demonstrated by the teacher, we rely on I samples ${}^s\widetilde{\mathbf{T}}_*^{(i)}$ from the voting distribution of each object o_s. We draw these samples with replacement such that the probability of sampling a vote ${}^s\hat{\mathbf{T}}_*^{(n)}$ is $\hat{w}_s^{(n)}$, see Eq. (4.20). Given a specific template γ_s representing a stationary object o_s, we use the I samples, transformed to the world frame, to approximate the distribution $p(\mathbf{s}_{t_e} \mid \mathbf{s}_{t_s}, a, \gamma_s)$ in Eq. (4.16) as follows:

$$p(\mathbf{s}_{t_e} \mid \mathbf{s}_{t_s}, a, \gamma_s) \approx \frac{1}{I} \sum_{i=1}^{I} \mathbf{k}\left(\mathbf{T}_s(t_s)\, {}^s\widetilde{\mathbf{T}}_*^{(i)}, \mathbf{T}_*(t_e)\right). \tag{4.21}$$

As in Eq. (4.17), we mix the votes from all stationary objects in the scene, weighted by their relevance $p(\gamma_s \mid \mathbf{s}_{t_s}, a)$, to construct the overall distribution of goals $p(\mathbf{s}_{t_e} \mid \mathbf{s}_{t_s}, a)$ as follows:

$$\begin{aligned} p(\mathbf{s}_{t_e} \mid \mathbf{s}_{t_s}, a) &\approx \widetilde{\eta} \sum_{\gamma_s \in \Gamma} p(\gamma_s \mid \mathbf{s}_{t_s}, a) \sum_{i=1}^{I} \mathbf{k}\left(\mathbf{T}_s(t_s)\, {}^s\widetilde{\mathbf{T}}_*^{(i)}, \mathbf{T}_*(t_e)\right), \\ &\approx \widetilde{\eta} \sum_{\gamma_s \in \Gamma} p(\gamma_s \mid \mathbf{s}_{t_s}, a) \sum_{i=1}^{I} \mathbf{k}\left(\widetilde{\mathbf{T}}_*^{(i)}, \mathbf{T}_*(t_e)\right), \end{aligned} \tag{4.22}$$

where I is the number of samples ${}^s\widetilde{\mathbf{T}}_*^{(i)}$ drawn from each voting distribution $\hat{p}_s$ of stationary object o_s, $\widetilde{\mathbf{T}}_*^{(i)} = \mathbf{T}_s(t_s)\, {}^s\widetilde{\mathbf{T}}_*^{(i)}$ is the transformed i-th vote sample, and $\widetilde{\eta}$ is a normalizer.

We note that Eq. (4.22) describes a mixture distribution based on a total of J votes from all stationary objects in the scene, i.e., $J = I\,|\Gamma|$. As in Section 4.3.3.2, each of the

J votes defines one possible goal state $\widetilde{\mathbf{s}}_{t_e}^{(j)}$ based on the corresponding pose $\widetilde{\mathbf{T}}_*$ of o_*. Moreover, each state $\widetilde{\mathbf{s}}_{t_e}^{(j)}$ is associated with a weight $\widetilde{w}_{\gamma_s}^{(j)}$, which is equal to the relevance $p(\gamma_s \mid \mathbf{s}_{t_s}, a)$ of the template γ_s responsible for the corresponding vote.

To discretize this distribution, we seek a finite set of goal states $\hat{\mathcal{S}}_{t_e} = \{\hat{\mathbf{s}}_{t_e}^{(1)}, \ldots, \hat{\mathbf{s}}_{t_e}^{(S)}\}$, along with their probabilities $P(\mathbf{s}_{t_e} = \hat{\mathbf{s}}_{t_e}^{(s)} \mid \mathbf{s}_{t_s}, a)$ for $s = 1, \ldots, S$. This approximates Eq. (4.22) such that the probability of any state not in $\hat{\mathcal{S}}_{t_e}$ is zero. For this, we are interested in determining regions in state space with a high density with respect to the continuous distribution. As in Section 4.3.4.1, we accomplish this using agglomerative hierarchical clustering on the set of goal state samples $\widetilde{\mathcal{S}}_{t_e} = \{\widetilde{\mathbf{s}}_{t_e}^{(1)}, \ldots, \widetilde{\mathbf{s}}_{t_e}^{(j)}\}$. We perform the clustering with respect to the poses of the moving object o_* with the same compactness threshold $\delta_{\max}$ used for learning the voting distributions $\hat{p}_s$. Starting with each state $\widetilde{\mathbf{s}}_{t_e}^{(j)}$ as its own cluster, we iteratively merge clusters until no two clusters exist such that the maximum distance between any of their states (i.e., the distance between the corresponding $\widetilde{\mathbf{T}}_*$ poses) is less than or equal to $\delta_{\max}$.

The result is a set of S clusters $\hat{\mathcal{S}}^{(1)}, \ldots, \hat{\mathcal{S}}^{(S)}$ that partition the samples $\widetilde{\mathcal{S}}_{t_e}$. We compute each desired discrete goal state $\hat{\mathbf{s}}_{t_e}^{(s)}$ as the mean state of the corresponding cluster $\hat{\mathcal{S}}^{(s)}$ by averaging the poses of each object independently. Moreover, since all clusters have the same upper bound on the distance between any two states inside them, we approximate the probability of each goal state $\hat{\mathbf{s}}_{t_e}^{(s)}$ using the density of state samples that fall in the corresponding cluster $\hat{\mathcal{S}}^{(s)}$. We do this by counting the (weighted) state samples in the cluster, i.e.,

$$\begin{aligned} P(\mathbf{s}_{t_e} = \hat{\mathbf{s}}_{t_e}^{(s)} \mid \mathbf{s}_{t_s}, a) &\approx \int_{B(\hat{\mathcal{S}}^{(s)})} p(\mathbf{s}_{t_e} \mid \mathbf{s}_{t_s}, a)\, d\mathbf{s}_{t_e} \\ &\approx \hat{\eta} \sum_{j=1}^{J} \widetilde{w}_{\gamma_s}^{(j)} \mathbf{1}_{\hat{\mathcal{S}}^{(s)}}(\widetilde{\mathbf{s}}_{t_e}^{(j)}), \end{aligned} \tag{4.23}$$

where $B(\hat{\mathcal{S}}^{(s)})$ is the volume in state space corresponding to cluster $\hat{\mathcal{S}}^{(s)}$ and $\mathbf{1}_{\hat{\mathcal{S}}^{(s)}}(\cdot)$ is the indicator function that is equal to 1 if $\widetilde{\mathbf{s}}_{t_e}^{(j)} \in \hat{\mathcal{S}}^{(s)}$ and to 0 otherwise. Moreover, $\hat{\eta} = 1/\left(\sum_{s=1}^{S}\sum_{j=1}^{J} \widetilde{w}_{\gamma_s}^{(j)} \mathbf{1}_{\hat{\mathcal{S}}^{(s)}}(\widetilde{\mathbf{s}}_{t_e}^{(j)})\right)$ is a normalizer. We illustrate the overall process of computing the goal distribution of an action according to our approach with an example in Figure 4.10.

Note that this distribution is conditioned on the starting state $\mathbf{s}_{t_s}$ in terms of which stationary objects exist in the scene and what their poses are. In this way, we reason about a finite set of potential goals when applying the action while still maintaining the flexibility of the continuous model in Section 4.3.3.2. Moreover, the clustering compactness threshold $\delta_{\max}$ defines a degree of coarseness for the discretization of goals. The smaller $\delta_{\max}$ is, the closer are the pairwise relations (relative poses) in the final goal states to those demonstrated by the teacher. As $\delta_{\max} \to 0$, we construct voting distributions $\hat{p}_s$ where each possible vote corresponds to one of the N_s poses

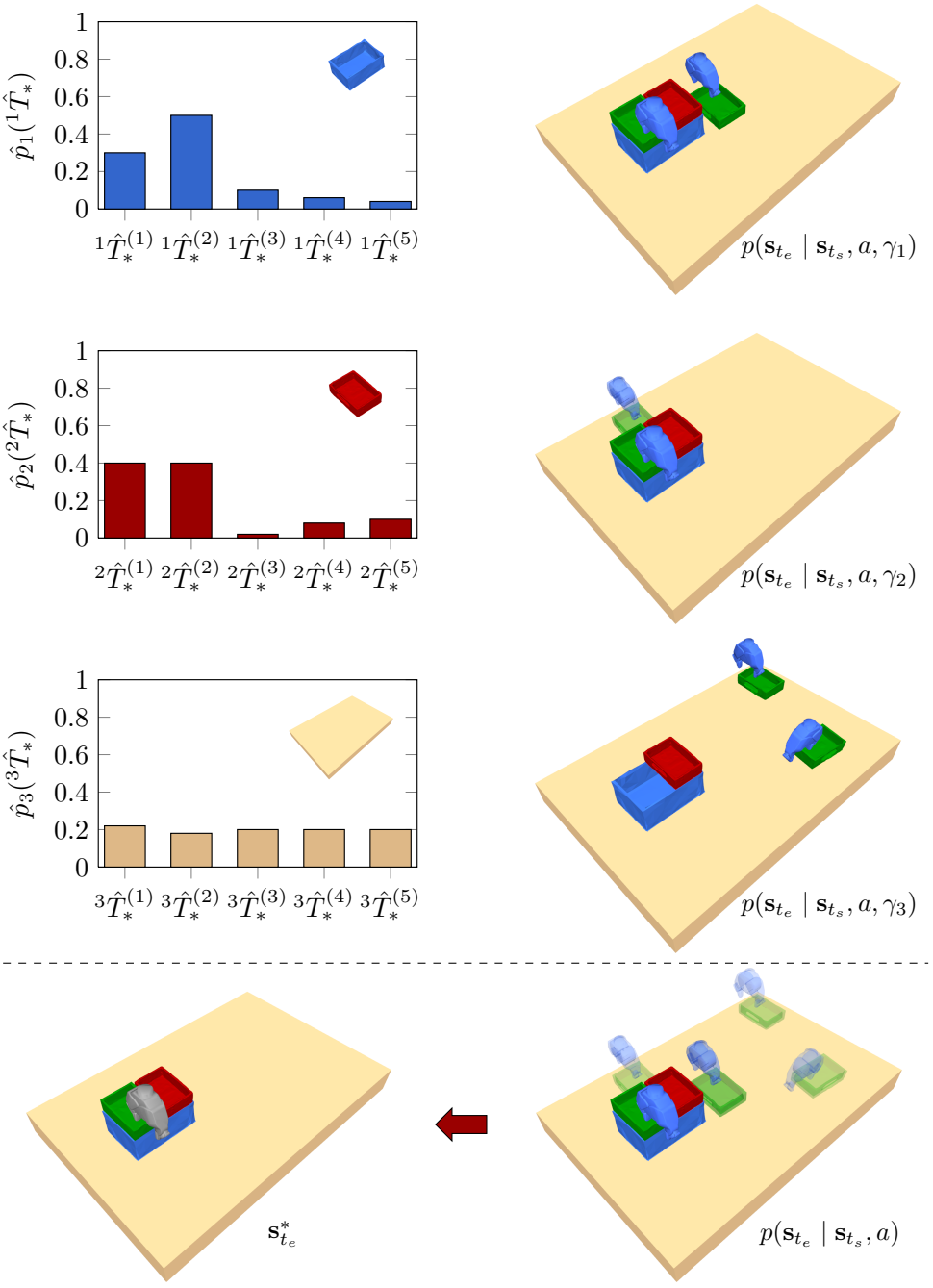


Figure 4.10.: Constructing the goal distribution for the action of moving the green box (see Figure 4.3). Top: we draw samples from the voting distributions (left) $\hat{p}_s$ of the three stationary objects to compute likely goal poses of the moving objects (right). Bottom-right: we aggregate all votes to construct the multi-modal mixture distribution, which we discretize via clustering (the color opacity reflects the probability of each goal state). Our approach tackles the ambiguity in the demonstrations by reasoning about likely goals in a data-driven way based on the initial state. Bottom-left: we select the feasible goal state with the largest probability to apply the action.

demonstrated by the teacher, resulting in a finer imitation of the action. On the other hand, larger values of $\delta_{\max}$ correspond to merging demonstrated relations that are close to each other, resulting in fewer discrete states that average the corresponding demonstrations.

Finally, the shape of this distribution depends on two main ingredients: the voting distribution $\hat{p}_s$ of each stationary object o_s, and the associated weight $\tilde{w}_{\gamma_s}^{(j)}$. These respectively determine which goal states are sampled and how strongly they are considered in the final distribution. In this work, we rely only on the teacher demonstrations to learn $\hat{p}_s$ (Section 4.3.4.1), and assume that all templates (i.e., stationary objects) have equal influence on the action, see Eq. (4.18). In Chapter 5, we present an approach that enables the robot to re-learn these distributions using its own experience of sequencing different actions to solve a complex task.

4.3.4.3. Selecting a Goal State

Having computed a finite set of goal states $\hat{\mathcal{S}}_{t_e} = \{\hat{\mathbf{s}}_{t_e}^{(1)}, \ldots, \hat{\mathbf{s}}_{t_e}^{(S)}\}$, and their probabilities, $P(\mathbf{s}_{t_e} = \hat{\mathbf{s}}_{t_e}^{(s)} \mid \mathbf{s}_{t_s}, a)$, we solve the problem in Section 4.1.6 by selecting the best goal state $\mathbf{s}_{t_e}^* \in \hat{\mathcal{S}}_{t_e}$. For this, we first exclude the infeasible sates in $\hat{\mathcal{S}}_{t_e}$, and simply return the state with the highest probability out of the remaining ones.

4.4. Related Work

In this chapter, we addressed the problem of learning models of manipulation actions from teacher demonstrations. Learning from demonstrations is a framework for transferring knowledge from non-expert users to robots (see the surveys by Billard et al. (2008); Argall et al. (2009)). The techniques we presented in this chapter enable a robot to reason about the relevant spatial constraints for reproducing an action despite ambiguity in the demonstrations of the teacher and a small number of training examples. In this section, we discuss prior work in the literature related to the problem of learning manipulation skills from teacher demonstrations.

Feature Selection and Abstraction Learning One key problem in learning manipulation actions from demonstrations is determining the relevant spatial constraints for reproducing an action. The most common representation of such constraints is in terms of the reference frames useful for reproducing a demonstrated motion. Whereas our work enables a robot to reason about several valid interpretations of the demonstrations with respect to these constraints, other approaches rely on interacting with the teacher to specify this information in terms of keyframes during the demonstrations (e.g., Akgun et al. (2012); Alexandrova et al. (2014); Akgun and Thomaz (2016)). On the other hand, the approach by Mühlig et al. (2009b) automatically selects task spaces for modeling

demonstrated motions using different criteria such as attention to salient objects or inter-trial variance. Related to that, Eppner et al. (2009) and Mühlig et al. (2009a) consider the variance in the demonstrations to determine less relevant parts of the tasks. The approach by Jäkel et al. (2011) uses demonstrations to generate a so-called strategy graph that segments tasks into sub-goals. An evolutionary algorithm is used to eliminate irrelevant spatial and temporal constraints using a motion planner in simulation.

The above works focus on selecting the relevant spatial constraints that enable a robot to reproduce demonstrated trajectories. In contrast to this, our work focuses on point-to-point actions and enables a robot to select the best goal state when reproducing an action based on the initial state by constructing multi-modal distributions of goals. We model these distributions either based on subsets of spatial constraints (expert templates) or as mixture models in a data-driven manner (implicit action models).

The problem of feature selection for modeling actions is also closely related to the problem of learning a suitable feature abstraction. In the context of hierarchical reinforcement learning, Konidaris and Barto (2009) presented an approach for choosing between different state abstractions to enable an agent to learn options (macro actions) out of primitive skills. Konidaris et al. (2012) leveraged this method when segmenting demonstrated trajectories into different options represented in a skill tree. Each option is assigned to a different abstraction that defines a small subset of relevant variables using a trade-off between model likelihood and model complexity. Similarly, our approach based on expert templates enables reasoning about different subsets of features to model a new action. However, we focus on enabling a robot to select a suitable model based on the initial state and do not tackle the problem of decomposing tasks into sequences of actions.

Whereas the above works considered learning abstractions in continuous domains, other approaches have addressed learning symbolic abstractions to leverage high-level, symbolic planners for solving tasks. For example, Veeraraghavan and Veloso (2008) learn symbolic representations of actions for planning by instantiating preprogrammed behaviors and learning the corresponding preconditions and effects. Moreover, Pasula et al. (2007) learn action models as probabilistic rules based on a dictionary of basic symbolic predicates for planning. Such rules have recently been successfully used in planning to solve complex manipulation tasks (Lang and Toussaint, 2010) or to learn kinematic models of articulated objects (Höfer et al., 2014). Whereas existing symbolic state abstractions facilitate planning complex tasks by reducing the dimensionality of the problem, we do not assume to have such prior semantic knowledge about demonstrated actions. Additionally, the approaches above require an expert to provide the mapping between the symbolic and continuous state representations.

To address this, other researchers proposed methods to learn both symbolic state abstractions and the mapping to geometric features to represent actions. In our previous work, we presented an approach to learn the relevant preconditions and effects of actions

by clustering the endpoints of segmented demonstrations and expressing the resulting models symbolically (Abdo et al., 2013). Recently, Cubek et al. (2015) presented an approach that infers relevant concepts from teacher demonstrations of manipulation actions using subspace clustering such that action effects can be reproduced using a symbolic planner. Moreover, Konidaris et al. (2014) presented an approach to construct a symbolic representation for planning in a continuous, low-level environment. They use supervised learning over continuous features to represent preconditions and effects. Recently, they extended this approach to a probabilistic formulation that captures the uncertainty in the learned action models (Konidaris et al., 2015). On the other hand, our implicit action models approach enables learning continuous, multi-modal distributions of action goals based on relative poses between the objects. We use hierarchical clustering to discretize these distributions by identifying regions of the state space corresponding to goals with a high likelihood. Whereas these clusters can be seen as analogous to symbolic state abstractions, we do not tackle the problem of sequencing actions using symbolic planners. In Chapter 5, we present a novel approach that leverages these models to plan sequences of actions for solving tasks without requiring symbolic information.

Learning Action Effects Based on Object Affordances Several approaches have addressed the problem of inferring action effects by reasoning about object affordances (Gibson, 1979) in terms of feature changes associated with actions (Krüger et al., 2011). For example, Dogar et al. (2007) compute differences of feature vectors such as the distances, shapes, and orientations of objects extracted from laser scans and cluster the difference vectors to represent the effects of actions. Analogously, Ugur et al. (2014) recently proposed an approach to learn affordance classifiers that predict effect categories given object features and based on single-object actions. Accordingly, these predictions are later used to model higher-level affordance features, which the robot leverages to learn more complex, multi-object affordances. Whereas the above methods considered actions such as pushing or stacking objects, the approach of Mar et al. (2015) enables a robot to autonomously learn the affordances of various tools based on visual features. Moreover, the approaches of Montesano et al. (2008) and Rudolph et al. (2010) model actions and their effects probabilistically using Bayesian networks. Moldovan et al. (2012) extended these approaches to learn interactions between multiple objects by leveraging statistical relational learning techniques. Related to that, Song et al. (2010) proposed an approach that models the relations between object- and action-related features using a Bayesian network to learn object grasping strategies. In contrast to such works, we do not consider learning the affordances of objects to reason about using tools or the effect of pushing an object based on its shape. Instead, we address learning point-to-point actions based on spatial constraints between the objects before and after applying an action. Specifically, we enable robots to reason about multiple modes in the intention of the teacher for the demonstrated actions. Finally, rather than relying on

teacher demonstrations, other approaches have leveraged physics simulations to reason about the effects of actions such as pushing objects or pouring liquids during cooking (Weitnauer et al., 2010; Kunze and Beetz, 2015), but this is orthogonal to the problem addressed in this chapter.

Learning Motor Skills from Demonstrations There exists a large spectrum of works that addressed encoding skills on the trajectory level to enable a robot to reproduce a desirable motion or gesture. One popular approach for learning movement primitives is to represent desired motions as stable dynamical systems with well-defined attractor dynamics that can generalize demonstrated trajectories to new goals (e.g., Ijspeert et al. (2002); Pastor et al. (2009)). To learn the attractor landscape of the system, these methods rely on locally weighted regression techniques. Related to that, other approaches leverage Gaussian mixture models to model movement primitives probabilistically (e.g., Calinon et al. (2007, 2010); Khansari-Zadeh and Billard (2011); Welschehold et al. (2016)). On the other hand, the approach by Asfour et al. (2006) models demonstrated movements using hidden Markov models and detects key points across demonstrations that the robot should reproduce. Similarly, Kulic et al. (2012) leverage hidden Markov models to encode, recognize, and reproduce demonstrated motions.

In the context of reinforcement learning, teacher demonstrations have been leveraged to initialize policies for complex motor skills such as opening doors or playing tennis, and relied on techniques such as policy search to enable robots to improve these policies by interacting with the environment (Kober and Peters, 2009; Kalakrishnan et al., 2011; Englert et al., 2013). Additionally, inspired by inverse optimal control, Jetchev and Toussaint (2011) addressed the problem of learning complex grasping policies as one of learning cost functions that reflect the important control dimensions of the skill. Finally, recent advancements in deep learning have enabled robots to learn grasps or manipulation skills without requiring engineered features (Lenz et al., 2015; Pinto and Gupta, 2016; Finn et al., 2016; Levine et al., 2016; Rahmatizadeh et al., 2016).

In contrast to these works, we do not consider learning control policies or primitives on the trajectory level, i.e., *how* to imitate. Instead, we focus on inferring the relevant spatial constraints for reproducing point-to-point actions given a small number of demonstrations, i.e., *what* to imitate. To execute an action based on the learned spatial constraints, our method can be combined with existing motion planners or controllers using the techniques above to generate the required trajectories.

Implicit Shape Models Our approach for learning implicit action models shares similarities with the concept of implicit shape models, a popular method for learning flexible models of object shapes in order to recognize them in 2D images (Leibe et al., 2004; Leibe and Schiele, 2003). The key idea is to learn a codebook of features from training image patches, and to associate a codeword with a distribution over its position

relative to the center of the target object. This results in learning an implicit model of what the target object looks like. Recognizing a new instance of the object is done by matching parts of an image to learned codebook entries and having them vote for the position of the object. An estimate of the object location is found by searching for local maxima in the voting density using mean-shift. This concept has been successfully applied to different pose estimation and object recognition problems in 3D (Arie-Nachimson and Basri, 2009; Knopp et al., 2010). Recently, Meißner et al. (2014, 2016) presented a hierarchical approach based on implicit shape models to address 3D scene recognition. By learning from scenes demonstrated by a teacher, they leverage implicit shape models to reason about expected object poses for object search. Similarly, our work enables a robot to reason about the likely poses for placing objects based on the current scene and teacher demonstrations.

4.5. Conclusion

In this chapter, we addressed the problem of learning models of point-to-point actions from a small number of teacher demonstrations. Specifically, we focused on inferring the spatial constraints that are relevant for reproducing an action and generalizing it to new initial states. This is challenging due to the small number of training examples, resulting in several valid and often contradictory ways of interpreting the demonstrations. We addressed this from two perspectives. Firstly, we proposed a novel approach that leverages expert-designed models of manipulation actions in general in order to bootstrap learning a new action. Each model represents the action based on a subset of the involved spatial constraints, thereby reducing the dimensionality of the problem. Accordingly, we formulated a model-selection problem that allows the robot to select the most likely model for applying the action based on the initial state. Secondly, we presented a novel approach that learns an implicit model of an action in a data-driven way and without requiring prior expert knowledge about actions. The idea of this approach is to allow the stationary objects of the action to vote for the likely poses of the moving objects. This allows the robot to construct a multi-modal distribution of intended action goals based on a mixture of interpretations of the demonstrations. By conditioning this distribution on the initial state, our approach is able to select the best goal state to achieve based on the stationary objects in the scene and their poses. This provides the flexibility to adapt to new initial states and without committing to a single model beforehand.

The majority of techniques for learning from demonstrations focus on learning the action on the motion level to reproduce demonstrated trajectories and achieve a specified goal. In contrast to that, our work considered point-to-point actions that do not depend on the dynamics of the motion. Accordingly, we focused on learning the relevant spatial constraints between objects before and after applying an action. In this way, we

eliminate the need for an expert to provide a suitable model or set of features to use for learning each new action. Additionally, we are able to reason about multiple modes in the intention of the teacher with respect to the goal states to achieve.

Finally, in this chapter, we focused on selecting the most likely goal state when applying a single action. For this, we assumed that the action is applicable based on the affordability of the initial state. In the next chapter, we leverage the implicit action models introduced in this chapter and tackle the problem of sequencing several actions to solve a task. In this context, we aim to select action sequences and goals such that we apply an action in states with high affordability. We demonstrate how our implicit action models provide our approach with the flexibility to select between different ways of applying an action while reasoning about a desirable final task goal to achieve. Additionally, we show how our discretized action goal distributions allow for efficiently sampling likely goal states when planning while still reasoning about desirable spatial relations in a continuous space.

5. Teach and Improvise: Simultaneously Inferring Task and Action Goals

In Chapter 3, we addressed the problem of learning task goals and focused on predicting user preferences with respect to object arrangements. We assumed the robot is able to achieve predicted arrangements by manipulating objects using existing action models and planners. Alternatively, Chapter 4 addressed learning new action models from teacher demonstrations. In this chapter, we tackle the problem of learning both task goals and action models simultaneously from non-expert teacher demonstrations. We introduce *teach and improvise*, a novel learning from demonstrations framework that enables a robot to compute plans for solving a manipulation tasks by generalizing and sequencing actions based on a small number of ambiguous demonstrations. We build our work on Monte Carlo tree search to reason about multiple interpretations of each action while searching for promising task solutions. A key contribution of our approach is that we do not require the user to specify an explicit goal state for the robot to solve the task. Instead, we formulate an optimization problem that aims at maximizing the likelihood of satisfying the intention of the teacher with respect to relevant spatial relations. This enables the robot to improvise feasible and desirable task solutions in an anytime fashion. Moreover, our approach enables the robot to continuously improve its performance by *practicing* the task in simulation and subsequently updating the learned action models without requiring additional teacher demonstrations. Through extensive experiments in simulation and with a real robot, we demonstrate the effectiveness and flexibility of our approach in improvising task solutions from arbitrary initial states.

So far in this thesis, we considered the problem of learning task goals and action models individually. On the one hand, in Chapter 3, we focused on learning user

preferences with respect to object arrangements using large amounts of data gathered from different users and environments. To achieve those arrangements in practice, we relied on expert-designed pick-and-place actions that the robot combined using existing planners. On the other hand, as it is infeasible for an expert to provide such actions for all tasks and domains, Chapter 4 tackled the problem of learning action models from non-expert teacher demonstrations. For example, this enables the robot to learn how to reach for a new object in order to grasp it, or where to place it relative to other objects in the scene.

In this chapter, we address the problem of jointly learning both task goals and action models from a teacher. Specifically, we consider the scenario of a teacher demonstrating to the robot how to manipulate objects in the scene, step by step, in order to achieve the goal of a new task. As in Chapter 3, we focus on tasks whose goals can be modeled in terms of the final object configuration. However, in this chapter, we consider learning arbitrary object arrangements of interest to a user (e.g., setting a table, arranging blocks, etc.) that are not confined to tidying up objects in containers.

For some domains, there exists a predefined library of logical predicates or relations that facilitate learning models for solving such tasks. In those cases, the robot could leverage such a library to abstract continuous states and make planning feasible. For instance, in Chapter 3, we assumed that the goal of the task is modeled in terms of placing pairs of objects in the same container, and focused on learning the preference of a user for such relations. Similarly, for learning action models, one could build on existing predicates to instantiate or extract symbolic or logical rules from teacher demonstrations (Pasula et al., 2007; Veeraraghavan and Veloso, 2008; Höfer and Brock, 2016). However, this is not always feasible since an expert is needed to program those predicates and provide the robot with the means to ground them based on its continuous sensor data. Moreover, the teacher may demonstrate arbitrary relations to the robot that are not accounted for in such a library, making it difficult to use these predicates to encode all goals it has to learn in the future.

Therefore, in this chapter, we assume to have no prior semantic knowledge about tasks and actions and rely only on a small number of teacher demonstrations to learn them. A major challenge in this context is the ambiguity in the demonstrations. On the one hand, the goal of each action is not always clear as the demonstrations may involve manipulating the objects in different orders. On the other hand, the teacher may demonstrate several valid object configurations representing multiple modes of the task goal, see Figure 5.1. Without prior knowledge of the task, the robot has to generalize these configurations and reproduce the task from new initial states *without an explicit goal being provided each time.* In other words, we cannot rely on traditional task and motion planning techniques, which expect a concrete goal state as input.

Therefore, we propose to formulate this as an optimization problem in which the robot has to maximize the likelihood that the final goal state aligns with the intention of

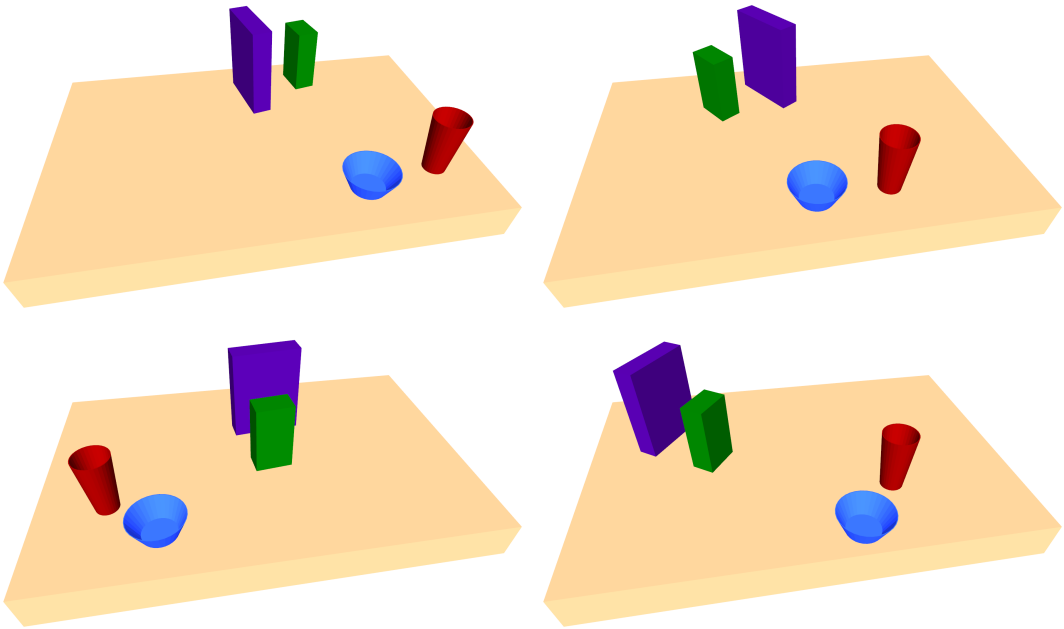


Figure 5.1.: We aim to learn tasks whose goals are modeled based on the spatial relations between the involved objects. This figure shows four examples of valid ways for solving a task as demonstrated by a teacher. Without providing the robot with an explicit formulation for the goal of the task, the robot has to generalize such configurations and reproduce the task from arbitrary initial states. To achieve this, the robot may improvise new goal states based on the demonstrated spatial relations between the objects.

the teacher given the demonstrations. Additionally, we model goals in terms of pairwise spatial relations between objects based on their relative poses. In this way, we bypass the need for an expert to provide predicates that are useful for learning the task, and directly tackle the continuous nature of real-world spatial relations.

However, maximizing the intention of the teacher in this continuous space is challenging and often infeasible without a good initialization. We address this issue by leveraging the discretized implicit action models proposed in Section 4.3.4 to explore the state space using the demonstrations. Accordingly, we propose a novel approach based on Monte Carlo Tree Search to compute plans that maximize the intention of the teacher for the task. We present an algorithm that tackles the ambiguity in the actions and task goals simultaneously. On the one hand, we compute plans by reasoning about the best model (template) to use for reproducing each demonstrated action based on the current state. On the other hand, we guide these decisions based on the final object configuration achieved. As a result, the robot generalizes a small number of ambiguous teacher demonstrations by *improvising* task solutions (i.e., final configurations and ways

of applying the learned actions). We refer to this framework as *teach and improvise*. Our approach is an anytime method in which the robot is able to achieve better solutions given more planning time.

We evaluated our approach extensively using several tasks in simulation and with a real robot. Our results show the ability of our method to use a few demonstration to reproduce the intention of the teacher for the task and achieve desirable spatial relations between the objects. Moreover, our approach is able to compute plans for achieving different valid modes of the task goals based on the initial state and without requiring the user to explicitly specify desired goals.

The remainder of this chapter is organized as follows. We formalize the problem of learning a task and its actions in Section 5.1, in which we also highlight the major challenges involved in this problem. In Section 5.2, we present our model for the intention likelihood, capturing how well a state aligns with the intention of the teacher for the goal of the task. In Section 5.3, we discuss how we leverage the implicit action models proposed in Chapter 4 when computing plans to solve the task. Subsequently, we use these models in Section 5.4 to cast our problem as one of sequential decision making, allowing us to use Monte Carlo tree search by drawing similarities between our problem and Markov Decision Processes. Accordingly, we present our model for the tree structure enabling us to make decisions on three levels when planning: actions, action templates, and action goals. In Section 5.5, we detail our algorithm for iteratively growing such a tree to search for task solutions guided by the intention likelihood as a heuristic. Additionally, in Section 5.6, we present a method by which the robot can improve its noisy action models by practicing the task in simulation and without requiring additional teacher demonstrations. We present an extensive experimental evaluation of our approach on several tasks in Section 5.7. Finally, we discuss related work in Section 5.8 before concluding the chapter in Section 5.9.

5.1. Problem Formulation

We aim at learning a model for a task $\mathcal{T}$ from a small number N of teacher demonstrations. As introduced in Section 2.2, we define a task as a tuple $\mathcal{T} = \langle \mathcal{O}, \mathcal{A}, \Psi \rangle$, where $\mathcal{O}$ and $\mathcal{A}$ are respectively the objects and actions associated with the task, and Ψ is the function modeling its desirable goal states. In this work, we consider tasks whose goals can be modeled in terms of the spatial configuration of the objects.

We assume that each demonstration is segmented (either by the teacher or automatically using a suitable technique) into a set of segments, which we use to learn the set of actions $\mathcal{A} = \{a^{(1)}, \ldots, a^{(|\mathcal{A}|)}\}$ associated with the task. By observing how the state evolves after each segment, our approach learns a model of each demonstrated action, as well as a model Ψ that captures the intention of the teacher for the overall goal of

the task. The approach we present in this chapter enables the robot to use these models to solve the same task from new starting configurations, and to generalize the intention of the teacher without requiring an explicit goal to be specified in each case.

5.1.1. State Representation

In this chapter, we adopt the state representation we used in the case of learning actions (Section 4.1.1). Specifically, we construct the state $\mathbf{s}_t$ at time t based on the poses $\mathbf{T}_k(t)$ of all objects o_k (including the end-effector of the robot), which we express as homogeneous transforms. Additionally, our state model considers the joint configuration $\mathbf{o}_{ee}(t)$ of the end-effector at time t, e.g., the degree of opening of the gripper.

5.1.2. Teacher Demonstrations

To learn the task $\mathcal{T}$ and set of actions $\mathcal{A}$, our approach relies on a teacher providing the robot with N demonstrations $\mathcal{D}^{(\mathcal{T})} = \{\mathbf{d}^{(1)}, \ldots, \mathbf{d}^{(N)}\}$ of how to solve the task from different starting states. We rely on the same demonstration techniques as in Section 4.1.2, see Figure 4.2. We make the following assumptions about the demonstrations:

1. Complete observability: the robot can observe the state, i.e., the poses of all objects, at each time step of a demonstration.

2. Each demonstration starts from an arbitrary state $\mathbf{s}_{t_0}$ and ends with a desired goal state $\mathbf{s}_{T_n}$ that reflects the intention of the teacher for this task.

3. Each demonstration $\mathbf{d}^{(n)} \in \mathcal{D}^{(\mathcal{T})}$ is segmented into a sequence of $T_n + 1$ states that correspond to applying T_n actions by the teacher, i.e., $\mathbf{d}^{(n)} = (\mathbf{s}_{t_0}^{(n)}, \mathbf{s}_{t_1}^{(n)}, \ldots, \mathbf{s}_{T_n}^{(n)})$, where $\mathbf{s}_t^{(n)}$ denotes the state at time step t of the n-th demonstration. The segmentation is performed either by the teacher or by post-processing the demonstrations using an existing technique, see Section 5.3 below. Note that each demonstration can have a different number of steps, T_n.

Figure 5.2 shows examples of demonstrations for the task in Figure 5.1. The main idea of our method is to use the final state $\mathbf{s}_{T_n}$ of each demonstration to learn the overall goal of the task, see Section 5.2. Additionally, we use the demonstrated sequences of states to learn actions that help the robot reproduce the intended task goal from different starting states by generalizing the demonstrations, see Section 5.3.

5.1.3. The Problem

Our goal is to enable the robot to first use the teacher demonstrations to infer a model of the task and demonstrated actions, and then to use this model to solve the task from

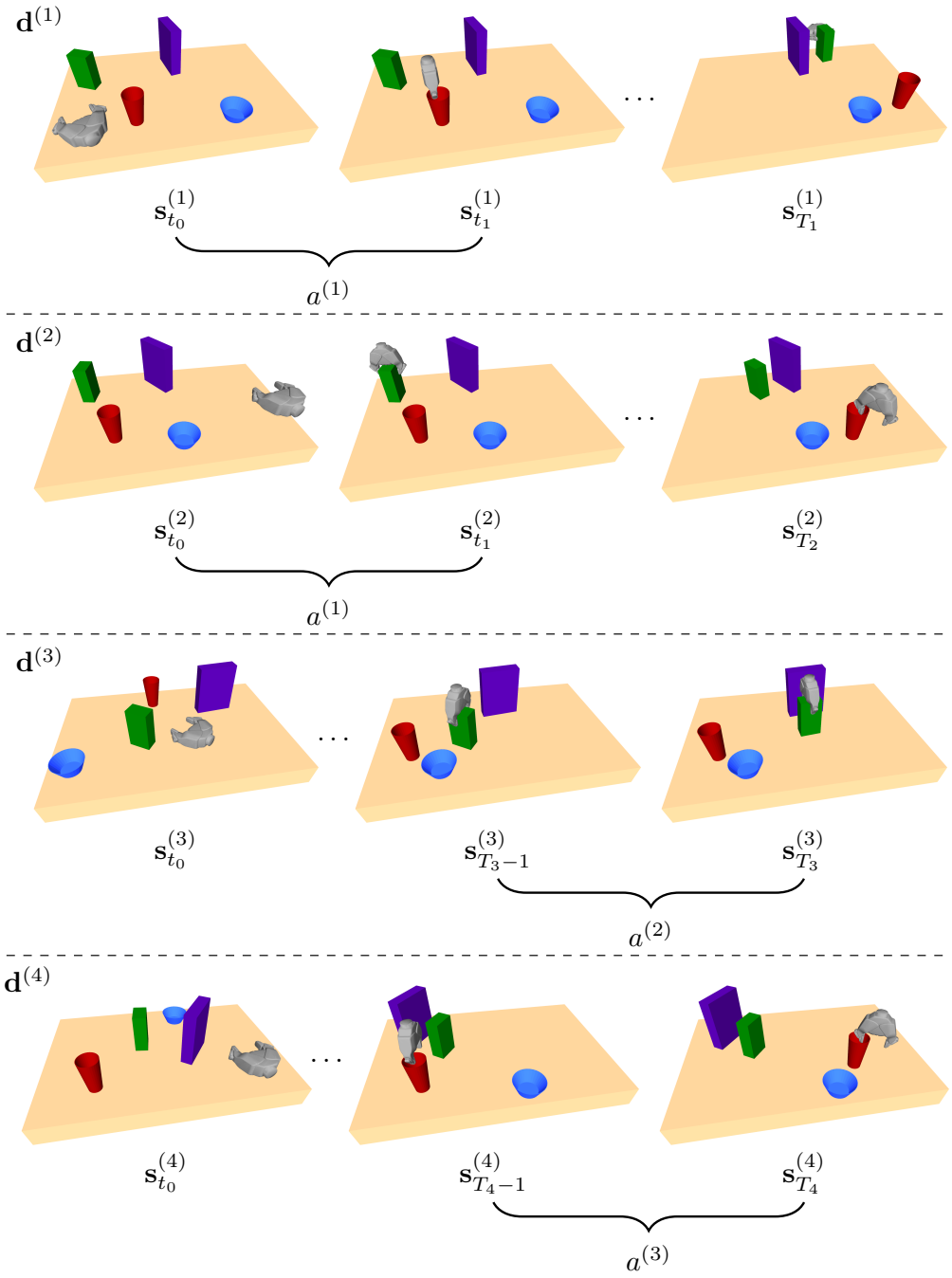


Figure 5.2.: Each row illustrates one demonstration for the task in Figure 5.1. We assume that the demonstrations are segmented such that each starts from an arbitrary state $\mathbf{s}_0^{(n)}$ and ends with a goal state $\mathbf{s}_{T_n}^{(n)}$. We use the final states $\mathbf{s}_{T_n}^{(n)}$ to learn a model for the overall goal of the task. Moreover, we use all segments (pairs of consecutive states) to learn action models that the robot uses when solving the task. We associate all segments in which the teacher moved the same set of objects with the same action, see Section 5.3.1. For example, $a^{(1)}$ represents the action of moving the end-effector (e.g., to reach for and grasp some object), whereas $a^{(2)}$ represents the action of moving the end-effector and the green box.

new starting states. This is a challenging problem as we rely only on a small number of demonstrations for learning, which results in ambiguity in the intention of the teacher. Specifically, we are interested in inferring which spatial relations between the objects are more relevant for each action and for the overall goal of the task, see Figure 5.1. This knowledge is crucial for the robot to be able to generalize the learned task and actions to new situations. Additionally, the robot should achieve this generalization without requiring the (non-expert) teacher to provide an explicit formulation of the desired goal.

We propose to formulate this as an optimization problem. Rather than requiring an explicit specification of the task goal, we model the *intention likelihood* Ψ of the teacher for the task, i.e., $\Psi(\mathbf{s})$ captures the degree to which a state $\mathbf{s}$ aligns with the goal states demonstrated by the teacher. Consequently, given an initial state $\mathbf{s}_0$, solving the task corresponds to computing a feasible *plan* $\Pi = \big(\langle a_0^*, \mathbf{s}_1^* \rangle, \ldots, \langle a_{T-1}^*, \mathbf{s}_T^* \rangle\big)$ consisting of a sequence $a_{0:T-1}^*$ of T actions and intermediate states $\mathbf{s}_{1:T}^*$ to achieve a desirable goal state $\mathbf{s}_T^*$ that maximizes the intention likelihood as much as possible.

Note that in general, we do not know the number of actions to apply in order to solve the task from $\mathbf{s}_0$. Therefore, the problem also entails computing the best number of steps T before achieving a desirable state $\mathbf{s}_T^*$. Moreover, we aim for a trade-off between the number of applied actions and the intention likelihood $\Psi(\mathbf{s}_T^*)$ of the final state achieved. We capture this formally by the following optimization problem:

$$\underset{\substack{T \in \mathbb{N}_{\geq 0} \\ a_{0:T-1} \in \mathcal{A} \\ \mathbf{s}_{1:T}}}{\text{maximize}} \quad \Psi(\mathbf{s}_T) - \sum_{t=0}^{T-1} \text{cost}(a_t), \tag{5.1}$$

$$\text{subject to} \quad p(\mathbf{s}_t \mid a_t) \geq \epsilon_s, \qquad \forall t, \tag{5.2}$$

$$\mathbf{s}_t \in \mathcal{S}_\text{f}, \qquad \forall t. \tag{5.3}$$

Here, a_t denotes the action (selected from $\mathcal{A}$) to apply at time step t, thereby transforming the state from $\mathbf{s}_t$ to $\mathbf{s}_{t+1}$. Furthermore, $\text{cost}(a_t)$ denotes the cost of applying an action, which is meant to bias the solution towards less expensive plans. In this work, we use a constant cost $c_0 > 0$ for all actions $a \in \mathcal{A}$.

Moreover, we impose two constraints on the computed plans. The first (Eq. (5.2)) is the *state affordability* (see Section 4.1.3), which models our desire to restrict the application of an action to starting states that are similar to those in which the teacher demonstrated the action. Accordingly, $p(\mathbf{s}_t \mid a_t)$ captures the likelihood that the teacher would have also selected state $\mathbf{s}_t$ to apply action a_t. The second constraint in Eq. (5.3) ensures that we compute (intermediate) goal states that are feasible, i.e., belong to the set of feasible states $\mathcal{S}_\text{f}$. Here, we check state feasibility as described in Section 2.2. This is important since the robot should not only reason about goal states with a high intention likelihood, but also consider how feasible those states are when executing the plan in practice. This formulation results in a flexible approach that allows the robot

to "improvise" task solutions by trading off the intention of the teacher with practical considerations.

5.1.4. Challenges

In this section, we briefly highlight the challenges involved in solving the problem in Eqs. (5.1)-(5.3). We summarize them as follows:

- **C1: The goal of the task is not explicitly specified.** We do not require the (non-expert) teacher to provide the goal of the task as a specific state or a formula. Instead, we infer a model Ψ of desirable task goals from the demonstrations, and use it to improvise goal states depending on the initial state $\mathbf{s}_0$.
- **C2: Action goals are ambiguous.** Due to the small number of demonstrations and the different order of applying the actions in the demonstrations, there is an inherent ambiguity in the intention of each learned action. Considering all demonstrated relations to be relevant for the action increases the dimensionality of the problem and limits the robot's ability to generalize to new situations.
- **C3: The stopping problem.** As the goal state is not explicitly specified, the optimal plan length T is unknown beforehand. Depending on the starting state $\mathbf{s}_0$, the robot may need to apply a different number of actions to achieve a desirable goal state.
- **C4: Non-convexity.** In general, the problem is highly non-convex due to the multi-modality of Ψ (Section 5.2) and action models $\mathcal{A}$ (Section 5.3). This makes it challenging to compute an optimal solution without a good initial guess.
- **C5: The hybrid nature of the problem.** Eqs. (5.1)-(5.3) is a mixed-integer optimization problem involving both the continuous state variables $\mathbf{s}_t$ and discrete variables (the solution length $T \in \mathbb{N}_{\geq 0}$ and action labels $a_t \in \{a^{(1)}, \ldots, a^{(|\mathcal{A}|)}\}$).
- **C6: Feasibility constraints.** The computed plan should correspond to physically feasible actions and states (e.g., collision free). This requires improvising goal states that adapt the intention of the teacher based on practical considerations.

We first address **C1** by presenting our model for the intention likelihood Ψ in Section 5.2, which captures multiple modes in desirable task goals based on a small number of demonstrations. We address **C2** by leveraging our discretized implicit action models presented in Chapter 4, which consider multiple interpretations for each action, see Section 5.3. Additionally, we tackle the stopping problem in **C3** by introducing a virtual *no-op* action that does not change the state when applied (Section 5.3.4). Given these

models, we revisit the problem in Eqs. (5.1)-(5.3) and reformulate it as one of sequential decision making (Section 5.4).

Finally, in Section 5.5, we propose a novel method based on Monte Carlo Tree Search to address challenges **C4-C6** by interleaving feasibility checks with exploring the space of actions and states. By doing so, we compute feasible plans that trade off the intention likelihood with the complexity of the solution based on the initial state $\mathbf{s}_0$.

5.2. Modeling the Intention Likelihood

In this section, we present our model for the intention likelihood Ψ, which captures how well a state aligns with the intention of the teacher for the goal of the task. In this work, we do not assume to have any semantic knowledge about the objects or the task. Instead, we assume that the intention of the teacher is to achieve certain pairwise spatial relations between the objects, which we use to model Ψ. We aim to learn these relations from a small number of demonstrations, and yet capture fine details in the intention of the teacher. Accordingly, we adopt the data-driven approach we introduced in Section 4.3.1, which models the relation between a pair of objects $p_{(k,l)} = \{o_k, o_l\}$ as a distribution based on their relative poses at the end of each demonstration. We use kernel density estimation to compute the likelihood of a pose ${}^l\mathbf{T}_k$ given the training poses as follows:

$$p\left({}^l\mathbf{T}_k\right) = \frac{1}{N}\sum_{n=1}^{N} \mathbf{k}\left({}^l\mathbf{T}_k^{(n)}(T_n),\, {}^l\mathbf{T}_k\right), \tag{5.4}$$

where ${}^l\mathbf{T}_k^{(n)}(T_n)$ is the pose of o_k relative to o_l at the end of the n-th demonstration, and $\mathbf{k}(\cdot,\cdot)$ is the Gaussian kernel as in Eq. (4.11).

This distribution captures multiple modes for a relation, as it can be seen as a mixture of Gaussians with each training example ${}^l\mathbf{T}_k^{(n)}(T_n)$ representing the center of one component. In this way, we model relations as continuous constraints defined over object poses, such that "satisfying" a relation translates to searching for regions of high density in its distribution. This corresponds to computing states $\mathbf{s}_T$ in which ${}^l\mathbf{T}_k(T)$ is close to those demonstrated by the teacher with respect to the distance function in Eq. (4.12). This is analogous to logical predicates (e.g., `NextTo(`o_k`,`o_l`)`) used in classical planning to define a goal formula for the task. Rather than using symbolic goals that are either satisfied (*true*) or not (*false*), our model provides a flexible interpretation of relations in a continuous space. In this way, we are able to learn arbitrary pairwise relations from the teacher without prior knowledge of such predicates or requiring classifiers that map the continuous state to the equivalent binary truth values.

We model Ψ based on the relation distributions for all pairs of objects. Typically, however, not all relations are equally relevant for the task, see Figure 5.3 for an example. Moreover, it is not always possible for the robot to satisfy all relations when solving

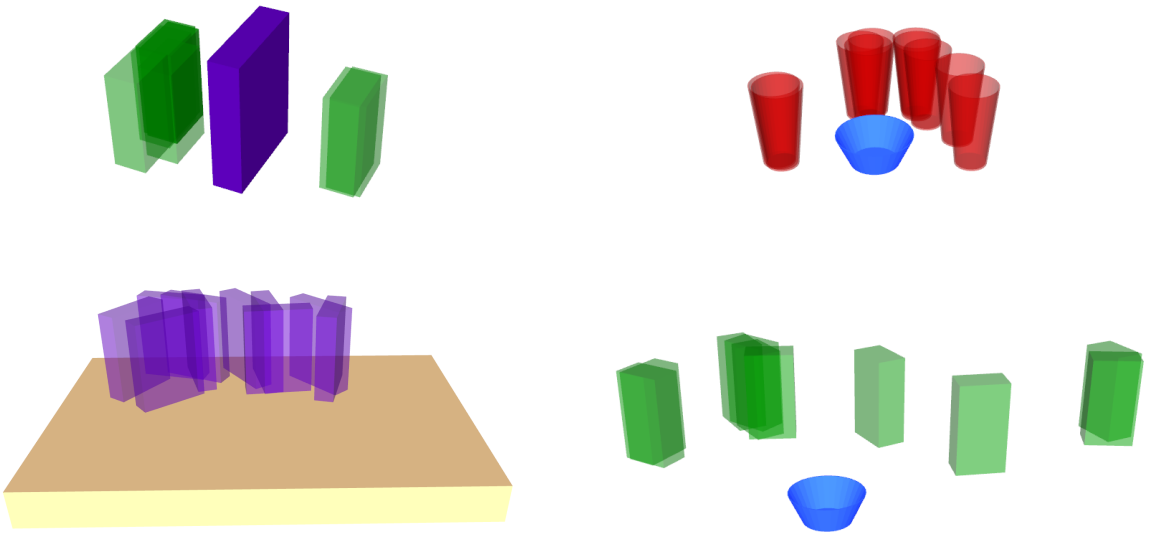


Figure 5.3.: We model the intention likelihood of the task based on the spatial relations between pairs of objects. We construct a non-parametric distribution for each relation based on the training examples (relative poses) demonstrated by the teacher. This figure shows four such relations for the task in Figure 5.1. We associate each relation with a weight that reflects its relevance (importance) for the task, which we estimate based on the entropy of the corresponding distributions. For example, the two relations in the upper row are more relevant for reproducing the task than those in the lower row since they exhibit a higher consistency across the demonstrations.

the task due to conflicting constraints in the corresponding object poses. Therefore, we assume the latent intention of the teacher with respect to the relevance of each relation to be captured by a function $\Omega : \mathcal{P} \rightarrow \mathbb{R}$, where $\mathcal{P}$ is the set of all object-pairs. We use $w_{(k,l)} = \Omega(p_{(k,l)})$ to denote the weight associated with pair $p_{(k,l)} = \{o_k, o_l\}$. Given such a function, we define the intention likelihood $\Psi(\mathbf{s}_T)$ of a state $\mathbf{s}_T$ as a weighted sum of the relation likelihoods over all pairs $\mathcal{P}(\mathbf{s}_T)$ in $\mathbf{s}_T$:

$$\begin{aligned} \Psi(\mathbf{s}_T) &= \eta \sum_{p_{(k,l)} \in \mathcal{P}(\mathbf{s}_T)} w_{(k,l)} \; p\left({}^{l}\mathbf{T}_k(T)\right) \\ &= \eta \sum_{o_k \in \mathcal{O}(\mathbf{s}_T)} \sum_{\substack{o_l \in \mathcal{O}(\mathbf{s}_T) \\ o_l \neq o_k}} w_{(k,l)} \; \frac{1}{N} \sum_{n=1}^{N} \mathbf{k}\left({}^{l}\mathbf{T}_k^{(n)}(T), {}^{l}\mathbf{T}_k(T)\right), \end{aligned} \tag{5.5}$$

where η is a normalizer such that $1/\eta$ is equal to the sum of all weights $w_{(k,l)}$. This formulation provides a flexible, multi-modal model of task goals. To maximize $\Psi(\mathbf{s}_T)$, we seek goal states $\mathbf{s}_T$ that correspond to peaks of the individual pairwise relation

distributions while considering their relative importance for the task. Moreover, by restricting the computation of $\Psi(\mathbf{s}_T)$ to consider only the objects $\mathcal{O}(\mathbf{s}_T)$ in state $\mathbf{s}_T$, we are flexible in interpreting the goal of the task based on the objects in the scene. For example, if a subset of the objects are missing, we adapt by imitating as many aspects of the demonstrations as possible.

In general, we do not assume prior knowledge about Ω. Therefore, we propose to estimate the weights $w_{(k,l)}$ to reflect the consistency in the demonstrations for the corresponding relation. We assume that relations that are more relevant for the task are associated with a lower dispersion in the demonstrated poses compared to less relevant ones. We use the differential entropy $H_{(k,l)}$ of the distribution in Eq. (5.4) as a measure of this dispersion, which we compute numerically as follows:

$$H_{(k,l)} = -\frac{1}{N_H} \sum_{h=1}^{N_H} \log \left(p({}^{l}\mathbf{T}_k^{(h)}) \right), \tag{5.6}$$

where ${}^{l}\mathbf{T}_k^{(h)}$ are samples we draw from the N modes (Gaussians) in Eq. (5.4), and N_H is the total number of samples. This is an estimate of the Shannon entropy for continuous distributions that are modeled using a non-parametric, kernel density estimate, as proposed by Ahmad and Lin (1976). Moreover, we assume that the pose of the end-effector o_{ee} is irrelevant for $\Psi(\mathbf{s}_T)$, i.e., the goal only depends on the objects that the robot can manipulate. Accordingly, we estimate relation weights as follows:

$$w_{(k,l)} = \begin{cases} 0 & \text{if } o_k = o_{ee} \text{ or } o_l = o_{ee} \\ \frac{1}{\epsilon_H + H_{(k,l)}} & \text{otherwise,} \end{cases} \tag{5.7}$$

where $\epsilon_H = 0.01 - \min(0, H_{\min})$ ensures that all weights $w_{(k,l)}$ are finite and positive, and $H_{\min}$ is the minimum entropy over all pairwise distributions.

Finally, we note that this model estimates the intention of the teacher given a small number of demonstrations and without prior knowledge of the task. However, the method we present in this chapter is generic with respect to the intention likelihood model used. For example, given additional semantic knowledge about the objects, one could infer relation weights $w_{(k,l)}$ according to the predicted preferences of the teacher as in Chapter 3. Similarly, one could leverage previous knowledge about typical spatial relations when modeling Eq. (5.4). This, however, is outside the scope of this chapter.

5.3. Modeling Actions

To solve the task $\mathcal{T}$ from a starting state $\mathbf{s}_0$, we seek a sequence of actions $a^*_{0:T-1}$, along with the corresponding intermediate states $\mathbf{s}^*_{1:T}$, see Section 5.1.3. Whereas the teacher demonstrations enable us to learn a model Ψ of desirable task goal states, they also

encode useful "moves" or actions that the robot can use to achieve these states. In this section, we describe how we learn a model of each action from the demonstrations such that the robot can leverage this knowledge when solving the task in new situations.

The main challenge in learning an action model from a small number of demonstrations is the inherent ambiguity with respect to the intention of the teacher. We assume to have no prior or expert knowledge about which features or spatial relations between the objects are more relevant for each action. Therefore, we rely on the approach we described in Section 4.3, which formulates an implicit model of the action goals based on a mixture of possible interpretations of the demonstrations. This enables us to reason about multiple modes in the intention of the teacher for each action.

However, this multi-modality also means that the corresponding distributions are characterized by local minima. This makes it challenging to solve the optimization problem in Eq. (5.1) without a good initial guess. Therefore, we adopt the discretized goal distributions we introduced in Section 4.3.4. This makes it feasible to leverage state-of-the-art decision-making techniques such as Monte Carlo Tree Search (MCTS) to sample useful intermediate goal states in order to compute a sequence of actions that solves the problem.

We first describe how we divide the task demonstrations into segments that correspond to each of the actions to learn. We then briefly recap our implicit model of action goals presented in Section 4.3 and discuss how we use it in this chapter to solve a task. Finally, we present an approximation that guarantees a lower bound on the state affordability in Eq. (4.14), allowing us to efficiently check the constraint in Eq. (5.2) and determine the set of applicable actions at each time step.

5.3.1. Action Demonstrations

As discussed in Section 5.1.2, we assume that each task demonstration $\mathbf{d}^{(n)} \in \mathcal{D}^{(\mathcal{T})}$ is segmented into a sequence of states $\mathbf{d}^{(n)} = (\mathbf{s}_{t_0}^{(n)}, \mathbf{s}_{t_1}^{(n)}, \ldots, \mathbf{s}_{T_n}^{(n)})$, either by the teacher or using an existing segmentation technique (Niekum et al., 2012; Meier et al., 2012; Ureche et al., 2015). We consider point-to-point actions that can be modeled in terms of the poses of the objects before and after applying an action. Our approach does not require the teacher to manually label each segment with a specific action name (e.g., "move-cup-to-plate"). Instead, we adopt a heuristic that specifies the endpoints of segments and labels them based on which objects were perceived to move during each part of the demonstrations. We adopt the action models introduced in Section 4.3 and assume that the intention of each action is to move a subset $\mathcal{O}_M^a$ of the objects to desired goal poses relative to the remaining, stationary objects $\mathcal{O}_S^a$.

Based on this, we assign a unique label to all segments $(\mathbf{s}_t^{(n)}, \mathbf{s}_{t+1}^{(n)})$ that share the same set of moving objects, $\mathcal{O}_M^a$, thus forming the set of actions $\mathcal{A}$, see Figure 5.2. Finally, we collect all segments with the same label a to construct a set of demonstrations

$\mathcal{D}^{(a)}$ for learning the action, where $\mathcal{D}^{(a)} = \{(\mathbf{s}_{t_s}, \mathbf{s}_{t_e})^{(1)}, \ldots, (\mathbf{s}_{t_s}, \mathbf{s}_{t_e})^{(N_a)}\}$, N_a is the number of action demonstrations, and t_s and t_e are the start and end time of a segment, respectively.

5.3.2. Modeling Action Goals

Given the demonstrations $\mathcal{D}^{(a)}$ of an action a, we seek a model $p(\mathbf{s}_{t+1} \mid \mathbf{s}_t, a)$ describing the likelihood of state $\mathbf{s}_{t+1}$ being an intended goal of the action when applying it in state $\mathbf{s}_t$. We adopt a discretized model of this distribution as described in Section 4.3.4, as this allows us to efficiently sample likely goal states when solving the task. Specifically, given $\mathbf{s}_t$, we seek a finite set of goal states $\hat{\mathcal{S}}_{t+1} = \{\hat{\mathbf{s}}_{t+1}^{(1)}, \ldots, \hat{\mathbf{s}}_{t+1}^{(S)}\}$, and their probabilities, $P(\mathbf{s}_{t+1} = \hat{\mathbf{s}}_{t+1}^{(s)} \mid \mathbf{s}_t, a)$ for $s = 1, \ldots, S$. We briefly recap this model before describing how we use it in the context of solving a task.

5.3.2.1. Action Templates

We address the ambiguity related to an action a using a finite set of action templates Γ^a. Each template $\gamma_s \in \Gamma^a$ represents one interpretation of the action based on a specific stationary object $o_s \in \mathcal{O}_S^a$. We combine these interpretations in one goal distribution for the action $p(\mathbf{s}_{t+1} \mid \mathbf{s}_t, a)$ given a starting state $\mathbf{s}_t$, i.e.,

$$p(\mathbf{s}_{t+1} \mid \mathbf{s}_t, a) = \sum_{\gamma_s \in \Gamma^a} p(\mathbf{s}_{t+1} \mid \mathbf{s}_t, a, \gamma_s)\, p(\gamma_s \mid \mathbf{s}_t, a), \tag{5.8}$$

where $p(\gamma_s \mid \mathbf{s}_t, a)$ is the *template-relevance* of γ_s, which captures the importance of o_s for the action. Note that an object o_s does not contribute to the goal distribution if it is not in the scene, i.e., $p(\gamma_s \mid \mathbf{s}_t, a) = 0$ if $o_s \notin \mathcal{O}_S^a(\mathbf{s}_t)$. Otherwise, we assume equal relevance for all templates, i.e., $p(\gamma_s \mid \mathbf{s}_t, a) = 1/|\mathcal{O}_S^a|$, see Section 4.3.3.2 for details.

5.3.2.2. Pairwise Voting Distributions

Given a template γ_s, the idea of our model is to have the corresponding object o_s vote for the likely goal poses of the moving objects. We associate each stationary object $o_s \in \mathcal{O}_S^a$ with a discrete voting distribution $\hat{p}_s$ over poses ${}^s\mathbf{T}_*$ of the representative moving object $o_* \in \mathcal{O}_M^a$ relative to o_s. These votes can take one of $\hat{N}_s$ values. We learn these votes ${}^s\hat{V}_* = \{{}^s\hat{\mathbf{T}}_*^{(1)}, \ldots, {}^s\hat{\mathbf{T}}_*^{(\hat{N}_s)}\}$ by clustering the set of poses ${}^sV_* = \{{}^s\mathbf{T}_*^{(1)}(t_e), \ldots, {}^s\mathbf{T}_*^{(N_s)}(t_e)\}$ of o_* relative to o_s as observed at the end of each action demonstration. We compute the probability $\hat{w}_s^{(n)}$ of sampling the n-th vote ${}^s\hat{\mathbf{T}}_*^{(n)}$ based on the density of demonstrated poses in the corresponding cluster, see Eq. (4.20).

5.3.2.3. Discrete Goal States

To construct the distribution of goal states, we draw I samples ${}^{s}\widetilde{\mathbf{T}}_{*}^{(i)}$ from each voting distribution $\hat{p}_s$. We transform each sample to the world frame based on the pose $\mathbf{T}_s(t)$ of the voting object o_s, i.e., $\widetilde{\mathbf{T}}_{*}^{(i)} = \mathbf{T}_s(t)\,{}^{s}\widetilde{\mathbf{T}}_{*}^{(i)}$. Each transformed vote $\widetilde{\mathbf{T}}_{*}^{(i)}$ defines a potential goal state $\widetilde{\mathbf{s}}_{t+1}^{(i)}$ by assuming that all moving objects other than o_* maintain their poses relative to o_* when applying the action. We use these states resulting from the votes of o_s to construct the distribution $p(\mathbf{s}_{t+1} \mid \mathbf{s}_t, a, \gamma_s)$ of goals given γ_s, see Eq. (4.21).

Accordingly, we combine the potential goal states $\widetilde{\mathcal{S}}_{t+1} = \{\widetilde{\mathbf{s}}_{t+1}^{(1)}, \ldots, \widetilde{\mathbf{s}}_{t+1}^{(j)}\}$ resulting from all stationary objects in the scene to construct the overall goal distribution $p(\mathbf{s}_{t+1} \mid \mathbf{s}_t, a)$, see Eq. (4.22). This model takes into account the weight $\widetilde{w}_{\gamma_s}^{(j)}$ of each state $\widetilde{\mathbf{s}}_{t+1}^{(j)} \in \widetilde{\mathcal{S}}_{t+1}$, which is equal to the relevance $p(\gamma_s \mid \mathbf{s}_t, a)$ of the template γ_s responsible for the corresponding vote.

Finally, we discretize this distribution by clustering the set of states $\widetilde{\mathcal{S}}_{t+1}$ using agglomerative hierarchical clustering with a distance threshold of $\delta_{\max}$, see Section 4.3.4.2. By doing so, we identify the regions in state space with high goal density. The result is a set of goal states $\hat{\mathcal{S}}_{t+1} = \{\hat{\mathbf{s}}_{t+1}^{(1)}, \ldots, \hat{\mathbf{s}}_{t+1}^{(S)}\}$, where each $\hat{\mathbf{s}}_{t+1}^{(s)}$ is the mean state of a cluster. We compute the probability $P(\mathbf{s}_{t+1} = \hat{\mathbf{s}}_{t+1}^{(s)} \mid \mathbf{s}_t, a)$ of each state based on the density of the (weighted) votes that belong to it, see Eq. (4.23).

5.3.2.4. Sampling Action Goals

In Chapter 4, we used the above model to select the best goal state $\mathbf{s}_{t+1}^{*} \in \hat{\mathcal{S}}_{t+1}$ when applying an action. For this, we chose the feasible state with the highest probability, as this represents the best way to apply the action given the starting state, see Section 4.3.4.3. In this chapter, we are not only interested in the best way to apply an action, but in the best sequence of actions leading to a goal state $\mathbf{s}_T^{*}$ of the whole task.

In this context, we identify two challenges that were not relevant when learning an action individually in Chapter 4. Firstly, the teacher may demonstrate how to solve the task by applying the actions in different orders. This results in more ambiguity (noise) in the learned action models compared to when learning an action individually. For example the relevance of a stationary object o_s when applying an action may vary based on whether the teacher had already moved o_s (using another action) to a desirable location. Secondly, when applying an action in state $\mathbf{s}_t$ to reach $\mathbf{s}_{t+1}$, we should consider which other actions can be subsequently applied in $\mathbf{s}_{t+1}$ based on the state affordability constraint in Eq. (5.2).

In this chapter, we tackle these issues as follows:

1. When computing a plan to solve the task, we reduce the dimensionality of the problem by considering one template (i.e., stationary object) at a time for applying

an action. This allows us to select the best way of applying the action at a specific step of the plan without committing to a particular template beforehand, see Sections 5.4.4 and 5.5.2.

2. Rather than selecting a specific goal state when applying an action, our approach explores goals leading to different solutions by sampling them according to their probabilities, see Section 5.5.1.3.

3. We present a method that enables the robot to use its experience from solving the task to *relearn* both the template relevance and the voting distribution of each stationary object. In this way, the robot overcomes the ambiguity in the demonstrations by practicing the task in simulation, and without requiring additional teacher demonstrations, see Section 5.6.

5.3.3. A Lower Bound on the State Affordability

Besides learning a model of action goals, we use the demonstrations $\mathcal{D}^{(a)}$ of an action a to learn a model of the state affordability $p(\mathbf{s}_t \mid a)$. For this, we use the approach we presented in Section 4.3.2, which models the affordability based on the poses of the moving objects in $\mathbf{s}_t$. Accordingly, the constraint $p(\mathbf{s}_t \mid a_t) \geq \epsilon_s$ in Eq. (5.2) ensures that we only select an action a_t to apply at time step t if state $\mathbf{s}_t$ sufficiently resembles the starting states of the action demonstrations.

However, specifying the threshold ϵ_s can be challenging as this depends on the number of training examples N_a and the bandwidth σ of the Gaussian kernel, see Eq. (4.14). Therefore, in this section, we present an approximation of this constraint that allows us to efficiently check the state affordability in a way that considers the discretization of action goal states discussed in Section 5.3.2 above.

Since we define the state affordability as the product of independent distributions over the pose of the representative object $o_* \in \mathcal{O}_M^a$ relative to all other moving objects $o_k \in \mathcal{O}_M^a$ in $\mathbf{s}_t$, we express the constraint $p(\mathbf{s}_t \mid a_t) \geq \epsilon_s$ using an equivalent one specifying a threshold ϵ_s' for the likelihood with respect to each pair of objects, i.e.,

$$\begin{aligned} p(\mathbf{s}_t \mid a) = \prod_{\substack{o_k \in \mathcal{O}_M^a \\ o_k \neq o_*}} \sum_{i=1}^{N_a} \frac{1}{N_a} \mathbf{k}\left({}^k\mathbf{T}_*^{(i)}(t_s),\, {}^k\mathbf{T}_*(t)\right) \geq \quad \epsilon_s \\ \Rightarrow \sum_{i=1}^{N_a} \frac{1}{N_a} \mathbf{k}\left({}^k\mathbf{T}_*^{(i)}(t_s),\, {}^k\mathbf{T}_*(t)\right) \geq \quad \epsilon_s', \quad \forall_{\substack{o_k \in \mathcal{O}_M^a \\ o_k \neq o_*}}. \end{aligned} \tag{5.9}$$

Moreover, considering that

$$\sum_{i=1}^{N_a} \frac{1}{N_a} \mathbf{k}\left(\cdot, \cdot\right) > \max_i \left(\frac{1}{N_a} \mathbf{k}\left(\cdot, \cdot\right) \right), \tag{5.10}$$

we satisfy Eq. (5.9) if we guarantee that

$$\max_i \left(\frac{1}{N_a} \mathbf{k} \left({}^k\mathbf{T}_*^{(i)}(t_s),\, {}^k\mathbf{T}_*(t) \right) \right) \geq \epsilon'_s, \;\; \forall_{\substack{o_k \in \mathcal{O}_M^a \\ o_k \neq o_*}}. \tag{5.11}$$

By taking the logarithm and considering that it is a monotonically increasing function, we formulate Eq. (5.11) in terms of a distance threshold between relative poses as follows

$$\begin{aligned} &\max_i \left(\log \left(\frac{1}{N_a} \mathbf{k} \left({}^k\mathbf{T}_*^{(i)}(t_s),\, {}^k\mathbf{T}_*(t) \right) \right) \right) \geq \log(\epsilon'_s), && \forall_{\substack{o_k \in \mathcal{O}_M^a \\ o_k \neq o_*}} \\ &\Rightarrow \max_i \left(- \operatorname{dist} \left({}^k\mathbf{T}_*^{(i)}(t_s),\, {}^k\mathbf{T}_*(t) \right)^2 \right) \geq \tilde{\epsilon_s}, && \forall_{\substack{o_k \in \mathcal{O}_M^a \\ o_k \neq o_*}} \\ &\Rightarrow \min_i \left(\operatorname{dist} \left({}^k\mathbf{T}_*^{(i)}(t_s),\, {}^k\mathbf{T}_*(t) \right) \right) \leq d_{\max}, && \forall_{\substack{o_k \in \mathcal{O}_M^a \\ o_k \neq o_*}}. \end{aligned} \tag{5.12}$$

In other words, we check the state affordability constraint for a state $\mathbf{s}_t$ by requiring that the minimum distance between the relative pose ${}^k\mathbf{T}_*(t)$ in the state and the corresponding poses ${}^k\mathbf{T}_*^{(i)}(t_s)$ demonstrated by the teacher is at most $d_{\max}$ for all objects $o_k \in \mathcal{O}_M^a, o_k \neq o_*$. In this work, we set $d_{\max}$ to the clustering threshold $\delta_{\max}$ we use when discretizing the goal states of the action, see Section 5.3.2.

Finally, note that this model is based on pairs of moving objects, and is therefore not defined in the special case of $\mathcal{O}_M^a$ consisting of the end-effector object only. In that case, we consider the affordability constraint to be satisfied by default, i.e., $p(\mathbf{s}_t \mid a) = \epsilon_s$.

5.3.4. The Special No-Op Action

One of the challenges in solving a task is that the number of steps T, and thus the number of optimization variables, is unknown beforehand (see **C3** in Section 5.1.4). As we are not given a specific goal state to reach, it is generally unclear if the robot can achieve a better goal state by exploring longer sequences of actions, e.g., by continuing to re-arrange the objects in the scene.

This is closely related to the optimal stopping problem in decision theory (Dynkin, 1963; Keller and Geißer, 2015), and has been extensively addressed in the context of Markov Decision Processes (MDPs). Different approaches have proposed a number of strategies for an agent to determine the best time to stop exploring new states that could lead to higher rewards and to return the current solution instead.

Analogously, we introduce a special (virtual) *no-op* action $a^{(0)}$, which does not change the state when applied. This action allows us to reason about reaching a final goal state $\mathbf{s}_T^*$ with which we are satisfied, as this is equivalent to repeatedly applying the $a^{(0)}$ action from that time step on. By design, we assume this action to be applicable from any state, i.e., the state affordability constraint (Eq. (5.2)) is satisfied by default. Moreover, this action considers one (dummy) template $\gamma^{(0)}$ and is deterministic with respect to its goal

state distribution, i.e., $P(\mathbf{s}_{t+1} = \mathbf{s}_t \mid \mathbf{s}_t, a^{(0)}, \gamma^{(0)}) = 1$, and $P(\mathbf{s}_{t+1} \neq \mathbf{s}_t \mid \mathbf{s}_t, a^{(0)}, \gamma^{(0)}) = 0$. Finally, applying this action incurs no cost for the objective function in Eq. (5.1), i.e., $\text{cost}(a^{(0)}) = 0$. In Section 5.4, we show how we use Monte Carlo Tree Search, along with this action, to address the stopping problem.

5.4. Search-Based Optimization

In this section, we first recap the models presented so far and how they address some of the challenges highlighted in Section 5.1.4. We then revisit the optimization problem in Section 5.1.3 and reformulate it to incorporate the action models in Section 5.3. Accordingly, we show how we leverage Monte Carlo Tree Search, a powerful sequential decision-making paradigm, to address the remaining challenges and solve the problem.

5.4.1. The Optimization Problem Revisited

Thus far, we have addressed the first challenge **C1** in our problem by proposing a model for the task intention likelihood $\Psi(\mathbf{s}_T)$ (Section 5.2), which captures desirable goal states of the task based on a small number of teacher demonstrations. Moreover, we presented our approach for modeling action goals $p(\mathbf{s}_{t+1} \mid \mathbf{s}_t, a)$ in terms of discrete distributions over the goal state $\mathbf{s}_{t+1}$ conditioned on the starting state $\mathbf{s}_t$, see Section 5.3.2.3. We construct $p(\mathbf{s}_{t+1} \mid \mathbf{s}_t, a)$ by combining different ways of applying a based on its templates Γ^a. As discussed in **C2** and Section 5.3.2.3, these distributions are generally noisy due to the ambiguity in the demonstrations. Our method alleviates this issue by reducing the dimensionality of the problem and using each template $\gamma \in \Gamma^a$ to compute a goal distribution $p(\mathbf{s}_{t+1} \mid \mathbf{s}_t, a, \gamma)$ by considering one stationary object o_s in the scene at a time. We now incorporate this model in the optimization problem in Eq. (5.1) by considering additional variables γ_t that determine the best interpretation of action a_t at time t. Accordingly, we reformulate the constrained optimization in Eqs. (5.1)-(5.3) as follows:

$$\underset{\substack{T \in \mathbb{N}_{\geq 0} \\ a_{0:T-1} \in \mathcal{A}' \\ \gamma_{0:T-1} \\ \mathbf{s}_{1:T}}}{\text{maximize}} \quad \Psi(\mathbf{s}_T) - \sum_{t=0}^{T-1} \text{cost}(a_t), \tag{5.13}$$

$$\text{subject to} \quad p(\mathbf{s}_t \mid a_t) \geq \epsilon_s, \qquad \forall t, \tag{5.14}$$

$$\mathbf{s}_t \in \mathcal{S}_\text{f}, \qquad \forall t, \tag{5.15}$$

where each γ_t variable can take a value from the set of templates Γ^{a_t} of the corresponding action a_t at each step of the solution. Finally, $\mathcal{A}' = \mathcal{A} \cup \{a^{(0)}\}$ denotes the set of actions consisting of both the actions $\mathcal{A}$ learned from the teacher demonstrations and the special

no-op action $a^{(0)}$ (Section 5.3.4). Our approach uses this action to decide when a final goal state $\mathbf{s}_T$ has been reached, thereby addressing the stopping problem in **C3**.

5.4.2. Relation to Markov Decision Processes

We note that our problem resembles that of computing an optimal policy in the context of MDPs (Bellman, 1957a,b; Sutton and Barto, 1998). There, the goal is to compute a sequence of actions to maximize the expected reward of an agent equipped with a library of deterministic or stochastic actions defined by their transition functions. By applying actions and interacting with the environment, the agent observes the outcomes and receives corresponding rewards. It then uses this information to guide its behavior until a certain horizon is met or a terminal state is reached. Analogously, we aim to compute a sequence of actions that maximizes the intention likelihood of the final state. Our action goal distributions correspond to the transition functions in an MDP. However, it is important to note that the stochasticity in our case is not the result of non-deterministic behavior when interacting with the environment as in the case of MDPs. Rather, it stems from the ambiguity with respect to the intention of the teacher for the action. In other words $P(\mathbf{s}_{t+1} = \hat{\mathbf{s}}^{(s)}_{t+1} \mid \mathbf{s}_t, a)$ reflects the probability that the teacher intended $\hat{\mathbf{s}}^{(s)}_{t+1}$ to be a goal of the action when applied from $\mathbf{s}_t$. After selecting $\hat{\mathbf{s}}^{(s)}_{t+1}$ as the action goal, we assume that the robot is able to achieve this state deterministically, e.g., using a motion planner.

Similarly, our problem shares aspects with other decision-theoretic domains such as games and multi-armed bandit problems (Auer et al., 2002; Gelly and Silver, 2011). There, an agent computes a sequence of moves or actions to maximize its (expected) reward or its chances of winning a game, for example by trading off exploiting known strategies with exploring new ones. This is analogous to the robot reasoning about different sequences and interpretations of the learned actions in order to achieve desirable goal states for the task.

Based on this insight, we propose to view our problem from a decision-theoretic perspective, and present an approach based on Monte Carlo Tree Search (MCTS) for solving it (Coulom, 2006). Recently, MCTS has been shown to be a powerful tool for addressing a large spectrum of problems including MDPs, probabilistic planning, games, and combined task and motion planning (Kocsis and Szepesvári, 2006; Keller and Eyerich, 2012; Toussaint, 2015; Silver et al., 2016). The main idea of MCTS is to iteratively grow a search tree to explore applying different sequences of actions while using the resulting "scores" to inform the search.

Our idea is to use this search-based approach to maximize the objective function in Eq. (5.13). By using the intention likelihood as a heuristic, we guide the search to favor sequences of actions leading to more promising goal states. Moreover, by incorporating randomness in our strategies of selecting promising tree paths to expand, we trade

this off with exploring less promising solutions. This alleviates the non-convexity of the problem (**C4**) by providing an informed method for navigating the large space of actions and states while considering different locally-optimal solutions of the task. Additionally, by preventing states that result from applying the no-op action $a^{(0)}$ from being expanded further, we tackle the stopping problem (**C3**) while still considering solutions of different lengths T during the search.

Furthermore, by using search nodes of different types, MCTS allows us to perform decision making on multiple levels, thus answering the following questions: *Which action should we apply next? Given the action, which template should we use for generating goals? Which goal state should we select as the result of applying the action?* This allows us to maximize our objective function with respect to both continuous and discrete variables, i.e., addressing **C5**.

Finally, by continuously checking the affordability (Eq. (5.14)) and feasibility (Eq. (5.15)) of states resulting from different search paths, we interleave optimizing the objective function with satisfying those constraints, thereby addressing the final challenge **C6**.

In the next section, we briefly summarize the steps of the basic MCTS algorithm. We then present our version of MCTS, which involves decision making on three levels: actions, templates, and goal states.

5.4.3. Monte Carlo Tree Search

There exist several variations of the Monte Carlo Tree Search algorithm. In this section, we present the common steps of the basic version of MCTS, before presenting our algorithm in Section 5.5. For an overview of different MCTS-based algorithms and applications, we refer the reader to the survey by Browne et al. (2012).

The MCTS algorithm iteratively grows a search tree to approximate the values (returns) of states and actions using Monte Carlo simulations. A root node is first created using the initial state. Each iteration grows the tree by adding one or more children to one promising node as a result of applying actions from its corresponding state. This goes on until a specified budget of iterations has been used up. Each iteration consists of four steps, which we summarize bellow.

Selection The aim of this step is to find a node from which the tree can be grown further. This is done by traversing the tree, starting from the root, until the most promising, expandable node is reached. This is typically done using a so-called tree policy for deciding which child to select next from any given node. For example, a greedy policy would always select the child with the largest value, whereas other policies trade off exploration with exploitation based on children statistics such as their values or visit counts.

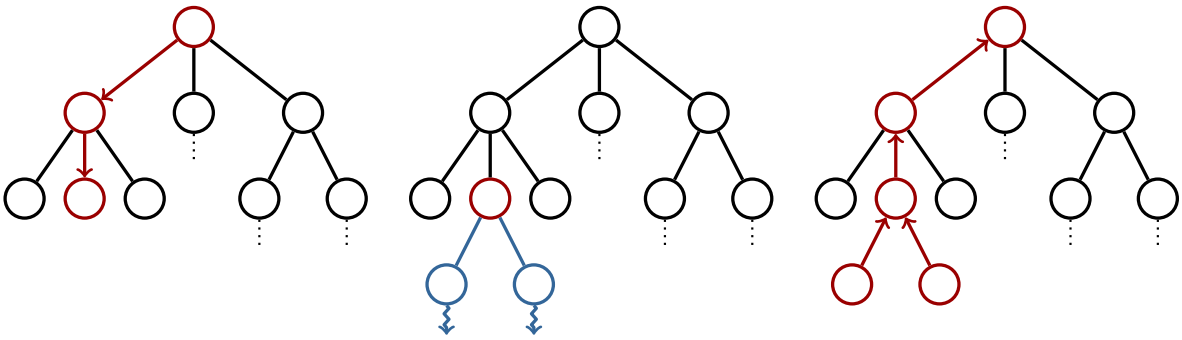


Figure 5.4.: The main steps of Monte Carlo tree search. Left: during the node selection stage, we traverse the tree, guided by the node values, in order to select a promising node to expand next (red path). Middle: we expand the selected node by applying actions in its state. This results in adding new successor nodes to the tree (blue). To estimate the values of the new nodes, we either simulate rollouts from their states using a default policy (blue arrows), or rely on a heuristic to initialize those values. Right: we propagate the new values back to the root node and update the values of all nodes on that path (red). This informs the node selection in the next iteration.

Expansion The selected node is then expanded by applying one or more actions, resulting in adding one or more nodes as children of this node.

Simulation To evaluate each new node, a simulation step is performed to compute the outcomes (e.g., rewards) as a result of performing the corresponding actions. This is typically done using a so-called default policy that simulates the domain and estimates the received values or rewards.

Backpropagation The result of the simulation step is used to update the statistics of the nodes on the path from the new nodes back to the root node. This typically involves updating the value estimates and visit counts of the nodes.

We illustrate these steps in Figure 5.4. Note that MCTS is an anytime algorithm that updates the value estimates after each iteration. We therefore specify the budget of MCTS iterations to trade-off the time available for solving the task with the quality of the found solution. After stopping the algorithm, we use the latest estimates to compute the next best action (or sequence of actions) to perform from the initial state. A typical strategy to do so is to traverse the tree from the root and greedily select actions leading to goals with the largest values.

5.4.4. Tree Structure

Our approach builds on the MCTS algorithm above. Starting with the initial state $\mathbf{s}_0$ as a root node, we construct a search tree by applying actions, modeled through different templates, to explore the different resulting goal states. In this section, we present the tree structure that we propose to accomplish this.

Typically, MCTS distinguishes two types of nodes: decision (or choice) nodes, and chance nodes. These nodes alternate such that decision nodes have chance nodes as children and vice versa. Decision nodes are responsible for selecting which action to apply from the state associated with the node. For each action, a child consisting of a chance node is used to reason about the potential outcomes of the action. This reflects the stochasticity of the environment typically associated with MDPs or games. Each state resulting from applying the action is then associated with a decision node successor, from which we can apply new actions again.

Analogously, we also incorporate nodes to decide which actions to apply from a given state. However, we assume that the only source of stochasticity when applying an action is the ambiguity in the intended model of the action, and is not due to unexpected events such as an object slipping from the end-effector. As we model actions using different templates, we propose a tree structure that alternates between three types of nodes: action-selection, template-selection, and goal-selection nodes. By incorporating template-selection nodes, we reduce the dimensionality of the problem when applying an action by considering one action interpretation at a time. This allows us to tackle the ambiguity in the teacher demonstrations and to consider several ways of applying the same action based on the initial state.

Action-Selection Nodes Action-selection nodes model the decision of selecting an action to apply from a given state. We define an action-selection node n_a as a tuple $n_a = \langle \mathbf{s}, P, V \rangle$, where $\mathbf{s}$ is the state associated with the node, and $V(\mathbf{s}) \in \mathbb{R}$ is the state-value estimate, which is analogous to the value of a state in the context of MDPs. In our context, $V(\mathbf{s})$ approximates the best objective function value (Eq. (5.13)) that can be achieved when solving the task starting from $\mathbf{s}$ as an initial state. With the exception of the root node, action-selection nodes are added to the tree as a result of applying actions using their parent nodes. Therefore, P is the probability of the state $\mathbf{s}$ of n_a being the intended goal of an action, i.e., $P(\mathbf{s}_{t+1} = \mathbf{s} \mid \mathbf{s}_t, a, \gamma)$, where $\mathbf{s}_t$, a, and γ, are the state, action, and action template of the parent of n_a, respectively. After adding n_a to the tree, new nodes can in turn be added by applying actions from its state $\mathbf{s}$. Each action choice is represented by an edge connecting n_a with a template-selection child node, defined next.

Template-Selection Nodes We use template-selection nodes to model the selection of a template γ to interpret an action a when applying it from state $\mathbf{s}$. We define a template-selection node n_γ as a tuple $n_\gamma = \langle \mathbf{s}, a, Q \rangle$. Here, a is the action associated with the node, which is the action of the edge between it and the parent action-selection node. Moreover, the state $\mathbf{s}$ of the node is the state of the parent action-selection node. The function $Q(\mathbf{s}, a) \in \mathbb{R}$ is the corresponding state-action value, which reflects how beneficial it is to apply action a in state $\mathbf{s}$ when solving the task. To model the choice between different interpretations of a, we use an edge representing each template $\gamma \in \Gamma^a$ to connect n_γ to a so-called goal-selection child node, which we define next.

Goal-Selection Nodes Given a state $\mathbf{s}$, an action a to apply from it, and a template $\gamma \in \Gamma^a$ for interpreting the action, a goal-selection node n_g models the sampling of a resulting goal state from the correspond action goal distribution. Accordingly, n_g is a tuple $n_g = \langle \mathbf{s}, a, \gamma, U \rangle$, where $U(\mathbf{s}, a, \gamma) \in \mathbb{R}$ is the value of applying a using template γ in $\mathbf{s}$. The state $\mathbf{s}$ and action a are those of the parent template-selection node n_γ, whereas γ is the template representing the edge with n_γ. Each goal-selection node has action-selection nodes as children, where each child is associated with one goal state $\mathbf{s}' \sim p(\mathbf{s}_{t+1} \mid \mathbf{s}_t = \mathbf{s}, a, \gamma)$. In other words, with the exception of the root node, each action-selection node has a state that represents one potential goal state of applying an action starting in the state of its great-grandparent action-selection node.

Node Statistics and Attributes Additionally, for all node types, we use variables E to keep track of the number of times the node has been visited during the search. We use $E(n) \in \mathbb{N}_+$ to denote the visit count of node n. These statistics enable us to use strategies for trading-off exploration and exploitation based on how many times a node has been previously considered. Furthermore, each node n is associated with a binary label $\rho \in \{\mathit{true}, \mathit{false}\}$ reflecting whether the node is *solved* or not, where $\rho(n)$ denotes the value of this variable for node n. Nodes that are not solved are still "valid" for different procedures during the search. In contrast to that, nodes that are solved should not be selected again as their values have already converged. For example, an action-selection node n_a whose state is physically infeasible will have $\rho(n_a) = \mathit{true}$, in which case we do not select this node for expansion. Moreover, we use $\mathrm{succ}(n)$ to denote the set of successors (children) of a node n, where $\mathrm{succ}(n) = \emptyset$ for leaf nodes. We denote the parent of n by $\mathrm{parent}(n)$, which is *null* only for the root node. Finally, with a slight abuse of notation, we use $\mathbf{s}(n)$, $V(n_a)$, $P(n_a)$, $a(n)$, $Q(n_\gamma)$, $\gamma(n_g)$, and $U(n_g)$ to respectively denote the state $\mathbf{s}$, value V, probability P, action a, value Q, template γ, and value U associated with the corresponding nodes. Figure 5.5 depicts our proposed hierarchical tree structure using an example subtree.

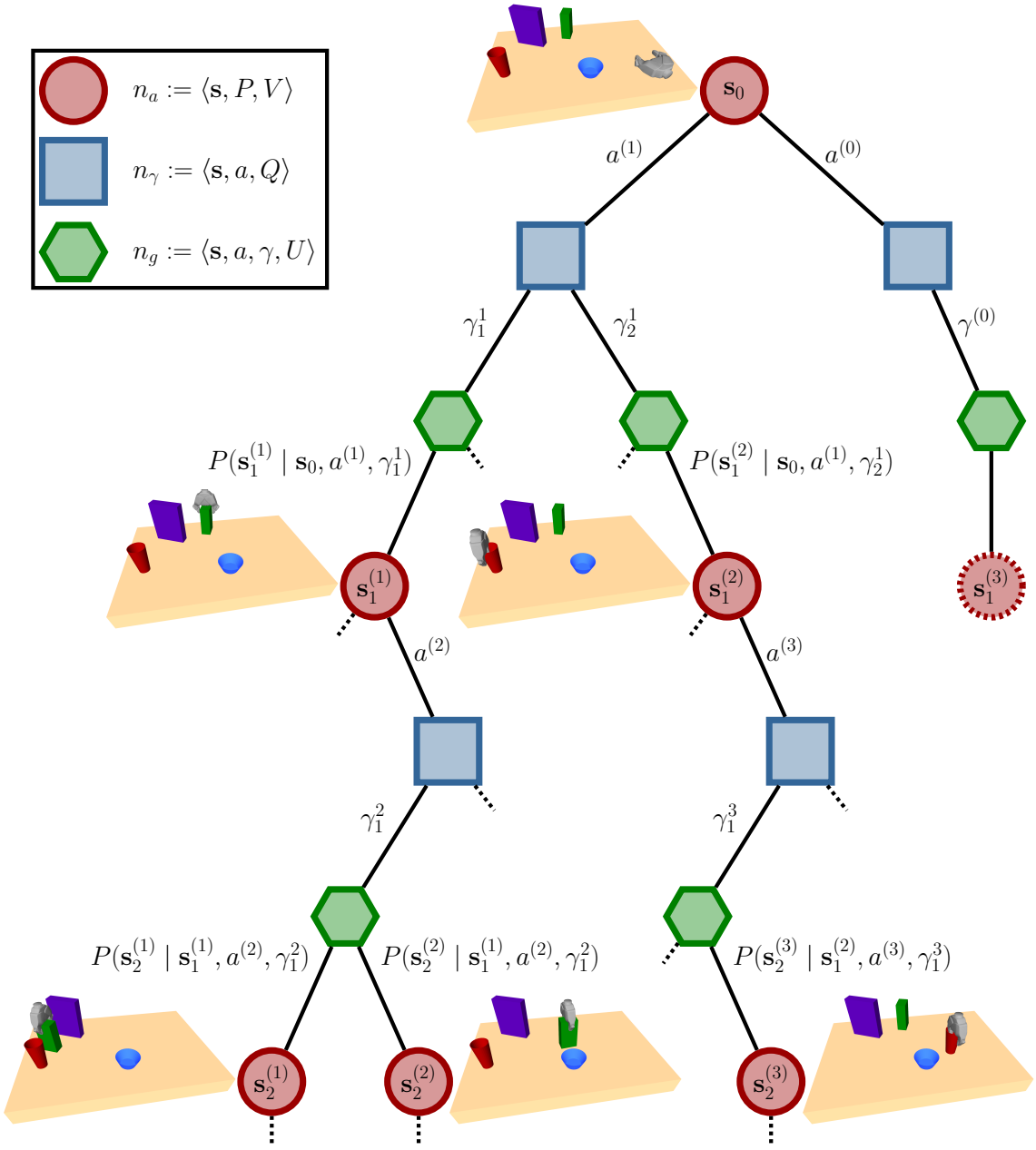


Figure 5.5.: Our proposed tree structure for solving a task based on Monte Carlo tree search. We use action-selection nodes n_a (red circles) to reason about which action to apply in the corresponding state $\mathbf{s}$. A template-selection node n_γ (blue squares) is then responsible for selecting a template $\gamma \in \Gamma^a$ for applying the corresponding action a. Finally, we use goal-selection nodes n_g (green hexagons) to select between the resulting goals based on their probabilities. We show example states next to the corresponding action-selection nodes based on the actions in Figure 5.2. Moreover, we use the special no-op action $a^{(0)}$ to address the stopping problem. Applying $a^{(0)}$ does not change the state (i.e., $\mathbf{s}_1^{(3)} = \mathbf{s}_0$) and results in a solved node ($\rho(n_a) = \mathit{true}$) that cannot be expanded, as indicated by the dotted border.

5.4.5. Heuristic-based MCTS

The basic algorithm in Section 5.4.3 iteratively grows a search tree in order to estimate the state- and state-action values using simulations. Specifically, once a promising node has been selected for expansion, an initial estimate of its value is computed using the simulation procedure, see Figure 5.4. This relies on a default policy (e.g., applying actions randomly until a terminal state is reached) and some model of the environment such as a reward function. At the end of this stage, the observed rewards are used to update the value estimates of all visited nodes in that iteration, thus informing node selection in the next iteration. This assumes a model or an oracle that provides the result of interacting with the environment. For example, this can be a (black box) simulator that models the reward function of an MDP, or a set of game rules to determine the winner at the end of the simulation.

However, we do not assume to have access to such an oracle for specifying feedback as a result of the robot interacting with the environment. At the same time, we assume the teacher does not provide the robot with a formula describing desirable goal states, which can be used to model terminal states with high immediate rewards, or as goals provided to a planner. Therefore, we propose to replace the simulation procedure of MCTS with a heuristic for initializing the values of newly-created nodes. As we aim for task solutions that align with the intention of the teacher with respect to the spatial relations between the objects, we rely on the intention likelihood $\Psi(\mathbf{s})$ of a goal state $\mathbf{s}$ as an initialization of the value $V(\mathbf{s})$ of the corresponding action-selection node n_a. This allows our approach to reason about how good $\mathbf{s}$ is as a final goal of the task.

The use of heuristics in conjunction with MCTS has been previously shown to result in performance improvements. Although MCTS is suited for solving domains where prior knowledge about good policies or heuristics is unavailable, complex domains usually suffer from the large dimensionality of the state space. This means that most states are typically visited a small number of times during search. Therefore, a large number of simulations is typically needed before reliable value estimates are computed in order to steer the search in the right direction. Instead, heuristics (that could rely on domain knowledge) can be used to bootstrap MCTS by initializing values in an informed manner. This heuristic initialization emulates performing a number of simulations that would result in the same value estimates. This circumvents the need for visiting each node at least once before an informed node selection can be made. This concept has been successfully applied to improve the performance of MCTS as in the cases of Gelly and Silver (2007); Winands and Björnsson (2009); Keller and Eyerich (2012); Lanctot et al. (2014). In the context of planning, some approaches argued for completely replacing simulations with heuristics that leverage domain knowledge to improve the efficiency of exploring the state space, e.g., Pellier et al. (2010).

Recently, Keller and Helmert (2013) proposed Trial-based Heuristic Tree Search

(THTS), a framework that subsumes Dynamic Programming, MCTS, and Heuristic Search algorithms for finite-horizon MDPs. This framework allows for generating a large number of algorithms by specifying a few ingredients (e.g., heuristic function, action-selection criterion, etc.), thereby allowing for the combination of both heuristic-based and Monte Carlo-based aspects of tree search. The basic MCTS algorithm can be derived from THTS by replacing the heuristic initialization of new nodes with the simulation stage of MCTS. Since we base our solution on MCTS but rely on the intention likelihood heuristic to initialize values, our algorithm can be seen as a variant of THTS. However, we modify the main THTS ingredients to account for the custom tree structure in Section 5.4.4. In the next section, we detail our heuristic-based MCTS algorithm for solving the optimization problem in Section 5.4.1.

5.5. Our Teach-and-Improvise Algorithm

In this section, we present our algorithm for solving a task from an initial state $\mathbf{s}_0$. As described in Section 5.1, our method relies on an initial *teaching* stage in which the teacher provides a set of task demonstrations. Accordingly, the robot constructs a model of the intention likelihood Ψ of the task as well as models of the actions $\mathcal{A}$. Since the robot is not provided with an explicit goal state when solving the task, it has to reproduce the intention of the teacher by *improvising* a feasible goal state and a plan to achieve it using the learned models. We refer to this concept as *Teach and Improvise* (TI). We build our TI algorithm on heuristic-based MCTS using the tree structure proposed in Section 5.4.4.

We summarize the main structure of TI in SolveTask, see Algorithm 1. As this is an anytime approach, we specify a *time budget* of K iterations as input. We start by using the initial state $\mathbf{s}_0$ to construct a root (action-selection) node n_a of the search tree, i.e., $\mathbf{s}(n_a) = \mathbf{s}_0$. We then execute the main loop of TI, which iteratively grows the tree until either the number K of iterations has been performed, or the root node n_a becomes solved ($\rho(n_a) = \mathit{true}$), indicating that no more node expansions are possible. Each iteration of TI consists of three main steps that are analogous to those of MCTS. The first step (line 8) selects a promising leaf node n_a^* in the tree to expand. This is done using the SelectLeafNode procedure (Algorithm 2), which we explain in Section 5.5.1. The second step (ExpandNode, line 9) expands n_a^* by applying actions from $\mathcal{A}'$ in its state $\mathbf{s}(n_a^*)$. If this succeeds, additional nodes are added to the tree, as we explain in Algorithm 3 and Section 5.5.2. The third step (UpdateValues, line 10) updates the values and statistics of all nodes on the path from n_a^* back to the root node based on the result of the node expansion.

After the main process of constructing the tree is completed, we traverse the tree in order to retrieve the solution of the task, i.e., the best plan leading to the desired

Algorithm 1 Overview of our teach-and-improvise algorithm for solving a task from an initial state $\mathbf{s}_0$ given a budget of K iterations. We assume that the intention likelihood Ψ and action models $\mathcal{A}'$ have been learned from the teacher demonstrations.

```
1: procedure SolveTask(Ψ, A', s_0, K)
2:     // Create root node
3:     prob ← 1                                   // initial state probability
4:     n_a ← CreateActionSelectionNode(s_0, Ψ, prob)
5:     // Run until time is up or the root node is solved
6:     k ← 0
7:     while k < K and ρ(n_a) = false do
8:         n*_a ← SelectLeafNode(n_a)             // select leaf node to expand
9:         ExpandNode(n*_a, Ψ, A')                // expand node by applying actions
10:        UpdateValues(n*_a)                     // backup results to the root
11:        k ← k + 1
12:    // Return the best solution by traversing the tree
13:    return RecommendBestPlan(n_a)
```

state $\mathbf{s}_T^*$. We achieve this using the RecommendBestPlan method, which we explain in Algorithm 6 and Section 5.5.4.

Note that in line 4, we use the method CreateActionSelectionNode for creating and initializing the root node n_a. As shown in Algorithm 4, this uses the intention likelihood Ψ to initialize the value $V(n_a)$ of the root node as the intention likelihood evaluated at $\mathbf{s}_0$, i.e., $V(n_a) = \Psi(\mathbf{s}_0)$. This corresponds to our proposed heuristic-based initialization, and we use this criterion for all subsequently-created action-selection nodes, see Section 5.5.2. For the special case of the root node, we set the probability $P(\mathbf{s}_0)$ of the initial state to 1, (Alg. 1, line 3). In the following sections, we present the details of the procedures in Algorithm 1.

5.5.1. Selecting a Promising Leaf to Expand

In this section, we explain the SelectLeafNode procedure (Alg. 1, line 8) responsible for selecting a leaf node in the tree to expand. We summarize this in Algorithm 2. Starting at the root node n_a, we descend the tree by repeatedly selecting actions (SampleAction), templates (SampleTemplate), and goals (SampleGoalState), until a leaf node is reached.

5.5.1.1. Action Selection

Given an action-selection node n_a, the SampleAction function selects a template-selection child n_γ representing an action $a(n_\gamma)$. We restrict the selection to children

Algorithm 2 Traversing the tree from the root node n_a to select a leaf node to expand.

procedure SELECTLEAFNODE(n_a)
  $n_a^* \leftarrow n_a$ // initialize
  // Sample children until a leaf node is reached
  while succ(n_a^*) $\neq \emptyset$ **do**
    $n_\gamma \sim$ SAMPLEACTION(n_a^*) // select template-selection child
    $n_g \sim$ SAMPLETEMPLATE(n_γ) // select goal-selection child
    $n_a^* \sim$ SAMPLEGOALSTATE(n_g) // select action-selection child
  return n_a^*

that are not solved, since solved nodes lead to leaves that cannot be expanded further. We denote the set of selectable children of n_a by $\mathcal{N}_\gamma(n_a)$, i.e.,

$$\mathcal{N}_\gamma(n_a) = \{n_\gamma \mid n_\gamma \in \text{succ}(n_a), \rho(n_\gamma) = \mathit{false}\}. \tag{5.16}$$

We base action selection on the current value estimates $Q(n_\gamma)$ of children in $\mathcal{N}_\gamma(n_a)$. A child n_γ with a large $Q(n_\gamma)$ value means that the corresponding action $a(n_\gamma)$ led to promising states with respect to the intention likelihood in previous iterations. However, adopting a purely-greedy strategy for action selection might prevent us from considering actions with currently sub-optimal value estimates, but that might lead to better solutions if pursued further. Therefore, we seek strategies to balance exploitation and exploration. We briefly present two strategies that we considered in our work.

ϵ-Greedy This strategy selects the best child (i.e., with the highest Q value) with a certain exploitation probability, and distributes the probability of exploring less optimal actions among the other children equally. We specify this exploitation-exploration trade-off based on the parameter ϵ as follows:

$$P\Big(n_\gamma \in \mathcal{N}_\gamma(n_a) \mid n_a\Big) = \begin{cases} 1 - \epsilon + \frac{\epsilon}{|\mathcal{N}_\gamma(n_a)|} & \text{if } n_\gamma = \underset{n'_\gamma \in \mathcal{N}_\gamma(n_a)}{\arg\max}\, Q(n'_\gamma) \\ \frac{\epsilon}{|\mathcal{N}_\gamma(n_a)|} & \text{otherwise,} \end{cases} \tag{5.17}$$

where $n_a = \text{parent}(n_\gamma)$ and $0 \leq \epsilon \leq 1$. As $\epsilon \to 0$, this converges to a greedy strategy that always selects the best action. On the other hand, as $\epsilon \to 1$, we approach a uniform selection strategy and consider all actions equally.

Boltzmann Exploration This is a common variant of softmax-based strategies (Sutton and Barto, 1998). We select n_γ given its parent n_a with a probability $P(n_\gamma \mid n_a)$:

$$P\Big(n_\gamma \in \mathcal{N}_\gamma(n_a) \mid n_a\Big) = \frac{\exp\Big(Q(n_\gamma)/\tau(n_a)\Big)}{\sum\limits_{n'_\gamma \in \mathcal{N}_\gamma(n_a)} \exp\Big(Q(n'_\gamma)/\tau(n_a)\Big)}. \tag{5.18}$$

Here, $\tau(n_a)$ is the *temperature* that controls the degree of exploration. As $\tau(n_a) \to 0$, we approach a greedy strategy favoring the child with the largest $Q(n_\gamma)$ value. As $\tau(n_a) \to \infty$, we approach a uniform strategy that selects all children with equal probability. Rather than fixing this parameter, we adopt a strategy based on Singh et al. (2000) and gradually decrease $\tau(n_a)$ the more n_a is visited during the search:

$$\tau(n_a) = \frac{1}{\ln\Big(E(n_a) + 2\Big)}, \tag{5.19}$$

where $E(n_a)$ is the visit count of n_a. This encourages more exploration for nodes with low visit counts, as their value estimates are unlikely to have converged yet. For frequently-selected nodes, we encourage exploiting actions with larger value.

5.5.1.2. Action Template Selection

Analogous to action selection, SampleTemplate selects between the (unsolved) children n_g of a template-selection node n_γ, thereby choosing a template $\gamma(n_g)$ of its action $a(n_\gamma)$. We denote the set of selectable children of n_γ by $\mathcal{N}_g(n_\gamma)$, i.e.,

$$\mathcal{N}_g(n_\gamma) = \{n_g \mid n_g \in \text{succ}(n_\gamma), \rho(n_g) = \textit{false}\}. \tag{5.20}$$

As for action selection, we rely on the ϵ-greedy and Boltzmann exploration strategies to compute the probabilities $P\left(n_g \in \mathcal{N}_g(n_\gamma) \mid n_\gamma = \text{parent}(n_g)\right)$ of selecting a template based on the $U(n_g)$ values of the corresponding children $n_g \in \mathcal{N}_g(n_\gamma)$.

5.5.1.3. Action Goal Selection

Finally, given a goal-selection node n_g, we use SampleGoalState to choose an action-selection node whose state represents the result of applying the action $a(n_g)$ using the template $\gamma(n_g)$. Let $\mathcal{N}_a(n_g)$ be the set of unsolved children of n_g, i.e.,

$$\mathcal{N}_a(n_g) = \{n_a \mid n_a \in \text{succ}(n_g), \rho(n_a) = \textit{false}\}. \tag{5.21}$$

We select a node $n_a \in \mathcal{N}_a(n_g)$ with a probability $P(n_a \mid n_g)$ proportional to the probability $P(n_a)$ of its state being an intended goal of the action, i.e.,

$$P(n_a \mid n_g = \text{parent}(n_a)) = \frac{P(n_a)}{\sum\limits_{n'_a \in \mathcal{N}_a(n_g)} P(n'_a)}, \tag{5.22}$$

where,

$$P(n_a) = P\Big(\hat{\mathbf{s}}_{t+1} = \mathbf{s}(n_a) \mid \hat{\mathbf{s}}_t = \mathbf{s}(n_g), a = a(n_g), \gamma = \gamma(n_g)\Big). \tag{5.23}$$

5.5.2. Node Expansion

The SelectLeafNode procedure above traverses the tree to find a promising leaf node n_a^* to expand. In this section, we describe the ExpandNode procedure (Alg. 1, line 9), which expands n_a^* by applying actions in its state $\mathbf{s}_t = \mathbf{s}(n_a^*)$, thereby adding new nodes to the tree. We summarize this procedure in Algorithm 3.

5.5.2.1. Lazy Feasibility Checks

Recall that our problem requires satisfying a feasibility constraint Eq. (5.15) for all states in the solution. This involves performing collision and stability checking computations that might be costly, see Section 2.2. To avoid performing this check for each newly-created node, we adopt a lazy strategy and check this constraint only for nodes that have been selected for expansion. If $\mathbf{s}_t$ is infeasible, we do not expand n_a^*. Instead, we set its value $V(n_a^*)$ to $-\infty$, consequently updating the value of its predecessors, see Section 5.5.3. Additionally, we label n_a^* as solved to prevent it from being considered for expansion again, Alg. 3, lines 4–8.

5.5.2.2. Determining Applicable Actions

If $\mathbf{s}_t$ is feasible, we expand n_a^* with actions from $\mathcal{A}'$. For this, we consider the subset of actions $\mathcal{A}^*$ that satisfy two criteria (GetApplicableActions in Alg. 3, line 9). The first is the state affordability constraint in Eq. (5.14), which we check as described in Section 5.3.3. The second is based on a heuristic that prevents applying the same action in two consecutive time steps. Our motivation for this is based on our point-to-point action models. In this context, there typically is no need for manipulating the same object(s) twice in a row. We use this insight to reduce the branching factor when expanding a node. This can be seen as a form of a history heuristic typically used to prioritize actions based on their performance in the past (Schaeffer (1989); Gelly and Silver (2011)). Accordingly, we define the set of applicable actions $\mathcal{A}^*$ as follows:

$$\mathcal{A}^* = \{a \mid a \in \mathcal{A},\ p(\mathbf{s}(n_a^*) \mid a) \geq \epsilon_s,\ a \neq a(\text{parent}(n_a^*))\}. \tag{5.24}$$

If $\mathcal{A}^*$ is nonempty, we augment it with the no-op action as well, i.e., $\mathcal{A}^* = \mathcal{A}^* \cup \{a^{(0)}\}$. Otherwise, GetApplicableActions returns the empty set (i.e., no node expansion takes place) and we label n_a^* as solved, see Alg. 3, lines 10–12.

5.5.2.3. Sampling Templates to Apply an Action

Given a non-empty set of actions $\mathcal{A}^*$, we expand n_a^* as described in Alg. 3, lines 14–35. For each action $a \in \mathcal{A}^*$, we add a new template-selection node n_γ as a child node of n_a^*, i.e., $\mathbf{s}(n_\gamma) = \mathbf{s}_t$, $a(n_\gamma) = a$ (Alg. 3, lines 14–18). Each n_γ is responsible for applying

Algorithm 3 Expanding a leaf node n_a^* by applying actions in its state.

procedure ExpandNode(n_a^*, Ψ, $\mathcal{A}'$)
  $\mathbf{s}_t \leftarrow \mathbf{s}(n_a^*)$ // the state in which to apply actions
  // Check state feasibility before applying actions
  if not Feasible($\mathbf{s}_t$) **then**
    // The leaf node should not be selected again
    $V(n_a^*) \leftarrow -\infty$
    $\rho(n_a^*) \leftarrow \mathit{true}$
    return
  $\mathcal{A}^* \leftarrow$ GetApplicableActions(n_a^*, $\mathcal{A}'$) // the subset of applicable actions
  if $\mathcal{A}^* = \emptyset$ **then** // no expansion possible
    $\rho(n_a^*) \leftarrow \mathit{true}$ // label leaf node as solved
    return
  // Apply actions and add the corresponding node successors:
  for all $a \in \mathcal{A}^*$ **do**
    // Add n_γ child for action a
    $n_\gamma \leftarrow$ CreateTemplateSelectionNode($\mathbf{s}_t$, a)
    parent(n_γ) $\leftarrow n_a^*$
    succ(n_a^*) $\leftarrow$ succ(n_a^*) $\cup \{n_\gamma\}$
    // Choose a subset Γ^* of the templates for action a
    $\Gamma^* \sim$ SampleSubsetOfTemplates(Γ^a)
    for all $\gamma \in \Gamma^*$ **do**
      // Add the corresponding goal-selection child
      $n_g \leftarrow$ CreateGoalSelectionNode($\mathbf{s}_t$, a, γ)
      parent(n_g) $\leftarrow n_\gamma$
      succ(n_γ) $\leftarrow$ succ(n_γ) $\cup \{n_g\}$
      // Compute discrete action goal states and their probabilities
      $[\hat{\mathcal{S}}_{t+1}, \hat{P}] \leftarrow$ ComputeDiscreteActionGoals($\mathbf{s}_t$, a, γ)
      // Create new action-selection children from the goal states
      for all $\hat{\mathbf{s}}_{t+1} \in \hat{\mathcal{S}}_{t+1}$ **do**
        $\mathit{prob} = P(\mathbf{s}_{t+1} = \hat{\mathbf{s}}_{t+1} \mid \mathbf{s}_t, a, \gamma)$ // goal state probability in $\hat{P}$
        $n'_a \leftarrow$ CreateActionSelectionNode($\hat{\mathbf{s}}_{t+1}$, Ψ, prob)
        if $a = a^{(0)}$ **then** // case of special no-op action
          $\rho(n'_a) \leftarrow \mathit{true}$ // label new leaf node as solved
        parent(n'_a) $\leftarrow n_g$
        succ(n_g) $\leftarrow$ succ(n_g) $\cup \{n'_a\}$ // add new leaf node

its action using its templates Γ^a. We consider multiple templates when applying a and rely on their resulting U values to guide the search in subsequent iterations as discussed

in Section 5.5.1.2.

One way to do this is to consider all action templates, i.e., adding a child n_g of n_γ for each $\gamma \in \Gamma^a$. In practice, however, the corresponding branching factor could render the search intractable, as the size of Γ^a increases with the number of stationary objects $\mathcal{O}_S^a$. Therefore, we propose to approximate the action goal distribution (Eq. (5.8)) using a subset $\Gamma^* \subseteq \Gamma^a$ of templates (SampleSubsetOfTemplates in Alg. 3, line 20). We specify the size of Γ^* based on the ratio $\zeta \in (0, 1]$ of templates to use, i.e., $|\Gamma^*| = \lceil \zeta |\Gamma^a| \rceil$. We construct Γ^* by sampling templates from Γ^a based on their relevance $p(\gamma \mid \mathbf{s}_t, a)$, see Section 5.3.2.1. In Section 5.6, we discuss how the robot can learn the relevance of each action template from experience based on a template's usefulness for solving the task. This allows us to favor sampling more promising templates for node expansion.

5.5.2.4. Computing Goal States

We use the sampled templates Γ^* to apply the corresponding action a and add the resulting nodes to the tree. As motivated in Section 5.3.2.4, we tackle the dimensionality of the problem and the noise in the learned action models by considering each template individually. We add a new goal-selection node n_g as a child of n_γ for each template $\gamma \in \Gamma^*$ to model applying a using γ, i.e., $\mathbf{s}(n_g) = \mathbf{s}_t$, $a(n_g) = a$, $\gamma(n_g) = \gamma$, see Alg. 3, lines 21–25. For this, we compute the distribution of goal states $p(\mathbf{s}_{t+1} \mid \mathbf{s}_t, a, \gamma)$ based on the votes of the stationary object associated with γ as described in Section 5.3.

We discretize this distribution by clustering the sampled goal states as described in Section 5.3.2.3. The result is a finite set of goal states $\hat{\mathcal{S}}_{t+1} = \{\hat{\mathbf{s}}_{t+1}^{(1)}, \dots\}$ and their probabilities $\hat{P} = \{P(\mathbf{s}_{t+1} = \hat{\mathbf{s}}_{t+1}^{(1)} \mid \mathbf{s}_t, a, \gamma), \dots\}$ (Alg. 3, line 27). To avoid expanding a node more than once, we add all resulting goals to the tree. We use each goal state $\hat{\mathbf{s}}_{t+1}^{(s)} \in \hat{\mathcal{S}}_{t+1}$ to construct a new leaf node n'_a as a child of n_g, i.e., $\mathbf{s}(n'_a) = \hat{\mathbf{s}}_{t+1}^{(s)}$, $P(n'_a) = P(\mathbf{s}_{t+1} = \hat{\mathbf{s}}_{t+1}^{(1)} \mid \mathbf{s}_t, a, \gamma)$, see Alg. 3, lines 27–35. In the special case of the no-op action $a^{(0)}$, we label the new leaf node as solved (Alg. 3, lines 32–33). This is equivalent to considering its state as a potential final goal $\mathbf{s}_T$ for solving the task, thereby addressing the stopping problem.

Note that by varying the clustering threshold $\delta_{\max}$, we control the branching factor at goal-selection nodes. Increasing $\delta_{\max}$ results in fewer new leaves, but with a coarser imitation of action goals. As $\delta_{\max} \to 0$, we achieve a fine reproduction of the teacher demonstrations at the price of a larger space of goal states to consider during the search.

5.5.2.5. Value Initialization

In this section, we describe how we initialize new nodes that we add to the tree as a result of the node expansion process in Algorithm 3. This includes new template-selection nodes for the applied actions (CreateTemplateSelectionNode, line 16), goal-selection nodes for their templates (CreateGoalSelectionNode, line 23), and action-selection

Algorithm 4 Initialization of new nodes.

```
1:  procedure CreateActionSelectionNode(s, Ψ, prob)
2:      n_a ← create new action-selection node
3:      s(n_a) ← s                                  // node state
4:      V(n_a) ← Ψ(s)                               // heuristic state-value initialization
5:      P(n_a) ← prob                               // node state probability
6:      E(n_a) ← 1                                  // visit count
7:      ρ(n_a) ← false                              // solved label
8:      return n_a
9:  procedure CreateTemplateSelectionNode(s, a)
10:     n_γ ← create new template-selection node
11:     s(n_γ) ← s                                  // node state
12:     a(n_γ) ← a                                  // node action
13:     Q(n_γ) ← 0                                  // state-action value
14:     E(n_γ) ← 1                                  // visit count
15:     ρ(n_γ) ← false                              // solved label
16:     return n_γ
17: procedure CreateGoalSelectionNode(s, a, γ)
18:     n_g ← create new goal-selection node
19:     s(n_g) ← s                                  // node state
20:     a(n_g) ← a                                  // node action
21:     γ(n_g) ← γ                                  // node action template
22:     U(n_g) ← 0                                  // state-action-template value
23:     E(n_g) ← 1                                  // visit count
24:     ρ(n_γ) ← false                              // solved label
25:     return n_g
```

nodes (leaves) for their goals (CreateActionSelectionNode, line 31). We present the procedures for creating and initializing these nodes in Algorithm 4. For all nodes, we initialize the visit count E to 1 and solved label ρ to *false*. Additionally, we set the probability $P(n_a)$ of each new leaf node n_a to the probability of $\mathbf{s}(n_a)$ being an intended goal of the action used to create n_a (Alg. 4, line 5).

Our approach relies on the value estimates of all nodes to steer the search to promising solutions (Section 5.5.1) and to compute the final plan for solving the task (Section 5.5.4). For template-selection and goal-selection nodes, we respectively initialize the Q and U values to zero (Alg. 4, lines 13 and 22), as we rely on the value-update procedure in Section 5.5.3 below to continuously update these estimates.

Moreover, we rely on the intention likelihood of the task as a heuristic for initializing the state-values V of new leaf nodes (Alg. 4, line 4) as motivated in Section 5.4.5.

Note that the value $V(n_a)$ of any action-selection node n_a estimates the best objective function value (Eq. (5.13)) that can be achieved if we solve the task starting in $\mathbf{s}(n_a)$. Therefore, the initialization $V(n_a) = \Psi(\mathbf{s}(n_a))$ intuitively answers the question: *How good is the state* $\mathbf{s}(n_a)$ *as a final goal state* $\mathbf{s}_T$ *of the task?*

5.5.3. Updating Node Values

In this section, we describe the UpdateValues procedure (Alg. 1, line 10). After each node expansion, this procedure recursively updates the value estimates, visit counts, and solved labels of all nodes on the path from an expanded node back to the root node of the tree. By doing so, we keep track of the best solutions found so far and inform the search in the next iteration.

Traditionally, MCTS-based algorithms for solving MDPs are used to compute optimal policies with respect to the Bellman equation. In such domains, there exist different strategies for updating state and state-action values (Sutton and Barto (1998)). For example, *full* Bellman updates are used when a declarative model of the MDP is available, (i.e., transition probabilities and rewards of all successor states are known). In other cases, one only has access to samples of outcomes and rewards. There, a *partial* or *sample-based* strategy is typically used to update values based on sampled outcomes (e.g., the UCT algorithm, Kocsis and Szepesvári (2006)).

As described in Section 5.5.2, our methods expands a node fully by considering all actions and templates, and subsequently adds all potential goal states to the tree. Therefore, we adopt full value-updating techniques that enable us to update estimates in a dynamic programming fashion that we apply bottom-up, i.e., from the leaves to the root. We explore two such strategies in this work: max-value, and Bellman updates. Algorithm 5 shows our approach with the max-value and Bellman update alternatives highlighted in red and blue for the relevant steps, respectively.

Max-Value Updates This strategy optimizes the objective function in Eq. (5.13) by greedily keeping track of the solution with the largest intention likelihood (minus the plan cost). Note that this does not consider action goal probabilities when computing values, and is therefore more robust to noise in the action models, i.e., if action goal probabilities do not accurately reflect the intention of the teacher. Recall that we initialize the state-value of a leaf node n'_a as the intention likelihood $\Psi(\mathbf{s}(n'_a))$ of its state (Section 5.5.2.5). This reflects the objective function value achieved by solving the task using $T = 0$ steps starting in $\mathbf{s}(n'_a)$. Consequently, we update the state-action-template value $U(n_g)$ of a goal-selection node n_g as the maximum value of its children minus the cost of applying the corresponding action $a(n_g)$, see Alg. 5, line 6. Similarly, we update the state-action value $Q(n_\gamma)$ of a template-selection node n_γ and the state-value $V(n_a)$ of a non-leaf action-selection node n_a as the maximum value of their children,

Algorithm 5 Updating node values and statistics after expanding a node n_a^*. We highlight the max-value and Bellman update alternatives in red and blue for the relevant steps, respectively.

1: **procedure** UPDATEVALUES(n_a^*)
2: **if** succ(n_a^*) $\neq \emptyset$ **then** // if actions were applied from n_a^*
3: // Update values and labels of direct successors of n_a^*
4: **for all** $n_\gamma \in$ succ(n_a^*) **do**
5: **for all** $n_g \in$ succ(n_γ) **do**
6: $U(n_g) \leftarrow -\text{cost}\big(a(n_g)\big) + \max_{n'_a \in \text{succ}(n_g)} V(n'_a)$ // max-value update
7: $U(n_g) \leftarrow -\text{cost}\big(a(n_g)\big) + \sum_{n'_a \in \text{succ}(n_g)} P(n'_a)\, V(n'_a)$ // Bellman update
8: $\rho(n_g) \leftarrow \begin{cases} \mathit{true} & \text{if } \rho(n'_a) = \mathit{true},\ \forall_{n'_a \in \text{succ}(n_g)} \\ \mathit{false} & \text{otherwise} \end{cases}$
9: $Q(n_\gamma) \leftarrow \max_{n_g \in \text{succ}(n_\gamma)} U(n_g)$ // max-value update
10: $Q(n_\gamma) \leftarrow \eta_\gamma \sum_{n_g \in \text{succ}(n_\gamma)} p\big(\gamma(n_g) \mid \mathbf{s}(n_g), a(n_g)\big)\, U(n_g)$ // Bellman update
11: $\rho(n_\gamma) \leftarrow \begin{cases} \mathit{true} & \text{if } \rho(n_g) = \mathit{true},\ \forall_{n_g \in \text{succ}(n_\gamma)} \\ \mathit{false} & \text{otherwise} \end{cases}$
12: // Update value and label of n_a^*
13: $V(n_a^*) \leftarrow \max_{n_\gamma \in \text{succ}(n_a^*)} Q(n_\gamma)$
14: $\rho(n_a^*) \leftarrow \begin{cases} \mathit{true} & \text{if } \rho(n_\gamma) = \mathit{true},\ \forall_{n_\gamma \in \text{succ}(n_a^*)} \\ \mathit{false} & \text{otherwise} \end{cases}$
15: $E(n_a^*) \leftarrow E(n_a^*) + 1$ // increment visit count
16: **if** parent(n_a^*) = *null* **then** // base case: reached the root node
17: **return**
18: $n_g \leftarrow \text{parent}(n_a^*)$
19: $E(n_g) \leftarrow E(n_g) + 1$ // increment parent visit count
20: $n_\gamma \leftarrow \text{parent}(n_g)$
21: $E(n_\gamma) \leftarrow E(n_\gamma) + 1$ // increment grandparent visit count
22: // Go up to the direct action-selection predecessor and run UPDATEVALUES again
23: $n_a^* \leftarrow \text{parent}(n_\gamma)$
24: UPDATEVALUES(n_a^*)

see Alg. 5, line 9 and Alg. 5, line 13, respectively.

Bellman Updates We build this strategy on classical Bellman updates. Rather than maximizing the intention likelihood of the goal state $\mathbf{s}_T^*$ (Eq. (5.13)), this strategy

results in maximizing the *expected* intention likelihood of $\mathbf{s}_T^*$. This results in plans that not only achieve a final goal state with a high intention likelihood, but that also align with the intention of the teacher for the goal of each action. For this, we assume to have action goal distributions that reflect the intention of the teacher accurately. In Section 5.6, we discuss how to learn such models from experience. The only difference between this strategy and max-value updates lies in computing the U and Q values of goal-selection and template-selection nodes, respectively. We compute the values of these nodes as weighted means of their children values based on the corresponding goal and template-relevance probabilities, see Alg. 5, line 7 and Alg. 5, line 10, respectively. In Alg. 5, line 10, $\eta_\gamma = 1/\sum_{n_g} p(\gamma(n_g) \mid \mathbf{s}(n_g), a(n_g))$ is a normalizer based on the children n_g of n_γ. Moreover, we do not incorporate infeasible children with $-\infty$ values, i.e., we consider their probabilities to be zero. If this is the case for all children of a node, we set the node's value to $-\infty$.

Algorithm 5 additionally updates the solved label $\rho(n)$ of a node n to *true* if all children of n are also solved. This prevents n and its successors from being considered for subsequent node selection or expansion. We repeat this recursively by moving one level up to the direct action-selection ancestor of n_a^* and performing the same updates from there while incrementing the visit counts of the encountered nodes, see Alg. 5, lines 18–24. We terminate this procedure after updating the value of the root node (Alg. 5, lines 16–17).

5.5.4. Computing the Best Solution Plan

So far, we presented the main components of TI (Algorithm 1). This explores solutions by repeatedly selecting nodes, expanding them, and updating value estimates until a budget of K iterations is used up or the root node is solved. At this point, we use RecommendBestPlan to traverse the resulting tree and retrieve a plan Π for solving the task (Alg. 1, line 13).

For this, we recall that our approach is an anytime algorithm in which we continuously update value estimates of nodes. Therefore, after any iteration, all node values represent the best estimates for the corresponding tree paths (plans). Specifically, the value $V(n_a)$ of the root node n_a is that of the plan with the maximum objective function value. Accordingly, we adopt a greedy policy and traverse the tree from the root by repeatedly selecting the node child with the largest value until we reach a leaf node n_a^* whose state is the desired task goal $\mathbf{s}_T^*$. We present this method in Alg. 6, lines 2–11.

Note that due to our lazy feasibility-checking strategy (Section 5.5.2.1), we can only guarantee the feasibility of the intermediate states along this path, but not for the state $\mathbf{s}_T^*$ of the leaf node. Therefore, we interleave the process of greedily traversing the tree to retrieve the best plan (GetPlanGreedy in Alg. 6, line 4) with a feasibility check of the final state. If $\mathbf{s}_T^*$ is infeasible, we set the leaf's value to $-\infty$ and update the values of

Algorithm 6 Traversing a search tree starting from the root node n_a in order to retrieve the best solution for the task

1: **procedure** RecommendBestPlan(n_a)
2: **while** *true* **do**
3: // Get max-value plan and leaf node of corresponding path
4: $[\Pi, n_a^*] \leftarrow$ GetPlanGreedy(n_a)
5: $\mathbf{s}_T^* \leftarrow \mathbf{s}(n_a^*)$ // last state of plan
6: **if** Feasible($\mathbf{s}_T^*$) **then**
7: **return** Π // found a feasible solution
8: **else**
9: // Set leaf node value to $-\infty$ (infeasible), update node values, and try again
10: $V(n_a^*) \leftarrow -\infty$
11: UpdateValues(n_a^*)

12: **procedure** GetPlanGreedy(n_a)
13: // Initialize values
14: $n_a^* \leftarrow n_a$
15: $\Pi \leftarrow ()$
16: $t \leftarrow 0$
17: **while** $\text{succ}(n_a^*) \neq \emptyset$ **do** // traverse tree until a leaf is reached
18: // Descend one level by selecting the max-value child of each node
19: $n_\gamma \leftarrow \underset{n_\gamma \in \text{succ}(n_a^*)}{\arg\max}\ Q(n_\gamma)$
20: $n_g \leftarrow \underset{n_g \in \text{succ}(n_\gamma)}{\arg\max}\ U(n_g)$
21: $n_a^* \leftarrow \underset{n_a^* \in \text{succ}(n_g)}{\arg\max}\ V(n_a^*)$
22: // The corresponding action and goal state for this step
23: $a_t \leftarrow a(n_g)$
24: $\mathbf{s}_{t+1} \leftarrow \mathbf{s}(n_a^*)$
25: $\Pi \leftarrow \Pi^\frown\big(\langle a_t, \mathbf{s}_{t+1}\rangle\big)$ // append step to plan
26: $t \leftarrow t + 1$ // increment time step
27: **return** $[\Pi, n_a^*]$

all nodes on the path to the root using UpdateValues as before (Alg. 6, lines 10–11).

We present GetPlanGreedy in Alg. 6, lines 12–27. Starting at the root node n_a, we descend the tree by selecting the child with the largest value. For each action-template-goal sequence of nodes we visit (Alg. 6, lines 19–21), we append the corresponding step $\langle a_t, \mathbf{s}_{t+1}\rangle$ to the plan Π and increment the time step index t (Alg. 6, lines 23–26). Note that the $^\frown$ operator (Alg. 6, line 25) denotes the concatenation of two sequences.

We repeat this until we find a feasible plan Π representing the solution we seek,

i.e., $\Pi = \Big(\langle a_0^*, \mathbf{s}_1^* \rangle, \ldots, \langle a_{T-1}^*, \mathbf{s}_T^* \rangle\Big)$. Note that by assuming that the initial state $\mathbf{s}_0$ is feasible and by incorporating the no-op action $a^{(0)}$, our approach computes solution plans that are at least one step long. As a special case, we compute the trivial plan that corresponds to applying the no-op action, i.e., $\Pi = \Big(\langle a_0^* = a^{(0)}, \mathbf{s}_1^* = \mathbf{s}_0 \rangle\Big)$. In other words, there are no actions to be executed by the robot to solve the task. This can happen if applying any action $a \in \mathcal{A}$ results in infeasible successor states, or if the value $V(\mathbf{s}_0) = \Psi(\mathbf{s}_0)$ of the initial state represents a local optimum such that we do not gain value by moving any object in $\mathbf{s}_0$.

5.5.5. Plan Execution

We briefly address practical considerations that enable a real robot to execute a plan computed using our approach. One way to solve our problem is to first compute a plan without performing feasibility checks, and then rely on a motion planner to check if each step of the plan is feasible during execution. However, this will likely require repeatedly re-planning until a feasible solution is found. On the other hand, computing plans that are feasible on the trajectory level is expensive.

We adopt a compromise based on our point-to-point action models. Our method ensures the feasibility of states before and after applying each action. Therefore, we execute the plan step by step by relying on an existing motion planner to compute a collision-free trajectory for the corresponding action a, and a controller for executing it. In each step, we provide the motion planner with the current object poses, as well as the target poses of the end-effector and other moving objects $\mathcal{O}_M^a$.

By imposing the state affordability constraint in Eq. (5.14), we apply actions in states that resemble the demonstrations with respect to relations between objects in $\mathcal{O}_M^a$. Therefore, we assume that all objects will move as expected when executing the action, as our state affordability model (Eq. (4.14)) does not consider the joint configuration of the end-effector. However, in practice, we additionally have to control the end-effector to grasp and release objects as needed. One way to address this is to rely on expensive physics simulations during planning to verify the grasps necessary for moving objects. Other approaches incorporate the end-effector state (e.g., gripper opening, contact forces, etc.) when learning action models (Niekum et al., 2012; Kalakrishnan et al., 2012; Herzog et al., 2014). This, however, is outside the scope of our work.

We address this issue using a meta-label that specifies whether the end-effector should be open or closed at the end-points of an action. This can either be provided by the teacher or based on the observed gripper opening $\mathbf{o}_{ee}$ during the kinesthetic demonstrations. In our experiments, we set this information using a heuristic based on $\mathcal{O}_M^a$, which requires opening the robot's gripper for actions in which only the end-effector is expected to move. Otherwise, we close the gripper before executing the action and open it afterwards.

Finally, the robot could encounter unexpected situations when executing a plan. For instance, an object might slip from the gripper, or there might be no feasible trajectory for a specific plan step. In such failure cases, one could re-plan starting from the current state. This, however, is outside the scope of this work.

5.6. Updating Action Models from Experience

Our method relies on the learned action models $\mathcal{A}$ to guide planning by sampling promising intermediate goal states. When expanding a node with an action a, we sample a subset of templates $\Gamma^* \subseteq \Gamma^a$ based on the relevance of each $\gamma \in \Gamma^a$. Without prior knowledge about a, we assumed this relevance $p(\gamma \mid \mathbf{s}, a)$ to be uniformly distributed over all templates, see Eq. (4.18) in Section 4.3.3.2. For each template $\gamma \in \Gamma^*$, we compute the distribution of goals $p(\mathbf{s}_{t+1} \mid \mathbf{s}_t, a, \gamma)$ using votes we sample from the voting distribution $\hat{p}_s$ of the corresponding stationary object, see Section 4.3.4.1.

In practice, however, not all templates (i.e., stationary objects) are equally relevant for applying an action. Moreover, as noted in Section 5.3.2.4, the voting distributions are typically noisy due to the small number of demonstrations in which the teacher might solve the task using different action sequences. These issues result in action goal distributions that do not accurately capture the intention of the teacher, thereby driving the search towards irrelevant regions of the state space.

In this section we address these issues by improving action models in two ways:

- We relearn the relevance $p(\gamma \mid \mathbf{s}, a)$ of each template $\gamma \in \Gamma^a$ to reflect its usefulness for solving the task. For instance, in a table-setting task, it is typically more useful to move a cup relative to a plate or a fork compared to a chair.

- Similarly, for each voting distribution $\hat{p}_s$ we aim to update the probability $\hat{w}_s^{(n)}$ of each vote ${}^s\hat{\mathbf{T}}_*^{(n)}$ to reflect its relevance for solving the task.

We propose a method to update these models without requiring additional teacher demonstrations. Instead, we enable the robot to autonomously acquire new experiences by generating additional training data in simulation.

5.6.1. Generating New Training Data

We propose to have the robot "practice" the task in simulation in order to generate new demonstrations. This is analogous to data augmentation methods used in machine learning techniques to complement existing training data with synthesized data (e.g., through adding noise) and achieve better generalization. Similarly, we propose to solve the task in simulation starting from random initial states that the teacher has

not demonstrated before. We treat the resulting plans as additional demonstrations provided by a virtual teacher.

Concretely, we sample a set $\mathcal{S}_0 = \{\mathbf{s}_0^{(1)}, \mathbf{s}_0^{(2)}, \ldots, \mathbf{s}_0^{(S)}\}$ of random and feasible initial states using the objects of the task. For each initial state $\mathbf{s}_0^{(s)} \in \mathcal{S}_0$, we compute a plan $\widetilde{\Pi}^{(s)}$ to solve the task using our method given the intention likelihood Ψ and noisy action models $\mathcal{A}$ learned from the teacher. We use the max-value update strategy (Section 5.5.3), as this does not rely on action goal probabilities for computing node values, and is therefore more robust to noisy action models.

When solving each case, we extend the typical plan structure (Section 2.2) to include the template γ_t that was used to apply the corresponding action a_t in each step, i.e., $\widetilde{\Pi} = \Big(\langle a_0, \gamma_0, \mathbf{s}_1\rangle, \ldots, \langle a_{T-1}, \gamma_{T-1}, \mathbf{s}_T\rangle\Big)$. This is a straightforward modification of Algorithm 6 as γ_t is simply the template $\gamma(n_g)$ of the goal-selection node n_g of the corresponding level of the tree (see Alg. 6, line 20). We therefore modify Alg. 6, line 25 to append $\langle a_t, \gamma_t, \mathbf{s}_{t+1}\rangle$ to the plan, where $\gamma_t \leftarrow \gamma(n_g)$. We denote the set of all plans we compute as $\widetilde{\mathbf{\Pi}} = \{\widetilde{\Pi}^{(1)}, \ldots, \widetilde{\Pi}^{(S)}\}$. We use this data to disambiguate the original teacher demonstrations by revealing which action templates and goals are typically associated with the best solution plan when solving the task, as described next.

5.6.2. Updating Action Template Relevance

Our method computes plans by traversing the search tree along the path of largest node values. Accordingly, all templates included in $\widetilde{\mathbf{\Pi}}$ maximize the U value of the corresponding goal-selection node n_g over all other templates considered for applying the action (see Alg. 6, line 20). For each action $a \in \mathcal{A}$, we compute the number of times each of its templates $\gamma_s \in \Gamma^a$ was selected as part of the solution, and use this as a measure of its relevance.

Let A_i be the total number of times the i-th action $a^{(i)} \in \mathcal{A}$ was applied across all plans in $\widetilde{\mathbf{\Pi}}$. Moreover, let B_s be the total number of times template γ_s was selected for applying this action over all its occurrences in $\widetilde{\mathbf{\Pi}}$, i.e., $A_i = \sum_s B_s$, where $s = 1, \ldots, |\mathcal{O}_S^a|$. We use these statistics to compute the weight $\widetilde{w}_{\gamma_s}$ of γ_s as follows:

$$\widetilde{w}_{\gamma_s} \leftarrow \frac{B_s + 1}{A_i + |\mathcal{O}_S^a|}. \tag{5.25}$$

Accordingly, we update the relevance distribution over templates as follows:

$$p(\gamma_s \mid \mathbf{s}_t, a) = \begin{cases} \widetilde{w}_{\gamma_s} & \text{if } o_s \in \mathcal{O}_S^a(\mathbf{s}_t) \\ 0 & \text{if } o_s \notin \mathcal{O}_S^a(\mathbf{s}_t), \end{cases} \tag{5.26}$$

see Eq. (4.18) for comparison. Note that this is equivalent to assuming a multinomial distribution over the templates, such that Eq. (5.25) approximates the maximum a-posteriori estimate of each template's probability given a Dirichlet prior defined by the initial weights $1/|\mathcal{O}_S^a|$.

5.6.3. Updating the Voting Distributions

Similarly, we use the generated plans $\widetilde{\mathbf{\Pi}}$ to update the voting distribution $\hat{p}_s$ of each stationary object of an action $a^{(i)}$. Recall that $\hat{p}_s$ is a distribution over goal poses ${}^s\mathbf{T}_*$ of the representative moving object o_* relative to a stationary object o_s, see Section 4.3.4.1. We update our estimate for the probability $\hat{w}_s^{(n)} = \hat{p}_s({}^s\mathbf{T}_* = {}^s\hat{\mathbf{T}}_*^{(n)})$ of each potential vote ${}^s\hat{\mathbf{T}}_*^{(n)} \in {}^s\hat{V}_* = \{{}^s\hat{\mathbf{T}}_*^{(1)}, \ldots, {}^s\hat{\mathbf{T}}_*^{(\hat{N}_s)}\}$. Analogous to our initial estimates of these probabilities (Eq. (4.20)), we do this by analyzing how often each vote contributed to goal states for solving the task.

Let ${}^s\widetilde{V}_* = \{{}^s\widetilde{\mathbf{T}}_*^{(1)}, \ldots, {}^s\widetilde{\mathbf{T}}_*^{(A_i)}\}$ be the set of poses of o_* relative to o_s after applying action $a^{(i)}$ across all plans $\widetilde{\mathbf{\Pi}}$, where A_i is the total number of times $a^{(i)}$ was used in $\widetilde{\mathbf{\Pi}}$. Based on our discretization, let $\widetilde{C}_s^{(n)}$ be the set of poses from ${}^s\widetilde{V}_*$ that are within a distance of $\delta_{\max}$ from ${}^s\hat{\mathbf{T}}_*^{(n)}$, i.e., $\widetilde{C}_s^{(n)} = \{{}^s\widetilde{\mathbf{T}}_* \mid {}^s\widetilde{\mathbf{T}}_* \in {}^s\widetilde{V}_*,\ \mathrm{dist}({}^s\widetilde{\mathbf{T}}_*, {}^s\hat{\mathbf{T}}_*^{(n)}) \leq \delta_{\max}\}$. In other words, the size of $\widetilde{C}_s^{(n)}$ indicates the contribution of the corresponding vote ${}^s\hat{\mathbf{T}}_*^{(n)}$ to the plans $\widetilde{\mathbf{\Pi}}$. Accordingly, we update the probability $\hat{w}_s^{(n)}$ of vote ${}^s\hat{\mathbf{T}}_*^{(n)}$ as follows:

$$\hat{w}_s^{(n)} \leftarrow \frac{|\widetilde{C}_s^{(n)}| + \hat{w}_s^{(n)}|{}^sV_*|}{(\sum_{n=1}^{\hat{N}_s} |\widetilde{C}_s^{(n)}|) + |{}^sV_*|}, \tag{5.27}$$

where ${}^sV_*$ is the set of goal poses demonstrated by the teacher, which we used to compute the initial estimate for $\hat{w}_s^{(n)}$, see Eq. (4.20). In this way, we treat the teacher demonstrations as a prior and update $\hat{p}_s$ based on how much the task solutions $\widetilde{\mathbf{\Pi}}$ reflect the usefulness of each vote. Accordingly, votes that are irrelevant for the action (i.e., for which $\widetilde{C}_s^{(n)} = \emptyset$) will be associated with lower probabilities after this procedure.

5.7. Experimental Evaluation

In this section, we present the experimental evaluation of our teach-and-improvise approach. We report the results of an extensive quantitative evaluation in simulation for a variety of tasks, and additionally present experiments with a real PR2 robot. Specifically, we demonstrate that: *i)* our model of the task intention likelihood captures the relevance of pairwise relations to the task, *ii)* our approach enables the robot to compute plans to solve the task by generalizing the demonstrations of the teacher and overcoming the ambiguities in the intended action goals, *iii)* our approach enables the robot to improve its learned action models by practicing the task in simulation, *iv)* our approach enables the robot to adapt the solution of the task based on the starting state, and *v)* our method is applicable on a real robot and is able to compute physically feasible plans that adapt to the starting state.

Table 5.1.: A summary of the notation we adopt for presenting our experimental results. We tested different variants of our algorithm based on the value update technique (max-value or Bellman), exploration-exploitation trade-off criterion (ϵ-greedy or Boltzmann exploration), and the ratio ζ of action templates sampled during node expansion.

	Before updating action models		After updating action models			
	ϵ-greedy	Boltzmann	ϵ-greedy		Boltzmann	
	$\zeta = 1$		$\zeta = 1$	$\zeta = 0.5$	$\zeta = 1$	$\zeta = 0.5$
Max-Value updates	MaxVal-ϵ	MaxVal-B	MaxVal-ϵ*	MaxVal-ϵ* (0.5)	MaxVal-B*	MaxVal-B* (0.5)
Bellman updates	Bellman-ϵ	Bellman-B	Bellman-ϵ*	Bellman-ϵ* (0.5)	Bellman-B*	Bellman-B* (0.5)

Table 5.2.: The color scheme we adopt when presenting the results of our algorithm based on the number of teacher demonstrations $\mathcal{D}^{(\mathcal{T})}$. We illustrate this for the MaxVal-B* variant in Table 5.1.

$\lvert\mathcal{D}^{(\mathcal{T})}\rvert = 2$ demos	$\lvert\mathcal{D}^{(\mathcal{T})}\rvert = 4$ demos	$\lvert\mathcal{D}^{(\mathcal{T})}\rvert = 6$ demos	$\lvert\mathcal{D}^{(\mathcal{T})}\rvert = 8$ demos	$\lvert\mathcal{D}^{(\mathcal{T})}\rvert = 10$ demos

5.7.1. Notation and Parameters

In this section, we describe the parameter values we used in our experiments and briefly summarize the notation we adopt when reporting our results. We tested variants of our algorithm based on the value update technique (max-value or Bellman, see Section 5.5.3), exploration-exploitation trade-off criterion for node selection (ϵ-greedy or Boltzmann exploration, see Section 5.5.1), and the ratio ζ of action templates sampled during node expansion, which we set to either 0.5 or 1 (see Section 5.5.2.3). Additionally, we evaluated the ability of our approach to compute task solutions before and after updating the learned action models by practicing the task in simulation as described in Section 5.6. We summarize the variants resulting from the combinations of these factors and parameters in Table 5.1. The table also shows the notation and symbols we adopt when plotting the corresponding results. Moreover, we evaluated our approach when learning using two, four, six, eight, or ten demonstrations of a task. Accordingly, Table 5.2 shows the color scheme we adopt for each case. Finally, unless otherwise stated, we set all other parameters of our algorithm as in Table 5.3.

Table 5.3.: The default parameter values we used in our experiments.

Gaussian kernel bandwidth σ (Eq. (4.11))	0.05
Clustering compactness threshold $\delta_{\max}$ (Section 5.3.2.3)	0.05
Number I of samples drawn per voting distribution (Section 5.3.2.3)	5
ϵ-greedy parameter ϵ (Section 5.5.1.1)	0.2
The cost c_0 of applying an action from $\mathcal{A}$	0.001

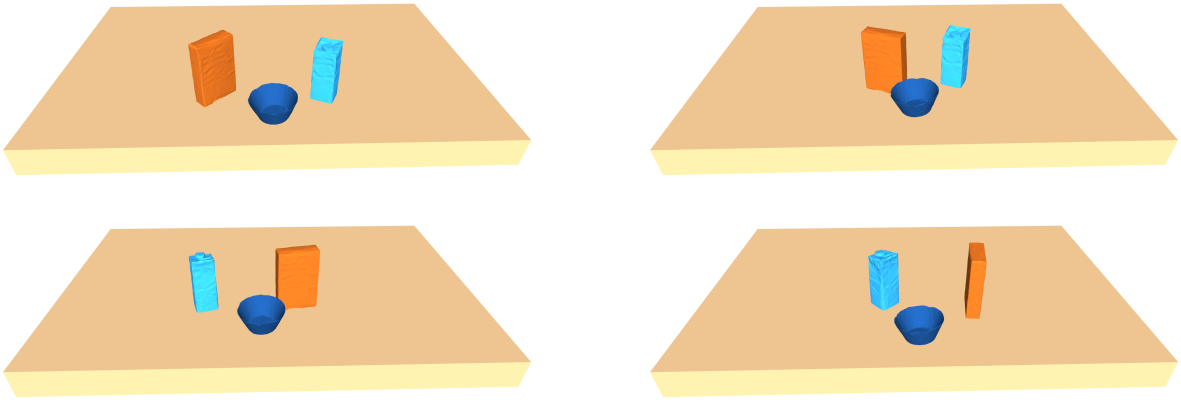


Figure 5.6.: Examples of demonstrated goal states for the breakfast setting task $\mathcal{T}_1$. The teacher placed the bowl at the front edge of the table and the other two objects on opposite sides relative to the bowl.

5.7.2. Tasks

We evaluated our approach extensively using four tasks, $\mathcal{T}_1$-$\mathcal{T}_4$, which we briefly describe below. Moreover, we present experiments with a real PR2 robot using additional tasks in Section 5.7.11.

5.7.2.1. Breakfast Setting ($\mathcal{T}_1$)

In this task, the teacher demonstrated how to set a breakfast table using a bowl, a box of cereals, and a carton of milk. The teacher always placed the bowl at a random spot near the front edge of the table, and placed the box of cereals and milk at opposite sides of the bowl. This corresponds to two different *modes* of solving the task as shown in Figure 5.6. Note that the exact poses of the objects in each mode varied randomly. As ground truth, we considered all six pairwise relations between the objects (including the table) to be relevant for solving the task. In total, this task involved four actions.

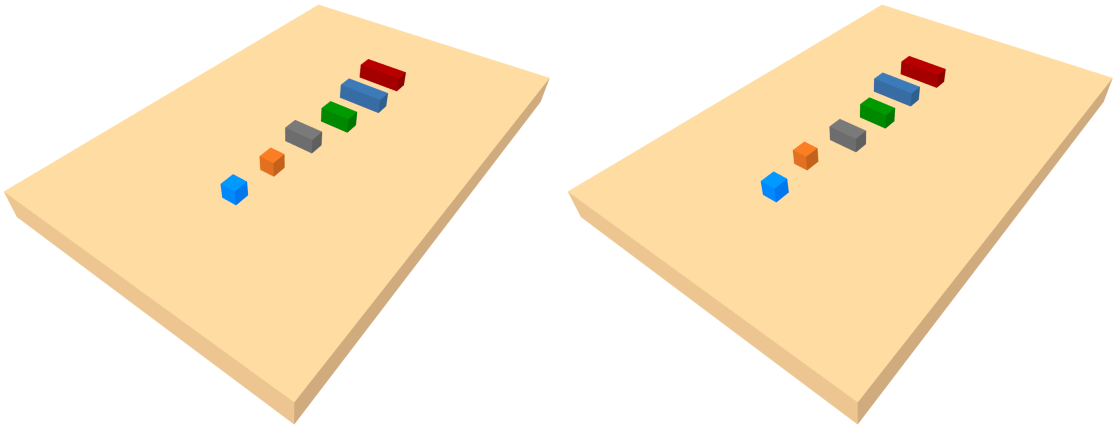


Figure 5.7.: Examples of demonstrated goal states for the aligning blocks task $\mathcal{T}_2$ in which the teacher constructed a row of blocks in a specific order.

5.7.2.2. Aligning Blocks ($\mathcal{T}_2$)

In this task, the teacher demonstrated how to construct a row of six blocks by aligning them in a specific order on the table as shown in Figure 5.7. The desired position of each block varied randomly with a tolerance of 2 cm such that neighboring blocks are 15 cm apart on average. Moreover, due to the symmetrical shape of the blocks, the teacher placed them at random orientations (0 or 180 deg) around the z-axis in each demonstration. As ground truth, we considered all 21 pairwise relations between the objects (including the table) to be relevant for solving the task. In total, this task involved seven actions.

5.7.2.3. Tower Building ($\mathcal{T}_3$)

In this task, the teacher demonstrated how to construct a tower of six blocks by stacking them in a specific order as shown in Figure 5.8. The teacher varied the position and orientation of the tower relative to the table across the demonstrations. When placing a block on top of another, the teacher varied the displacement between the two blocks along their main axis within a range of ± 2 cm. The teacher also randomly changed the relative orientation between the blocks to be either 0 or 180 deg around the z-axis. As ground truth, we considered only the 15 pairwise relations between the six blocks to be relevant for solving the task (i.e., all relations involving the table are irrelevant). In total, this task involved seven actions.

5.7.2.4. Tidy-Up ($\mathcal{T}_4$)

In this task, the teacher demonstrated a tidy-up task using six objects: three toy boxes, a bowl, a cup, and a pack of sugar. The goal of the task is to tidy up the boxes by

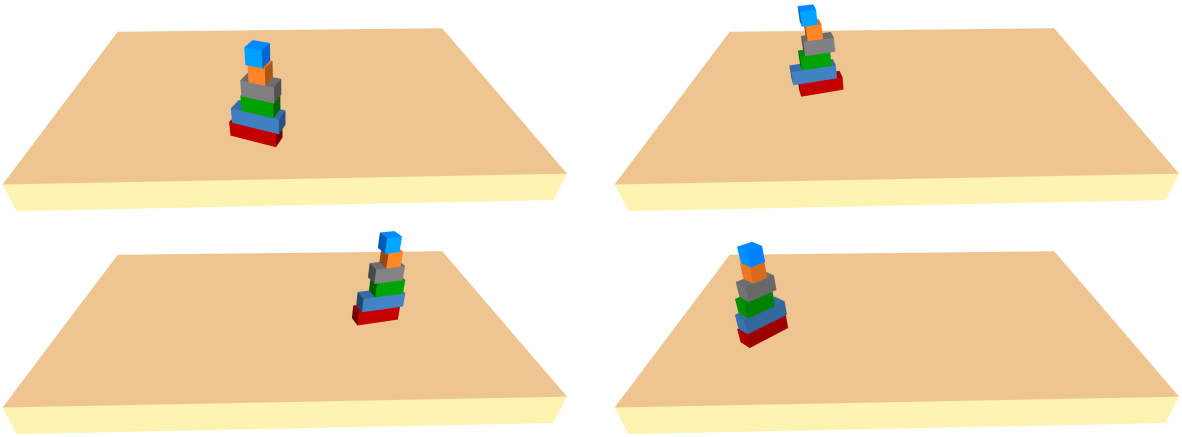


Figure 5.8.: Examples of demonstrated goal states for the tower building task $\mathcal{T}_3$. The teacher stacked six blocks in a specific order such that the position of the tower on the table is irrelevant.

moving them to the back of the table while arranging the kitchen-related objects on the opposite end of the table similar to the breakfast task $\mathcal{T}_1$ above. The teacher showed different modes for solving the task based on the arrangement of the boxes and those of the kitchen-related objects, see Figure 5.9. The teacher varied the object poses randomly around their desired values. As ground truth, we considered only 12 pairwise relations to be relevant for solving the task: those involving the kitchen-related objects with each other and the table, and those involving the boxes with each other and the table. We considered relations between the boxes and the kitchen-related objects to be irrelevant for the task. In total, this task involved seven actions.

We summarize the above tasks in Table 5.4.

5.7.3. Task Demonstrations

In this section, we briefly describe how we acquired demonstrations for learning and evaluating the tasks above. For an extensive evaluation, we aimed to generate a large number of demonstrations for each task. This enables us to sample small subsets of demonstrations to use for learning each task and evaluating our approach. For recording demonstrations, we constructed 3D models of the involved objects and relied on SimTrack (Pauwels and Kragic, 2015) to recognize them and compute their poses using a depth camera, see Figure 5.10. Additionally, we relied on fiducial markers to emulate the end-effector of the robot and estimate the table surface as in Figure 4.2. We used the ar_pose library for detecting the fiducial markers (Dryanovsk et al.). Note that for tasks $\mathcal{T}_2$ and $\mathcal{T}_3$ above we relied on virtual blocks in simulation.

Using this setup, we first recorded ground truth point-to-point action models by

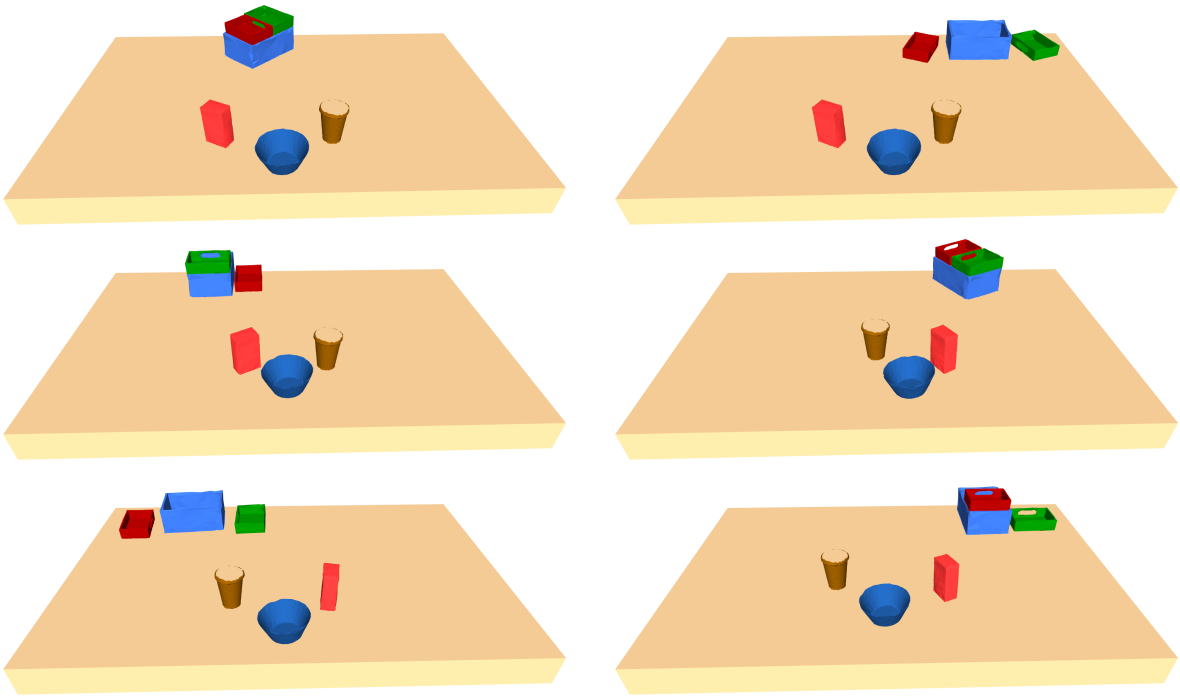


Figure 5.9.: Examples of demonstrated goal states for the tidy-up task $\mathcal{T}_4$. The teacher arranged the toy boxes on one side of the table and the kitchen-related objects on the opposite end. The demonstrations involved different modes of solving the task with respect to the object arrangements.

Table 5.4.: Summary of the tasks we considered in our evaluation along. The number of objects includes the table.

Task	# objects	# relevant relations	# actions
Breakfast setting ($\mathcal{T}_1$)	4	6	4
Aligning blocks ($\mathcal{T}_2$)	7	21	7
Tower building ($\mathcal{T}_3$)	7	15	7
Tidy-up ($\mathcal{T}_4$)	7	12	7

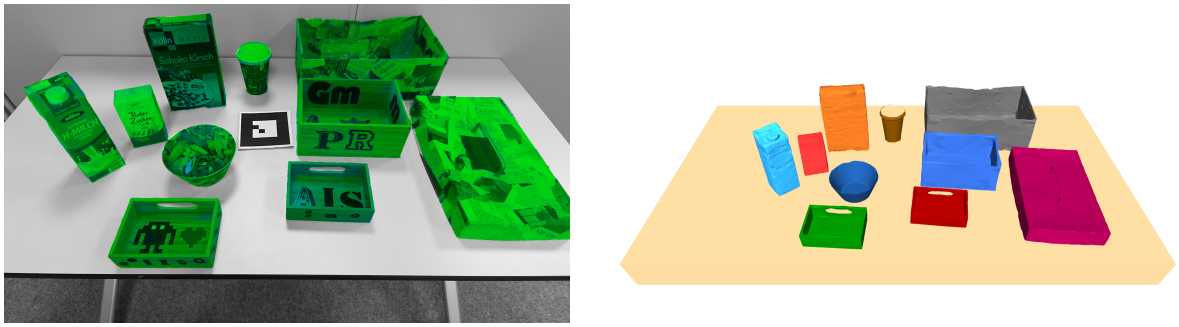


Figure 5.10.: To record demonstrations, we relied on an existing pose detector and tracker (Pauwels and Kragic, 2015) based on 3D models of the objects. Additionally, we used fiducial markers to facilitate recognizing the table and to emulate the end-effector of the robot. Left: the camera image of a scene with the object detections (green masks) superimposed. Right: the corresponding recorded scene in the rviz visualization environment.

demonstrating the spatial relations between the objects before and after applying an action. We provided several variations for each action with respect to the starting and goal states. Moreover, we recorded 100 ground truth goal states for each task. Using these models, we used our MCTS-based planner to compute plans for solving each task (i.e., achieving a ground truth goal state) using the ground truth action models. In this way, we generated 100 plans to solve each task starting from random initial states with all objects on the table. Finally, we used the resulting plans as demonstrations provided by a virtual teacher for learning each task, which we segmented using our approach. It is important to note that we only used the ground truth action and task goal models for the purpose of generating this pool of demonstrations. Beyond this, we do not use these models in our evaluation, and rely solely on the resulting demonstrations to learn each task using our approach. Figure 5.11 shows an example of such a demonstration for the tidy-up task $\mathcal{T}_4$.

5.7.4. Estimating the Relevance of Relations for Task Goals

As a first evaluation, we investigated the weights estimated by our method for the relevance of pairwise relations for the intention likelihood of each task. We computed the weight of each relation based on the entropy in the demonstrations as in Eq. (5.7). To measure how closely these weights reflect the intention of the teacher, we computed the cosine similarity between the vector of estimated weights using our method and that consisting of the ground truth weights (with zeros for irrelevant relations and ones for relevant ones). We performed this for each task given two, four, six, eight, and ten demonstrations. The results are shown in Figure 5.12 averaged over 25 runs with

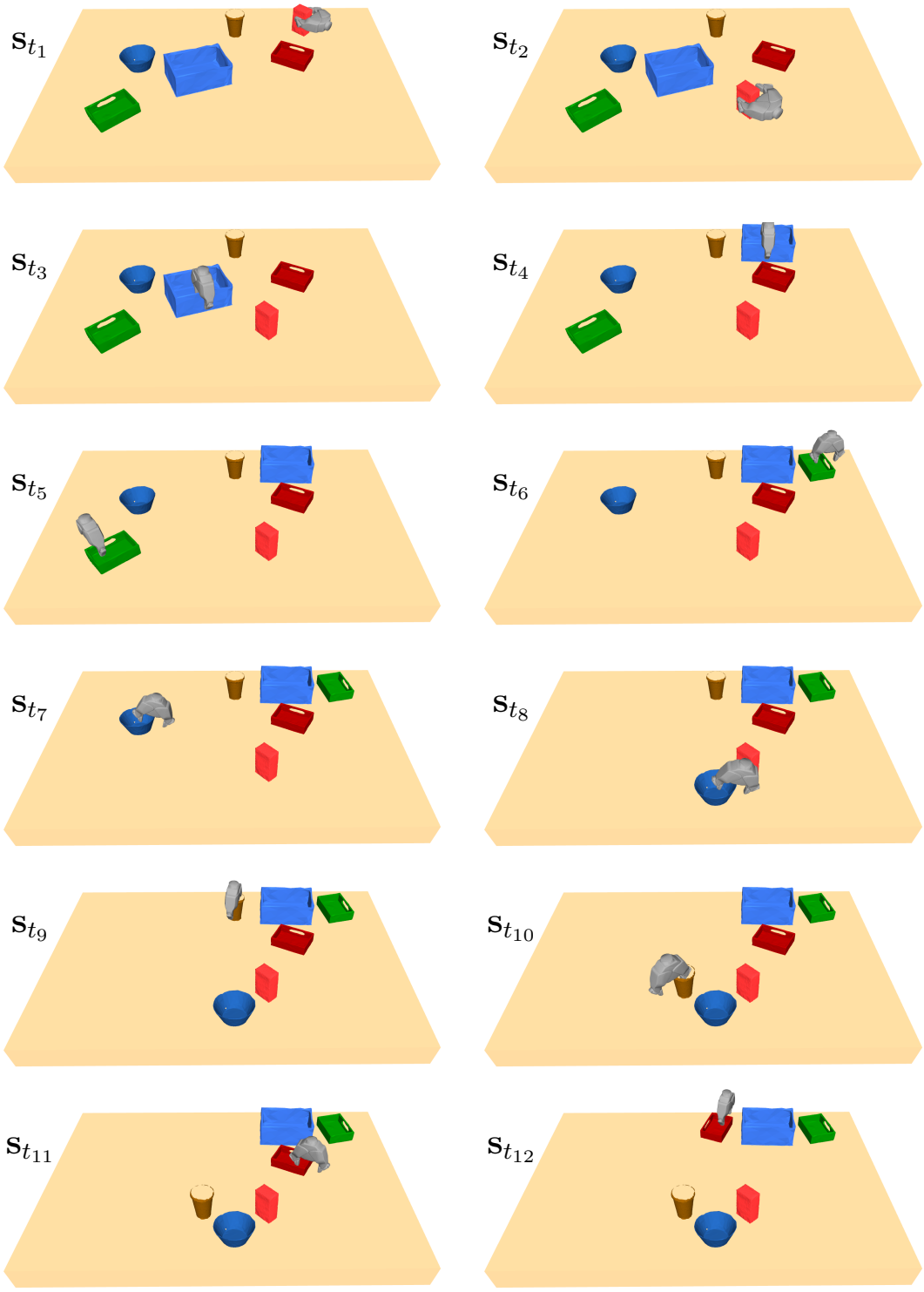


Figure 5.11.: Example demonstration for the tidy-up task $\mathcal{T}_4$. We omit the initial state at t_0 before moving the end-effect from a default starting pose. We generated such demonstrations in simulation for all tasks in our experiments. All demonstrations start with the objects placed randomly on the table. The demonstrations varied with respect to the order of applying actions.

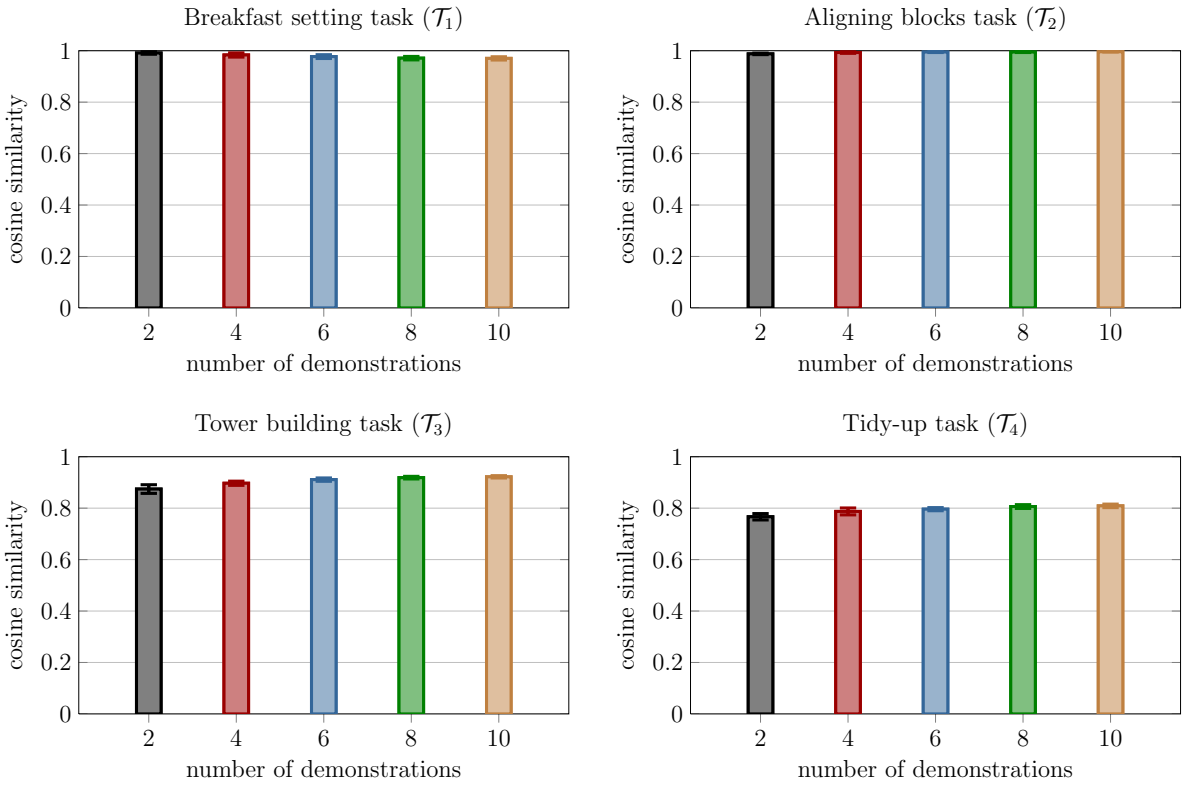


Figure 5.12.: The cosine similarity between the weights of pairwise relations estimated by our approach and the ground truth weights for tasks $\mathcal{T}_1$-$\mathcal{T}_4$. We compute the weights based on the entropy in the demonstrations as in Eq. (5.7). For $\mathcal{T}_1$ and $\mathcal{T}_2$, we achieve a similarity between 0.97 and 0.99. For $\mathcal{T}_3$ and $\mathcal{T}_4$, the similarity improves from 0.87 to 0.92 and from 0.77 to 0.81, respectively, as the number of demonstrations increases from two to ten. For those tasks, only a subset of the relations are relevant (15 and 12 out of 21, respectively). This makes it challenging to estimate the relevance of each relation from a few demonstrations.

randomly-selected demonstrations for each task.

For tasks $\mathcal{T}_1$ and $\mathcal{T}_2$, we achieve a similarity between 0.97 and 0.99 for all numbers of demonstrations. Note for those two tasks, all pairwise relations are relevant. On the other hand, tasks $\mathcal{T}_3$ and $\mathcal{T}_4$ involve 15 and 12 relevant relations out of all 21 possible pairwise relations, respectively. It is therefore more challenging to estimate which relations are more relevant than the others given a small number of demonstrations. This is particularly the case for $\mathcal{T}_4$ since the teacher demonstrates multiple ways of solving the task as shown in Figure 5.9. Additionally, some objects in those tasks have symmetries in their shapes, which adds to the ambiguity with respect to the geometric relations involving them. This, however, is an aspect we do not account for

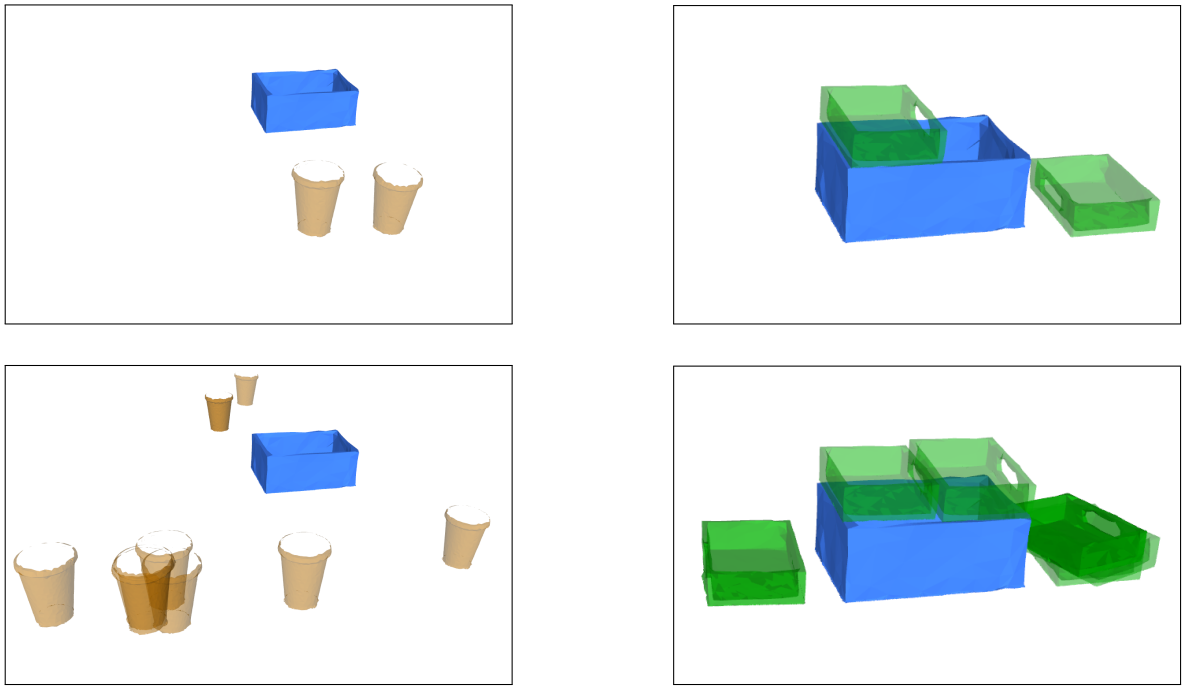


Figure 5.13.: Two examples of relations in the tidy-up task, $\mathcal{T}_4$. The left side shows the relation between the blue box and the cup, and the right side shows the relation between the blue and green boxes. We illustrate the distributions of these relations given two (top) and ten (bottom) demonstrations. With more demonstrations, we estimate a higher weight for the relation between the boxes due to more consistency in the corresponding relative poses.

in our work. Overall, for $\mathcal{T}_3$, the cosine similarity increases from 0.87 to 0.92 as the number of demonstrations increases from two to ten. Similarly, the cosine similarity for $\mathcal{T}_4$ changes from 0.77 to 0.81 as the number of demonstrations increases from two to ten. See Figure 5.13 for examples of two relations for task $\mathcal{T}_4$.

In the next experiments, we demonstrate that, despite the ambiguity in the intention of the teacher with respect to the task goals, our approach is able to leverage the noisy action models learned from the demonstrations in order to compute goal states corresponding to peaks of those relations.

5.7.5. Solving Tasks Starting in Arbitrary States

The purpose of this experiment is to evaluate the ability of our approach to generalize the demonstrations of the teacher and to solve tasks from arbitrary initial states that have not been demonstrated before. For this, we conducted an extensive evaluation using tasks $\mathcal{T}_1$-$\mathcal{T}_4$ in simulation.

5.7.5.1. Training and Testing Data

For training, we used N demonstrations for each task, which we generated as described in Section 5.7.3 above. We varied the number of demonstrations between two and ten in increments of two. In each case, we used our approach to learn action models from the segmented demonstrations, and to learn a model of the intention likelihood based on the final state. For each task, we then sampled 100 random initial states with all objects placed at an arbitrary pose on the table.

We used our teach-and-improvise algorithm (TI) described in Section 5.5 to solve each task starting in each of the 100 random initial states using a budget K of 50,000 iterations, and *without* providing an explicit goal state to be achieved. In this experiment, we relied on a 3D model of the robot's end-effector and used feasibility constraints that consider collisions between the objects as well as state stability as described in Section 2.2, i.e., we did not consider inverse-kinematic constraints.

We repeated this experiment 25 times, each time using a different random set of N demonstrations. In total, we evaluated each task on 2,500 cases using different combinations of demonstrations and novel initial states. Note that here, we used the action models learned directly from the teacher demonstrations, i.e., without updating the action distributions by practicing the task. Since TI is an any-time algorithm, we analyzed the computed solutions every 500 iterations.

5.7.5.2. Analyzing Satisfied Relations

We investigated the task solutions in the experiment above with respect to the pairwise relations achieved in the final goal state $\mathbf{s}_T$ of each computed plan. Specifically, we computed the percentage of *relevant* relations that were successfully reproduced for each task (see Table 5.4 for an overview of how many relevant relations are involved in each task). We considered a relation to be *satisfied* or successfully reproduced if the relative pose between the two corresponding objects in $\mathbf{s}_T$ is not further than a distance of $\delta_{\max} = 0.05$ from one of the relation poses demonstrated by the teacher for that task, where $\delta_{\max}$ is the compactness threshold used for discretizing the action distributions. For all tasks $\mathcal{T}_1$-$\mathcal{T}_4$, we first performed this experiment using max-value updates and Boltzmann exploration (MaxVal-B). Figure 5.14 shows the results for all numbers of demonstrations considered. Note that all figures show results evaluated at 100 points in each run (every 500 iterations). However, we visualize the markers and standard deviations in the plots at sparser intervals for readability.

For the breakfast setting task ($\mathcal{T}_1$), we achieved success rates of 76.7%-85.6% on average after only 500 iterations, reaching 98.1% and 95.3% for two demonstrations and ten demonstrations, respectively. For all numbers of demonstrations, we achieved between 96.5% and 98.3% on average after 50,000 iterations, see Figure 5.14-top left. Despite the fact that all tested cases started in initial states not demonstrated by the

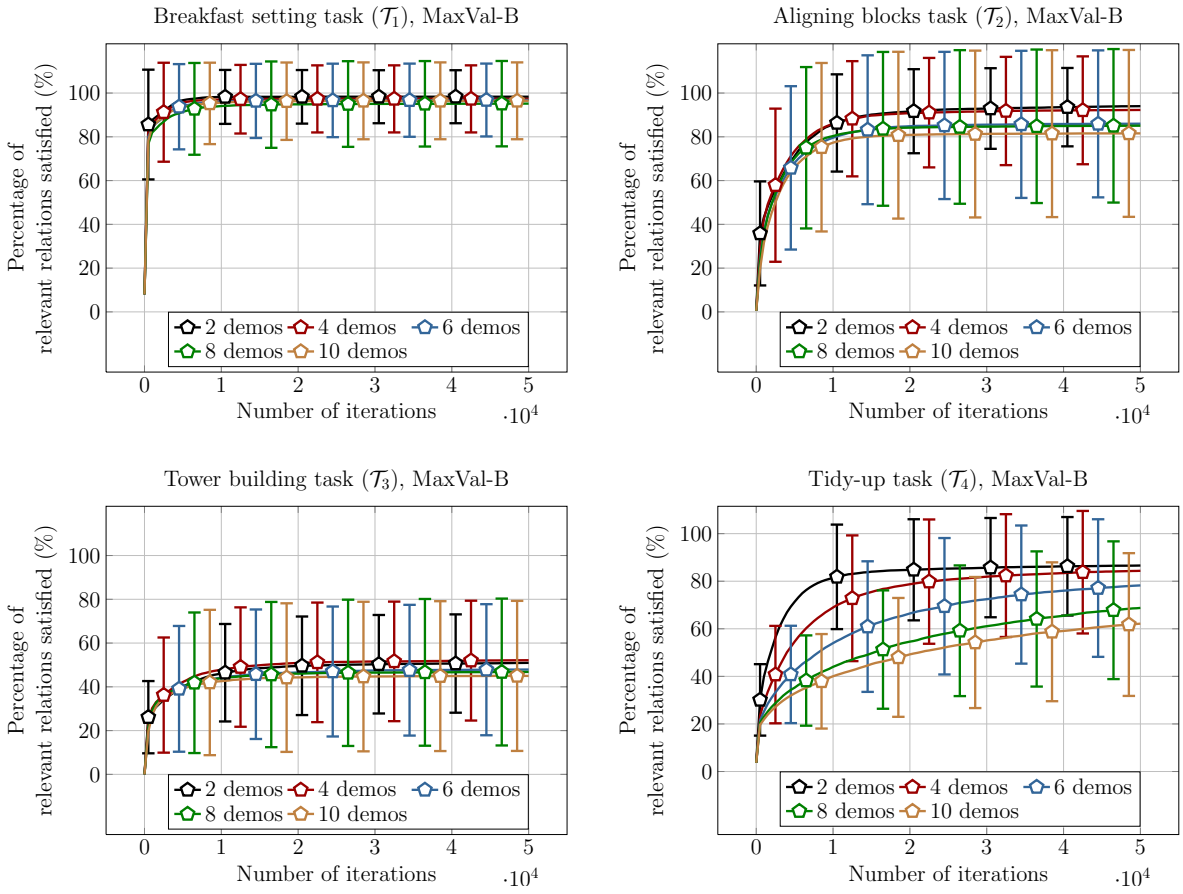


Figure 5.14.: The results of our approach using max-value updates when solving tasks $\mathcal{T}_1$-$\mathcal{T}_4$ starting in 100 random initial states. Despite not providing our approach with an explicit goal state to achieve in each case, we are able to generalize the demonstrations of the teacher and compute plans to satisfy the relevant relations between the objects. Our approach is anytime in nature and improves the computed arrangements based on the specified budget of iterations. See Figures 5.15 and 5.16 for examples of solutions computed by our approach for tasks $\mathcal{T}_3$ and $\mathcal{T}_4$.

teacher beforehand, we are able to generalize the demonstrations by sequencing actions in order to maximize the intention likelihood of the goal state and satisfy the desired spatial relations between the objects. Using MaxVal-B, our approach computed plans consisting of 8.2 actions on average for $\mathcal{T}_1$.

Analogously, we show the results for the aligning blocks task ($\mathcal{T}_2$) in Figure 5.14-top right. After 50,000 iterations, our approach (using MaxVal-B) achieves 94.0% of the (21) relevant relations on average when learning from two demonstrations, and 81.6% of the relations when learning from ten demonstrations. Our approach is able to generalize the teacher demonstrations and compute plans in novel initial states to maximize the

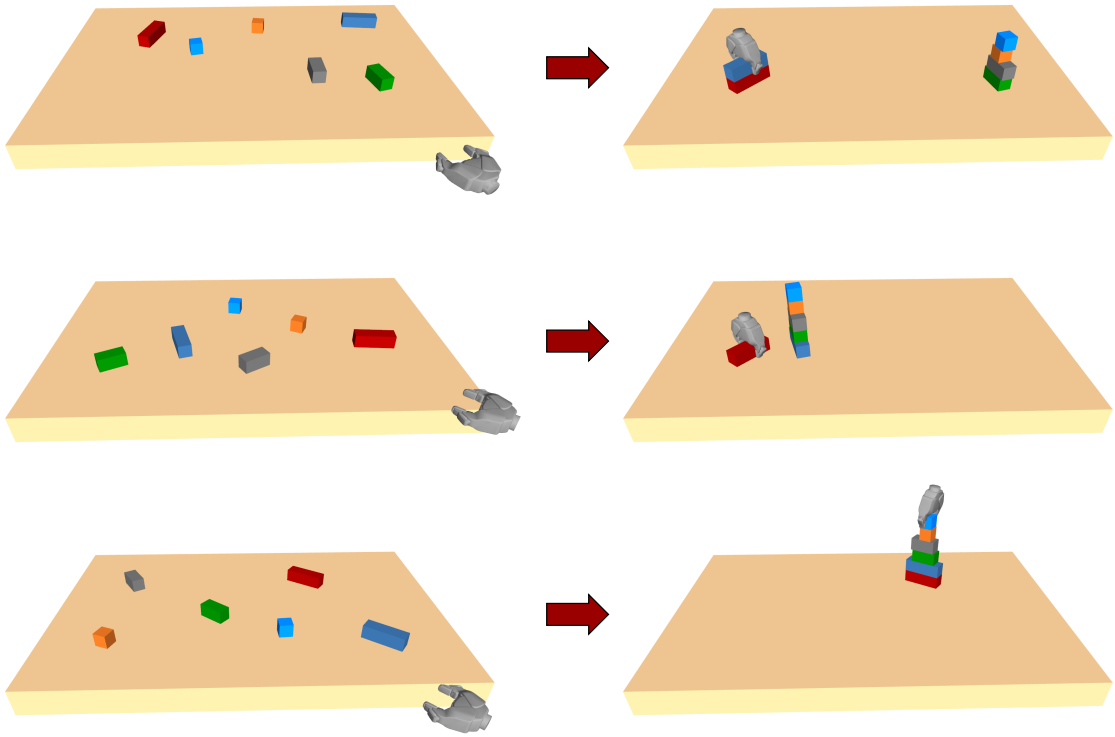


Figure 5.15.: Examples of solutions computed using our approach for the tower building task $\mathcal{T}_3$. The left side shows random initial states (previously not demonstrated by the teacher), and the right side shows the corresponding goal states achieved using our method. In each example, our approach generalizes the demonstrations of the teacher to compute solutions that satisfy 46.7% (top), 66.7% (middle), and 100% of the desired relations between all six blocks.

number of desired relations between the blocks on the table. Note that since we adopt a non-parametric model for the action distributions based on the demonstrated poses, more demonstrations result in an increased branching factor when expanding nodes during planning. This results in a slower convergence when using more demonstrations, leading to less satisfied relations for the same number of iterations. In Section 5.7.7, we show how this effect diminishes when allowing the robot to update the action models by practicing the task in simulation.

Additionally, we show the results for the tower building task ($\mathcal{T}_3$) in Figure 5.14-bottom left. Here, our approach with MaxVal-B is able to compute plans that result in satisfying between 45.0% and 52.2% of the desired relations between all blocks on average. As an example, given six demonstrations, MaxVal-B computed plans consisting of 10.4 actions

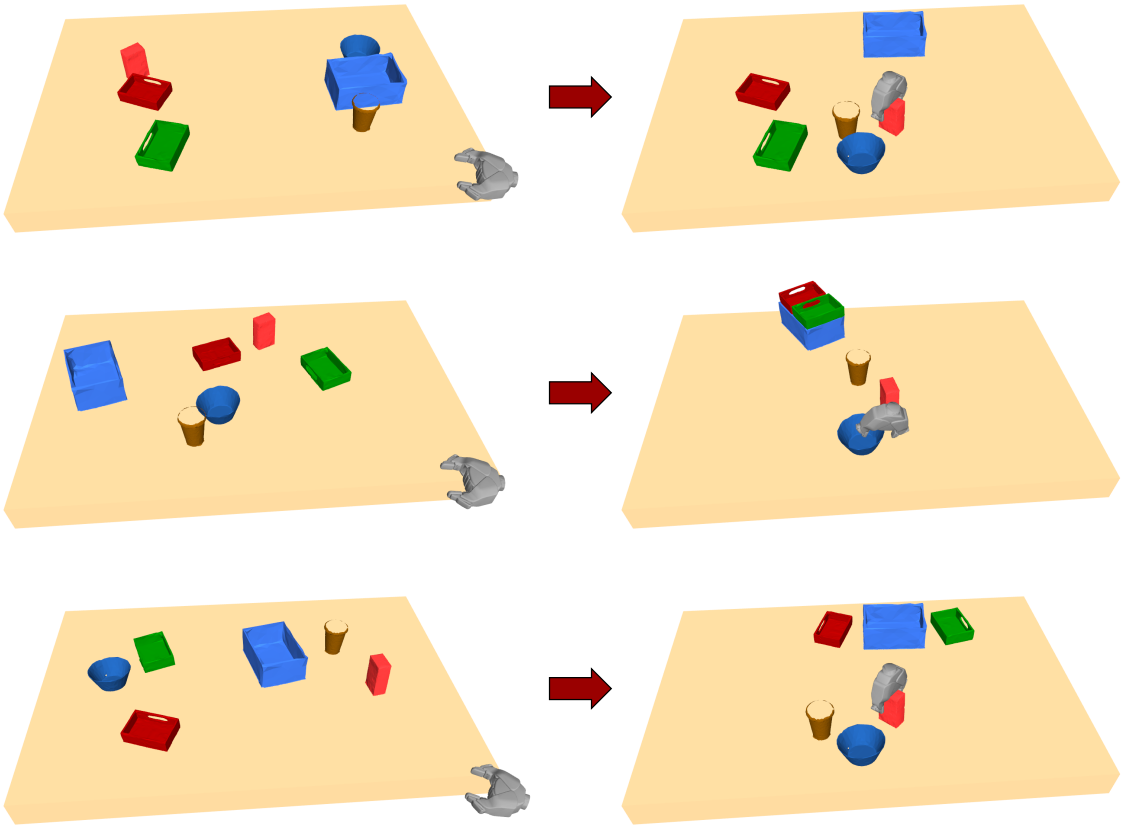


Figure 5.16.: Examples of solutions computed using our approach for the tidy-up task. The left side shows random initial states (previously not demonstrated by the teacher). The right side shows the corresponding goal states achieved using our method. In each example, our approach generalizes the demonstrations of the teacher to compute solutions that satisfy 58.3% (top), 75% (middle), and 100% of the desired relations between the objects.

on average. These resulted in satisfying between 40% and 50% of the relations in 47.0% of the solved cases, and 100% of the relations in 15.2% of the cases. Note that in this task (as opposed to the aligning blocks task), the order of applying actions to stack the blocks is crucial for the quality of the final solution due to the stability constraints (a block can only be moved to be placed on top of another). Therefore, constructing a tower of six blocks requires manipulating the blocks in the correct order. This makes it challenging to recover from locally-optimal solutions during the tree search since the teacher did not demonstrate how to unstack blocks, for example by placing them on the table. Nonetheless, our approach improvises solutions that correspond to locally-optimal stacks of blocks with respect to the intention likelihood. Figure 5.15 shows examples of such solutions computed by our approach.

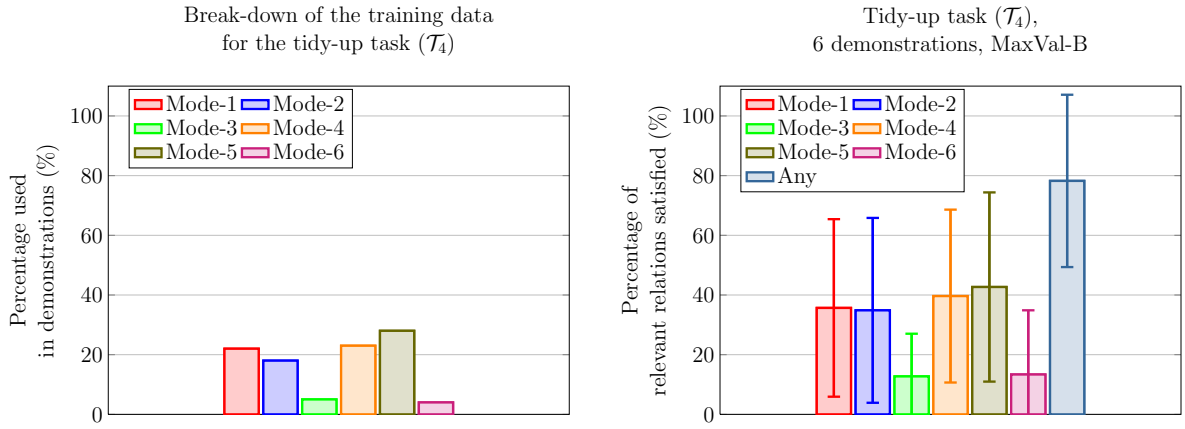


Figure 5.17.: Left: the distribution of modes demonstrated by the teacher for the tidy-up task ($\mathcal{T}_4$). Modes 1-6 correspond to different ways of solving this task as shown in Figure 5.9. We provided our approach with random sets of six demonstrations for learning the task, which contained examples from all modes according to this distribution. Right: the percentage of relevant relations demonstrated in each mode that were satisfied by our approach when solving the task starting in random initial states. Our approach is able to combine aspects from all modes when solving the task. We improvised goal states that satisfy the different modes to different degrees that reflect how often they appeared in the demonstrations. When considering relevant relations taken from any of these modes, we are able to satisfy 78.2% of the relations on average using MaxVal-B.

Moreover, we present the results for the tidy-up task ($\mathcal{T}_4$) using MaxVal-B in Figure 5.14-bottom right. Our approach is able to solve the task starting in the previously-unseen initial states to satisfy between 62.2% (10 demonstrations) and 86.6% (2 demonstrations) of the relevant spatial relations between the objects. Figure 5.16 shows examples of solutions computed by our method for $\mathcal{T}_4$ in this experiment. On average, we computed plans that range in length between 10.1 and 16.1 actions using ten and two demonstrations, respectively. Note that, as in the previous tasks, increasing the number of task (and hence action) demonstrations results in a larger branching factor during node expansion. This results in a slower convergence rate when using more demonstrations, and thus in satisfying less relations given the same number of iterations. In Section 5.7.7, we show how updating the action models by practicing the task results in mitigating this effect.

Since the teacher demonstrated six modes of solving the tidy-up task (see Figure 5.9), we additionally investigated how this is reflected in the computed task solutions using our approach. Figure 5.17-left shows the distribution of the different modes demonstrated

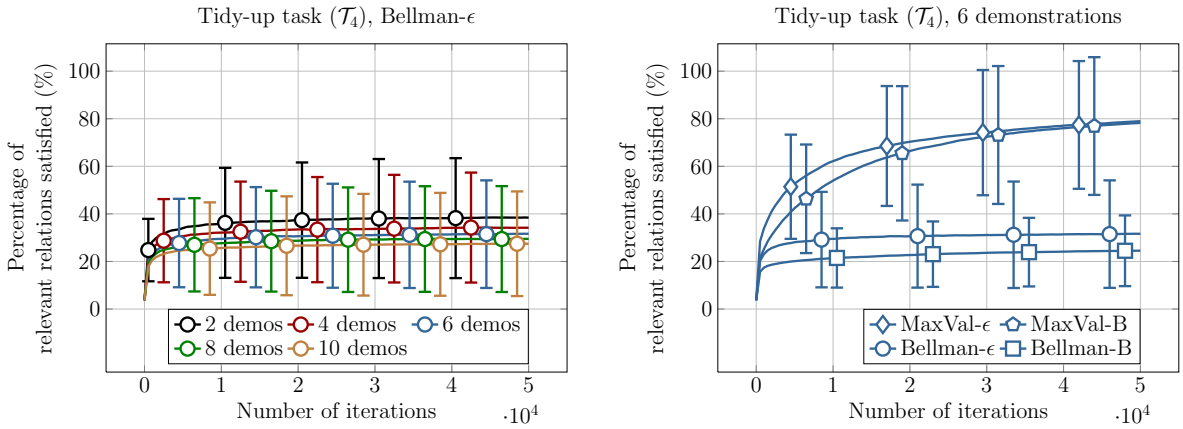


Figure 5.18.: Left: the percentage of satisfied relations for the tidy-up task $\mathcal{T}_4$ when using Bellman updates and an ϵ-greedy exploration criterion. Right: a comparison of max-value and Bellman value update strategies for this task when learning using six demonstrations. In this experiment, we used the action models learned from the teacher demonstrations and without allowing our approach to update those models by practicing the task. Max-value updates are more robust to noise (ambiguity) in the action goal distributions, allowing our approach to satisfy 78.2% of the relevant relations on average. On the other hand, Bellman updates result in satisfying not more than 31.6% of the relevant relations on average since this strategy assumes accurate action goal distributions.

by the teacher, and which were used to train the action and intention likelihood models. Accordingly, Figure 5.17-right shows the percentage of spatial relations specific to each mode that were satisfied in the goal states computed by our approach when learning from six demonstrations. Since we sampled demonstrations according to the distribution in Figure 5.17-left, our solutions satisfied on average 42.7% and 39.6% of the relations in modes 5 and 4, respectively, which were demonstrated more often by the teacher. Moreover, since our approach learned from a mixture of all modes, we improvised solutions that combine aspects from all of them, resulting in satisfying 78.2% of the relations considered relevant for the task on average.

Finally, we repeated the above experiment for the tidy-up task using Bellman updates and an ϵ-greedy selection criterion. The results are shown in Figure 5.18-left. In this case, our approach is able to satisfy only 38.4% of the relevant relations for the task using two demonstrations. Our approach computed plans that range between 4.3 actions (ten demonstrations) and 6.7 actions (two demonstrations) on average. Recall that in this experiment, we used the action models that have been learned from the demonstrations and without allowing our approach to practice the task in order to update these models.

Since Bellman updates aim to maximize the expected value of the intention likelihood, this assumes action models whose goal distributions accurately reflect the intention of the teacher when applying an action. Therefore, max-value updates are more robust to noise in the action models as they use the action distributions to guide the tree search but not for estimating the achieved values. To illustrate this, Figure 5.18-right shows the results for the tidy-up task using both max-value and Bellman updates for the case of six demonstrations. In Section 5.7.7, we investigate how the performance of our approach using Bellman updates significantly improves when allowing the robot to update the action goal distributions from its experience of solving the task.

5.7.6. Updating Action Models from New Experiences

In this experiment, we qualitatively investigate our approach for updating the learned action models using the robot's experience in solving the task. In the previous experiment, we used our approach with max-value updates to compute plans that generalize the teacher demonstrations to solve the learned tasks starting in arbitrary initial states. In each case, our method computed plans driven by maximizing the intention likelihood of the task. Using the approach we propose in Section 5.6, we used these plans as virtual demonstrations to disambiguate the intention of each action and update the action goal distributions learned from the teacher. Specifically, we updated the relevance of each action template and the corresponding voting distribution to reflect the usefulness of each template and vote for solving the task, respectively.

Figure 5.19 shows template selection rates for two actions from the tidy-up task ($\mathcal{T}_4$): moving the red box (top) and moving the pack of sugar (bottom). In each case, our approach is able to leverage different templates to achieve desirable relations between the objects. When moving the red box, our approach relied mostly on the templates representing the table (38.7% of the time), the blue box (30.7% of the time), and the green box (15.7% of the time). When moving the pack of sugar, our approach relied on the template representing the table in 49.5% of the cases since the pairwise relation between those objects is relevant for the goal of this task. Additionally, our approach used templates corresponding to the other stationary objects between 7.9% and 13.5% of the time for the cup and bowl, respectively. This highlights the importance of considering multiple interpretations of each action based on all stationary objects. Depending on the order of manipulating the objects, different templates can be useful when voting for the goal poses of the moving objects.

Finally, Figure 5.20 shows an example for updating the action of moving the green block in the aligning blocks task ($\mathcal{T}_2$). Here, our approach learns a distribution that corresponds to high probabilities for aligning the green block with the other blocks in the scene as demonstrated by the teacher. Thus, our approach disambiguated the demonstrations by revealing the intention of the teacher for this action. We refer

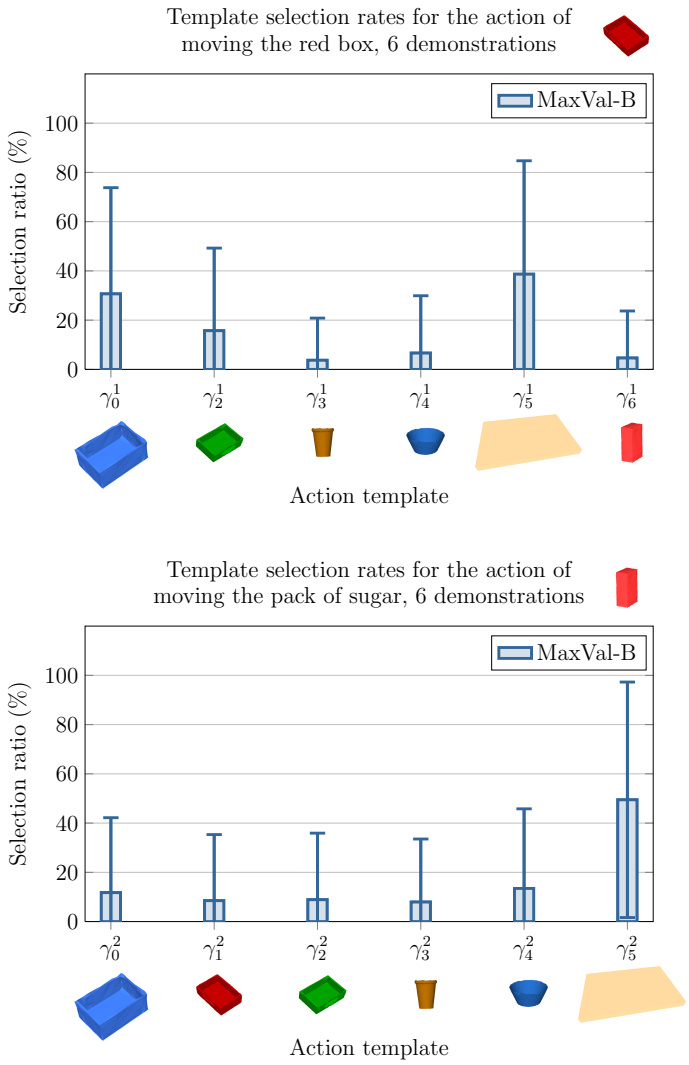


Figure 5.19.: Average action template selection rates for two actions in the tidy-up task ($\mathcal{T}_4$). We extracted these statistics by solving the task using our approach starting in 2,500 random initial states. Rather than adopting a fixed model for each action, our method enables leveraging multiple templates when applying an action based on the current state. When moving the red box (top), our method relied on the templates corresponding to the table and other boxes most of the time. Similarly, for moving the pack of sugar (bottom), our method relied on the table in 49.5% of the time, since this is conducive to satisfying desirable relations with the table and other kitchen-related objects (see Figure 5.9). Our approach uses these results to update the relevance of each template and hence the action goal distributions without requiring additional teacher demonstrations.

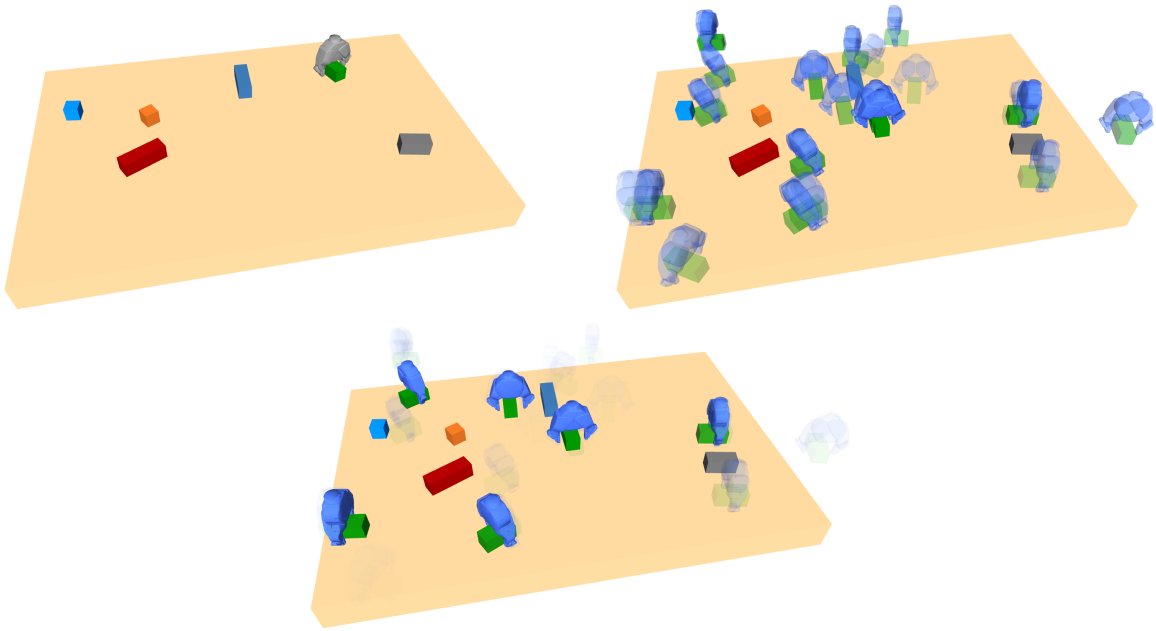


Figure 5.20.: An example of the action for moving the end-effector and the green block in the aligning blocks task ($\mathcal{T}_2$). Top left: a random initial state. Top right: the corresponding distribution of goal states. The teacher demonstrations result in ambiguity with respect to the likely goal poses of the green block relative to the other objects. Bottom: after practicing the task in simulation, we update this distribution to reflect the intention of the teacher more accurately, i.e., we are more likely to sample goal states that align the green block with other blocks or that place it in a desirable pose relative to the table, see Figure 5.7.

the reader to Appendix A.1 for additional examples of updating action models in the breakfast setting and tower building tasks ($\mathcal{T}_1$ and $\mathcal{T}_3$).

5.7.7. Solving Tasks Using the Updated Action Models

The purpose of this experiment is to evaluate our approach for solving the learned tasks when using the action models that the robot updated using its experience. For this, we repeated the experiment in Section 5.7.5 but used the action models we updated as in Section 5.7.6 above. As before, we did not provide our algorithm with a specific goal state to achieve in each case. For all tasks, we provided our algorithm with a budget of 50,000 iterations and analyzed the solutions extensively every 500 iterations. We demonstrate the improved performance of our approach for both max-value and Bellman criteria when using the updated action goal distributions compared to using the noisy models before practicing the task.

5.7.7.1. Results for the Breakfast Setting Task ($\mathcal{T}_1$)

We evaluated our approach for solving the breakfast setting task ($\mathcal{T}_1$) starting in 100 random initial states using the updated action models. We repeated this 25 times using different sets of demonstrations in each case. We show the results using max-value updates with a Boltzmann exploration strategy (MaxVal-B*) and using Bellman value updates with an ϵ-greedy exploration strategy (Bellman-ϵ*) in Figure 5.21-top left and Figure 5.21-top right, respectively. For both variants, our approach is able to satisfy between 98.5% and 100% of the relevant relations on average. Figure 5.21-bottom left also compares our approach using these two variants (for the case of six demonstrations) to the corresponding results in Section 5.7.5 for MaxVal-B, i.e., before updating the action models. Using the updated models, MaxVal-B* and Bellman-ϵ* are able to satisfy 99.5% and 95.5% of the relations after only 500 iterations, respectively. The updated action goal distributions more accurately reflect the intention of each action in the task, which results in sampling more promising goal states during the tree search. In contrast, MaxVal-B requires 3000 iterations before satisfying 90.6% of the relations.

Finally, Figure 5.21-bottom right shows the corresponding plan lengths computed by those three variants of our approach. MaxVal-B requires 8.14 actions to satisfy 96.8% of the relations on average. On the other hand, using the updated action models, MaxVal-B* and Bellman-ϵ* computed plans consisting of only 6.52 and 6.15 actions, and satisfying 100% and 99.2% of the relations, respectively. This demonstrates the ability of our approach to leverage the experience from solving the task to improve its performance with respect to both achieving desired object arrangements and the cost of the solution.

5.7.7.2. Results for the Aligning Blocks and Tower Building Tasks ($\mathcal{T}_2$-$\mathcal{T}_3$)

As for $\mathcal{T}_1$, we evaluated our approach when solving tasks $\mathcal{T}_2$ and $\mathcal{T}_3$ using the updated action models. We refer the reader to Appendix A.2 for the detailed results. After practicing those tasks, our approach improved with respect to the percentage of relevant relations satisfied in the final state and the efficiency of the solutions. In the following, we present a detailed evaluation of our approach for $\mathcal{T}_4$ as an interesting complex task involving both relevant and irrelevant relations as well as multiple solution modes.

5.7.7.3. Results for the Tidy-Up Task ($\mathcal{T}_4$)

We evaluated our approach for solving the tidy-up task ($\mathcal{T}_4$) starting in 100 random initial states using the updated action models. We repeated this 25 times using different sets of demonstrations in each case. Figure 5.22 shows the results when using MaxVal-B* (top-left), MaxVal-B* (0.5) (top-right), and Bellman-ϵ* (bottom-left). Using MaxVal-B*, our approach achieved between 79.1%-88.4% of the relevant relations on average over all

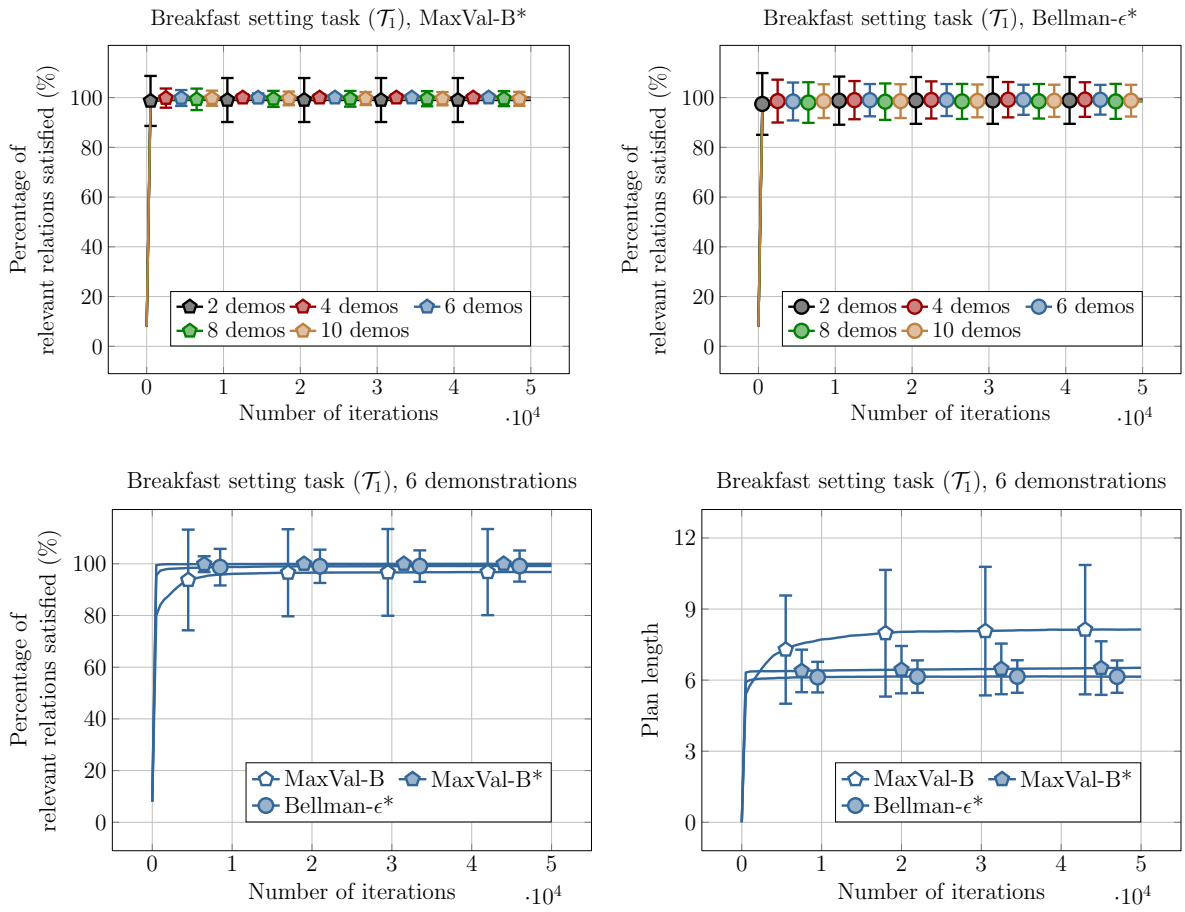


Figure 5.21.: Results of our approach for solving the breakfast setting task ($\mathcal{T}_1$) using the updated action models learned after practicing the task. Top: MaxVal-B* (left) and Bellman-ϵ* (right) are able to satisfy 99.5%-100% and 98.5%-99.2% of the relevant relations on average. Bottom left: Using the updated action models, MaxVal-B* and Bellman-ϵ* are able to satisfy 99.5% and 95.5% of the relations after only 500 iterations, respectively, when learning from six task demonstrations. In contrast, MaxVal-B requires 3000 iterations before satisfying 90.6% of the relations. Bottom right: MaxVal-B requires 8.14 actions to satisfy 96.8% of the relations on average. Using the updated action models, MaxVal-B* and Bellman-ϵ* required only 6.52 and 6.15 actions on average to satisfying 100% and 99.2% of the relations, respectively. Our approach leverages the experience from solving the task to improve its performance with respect to both achieving desired object arrangements and the cost of the solution.

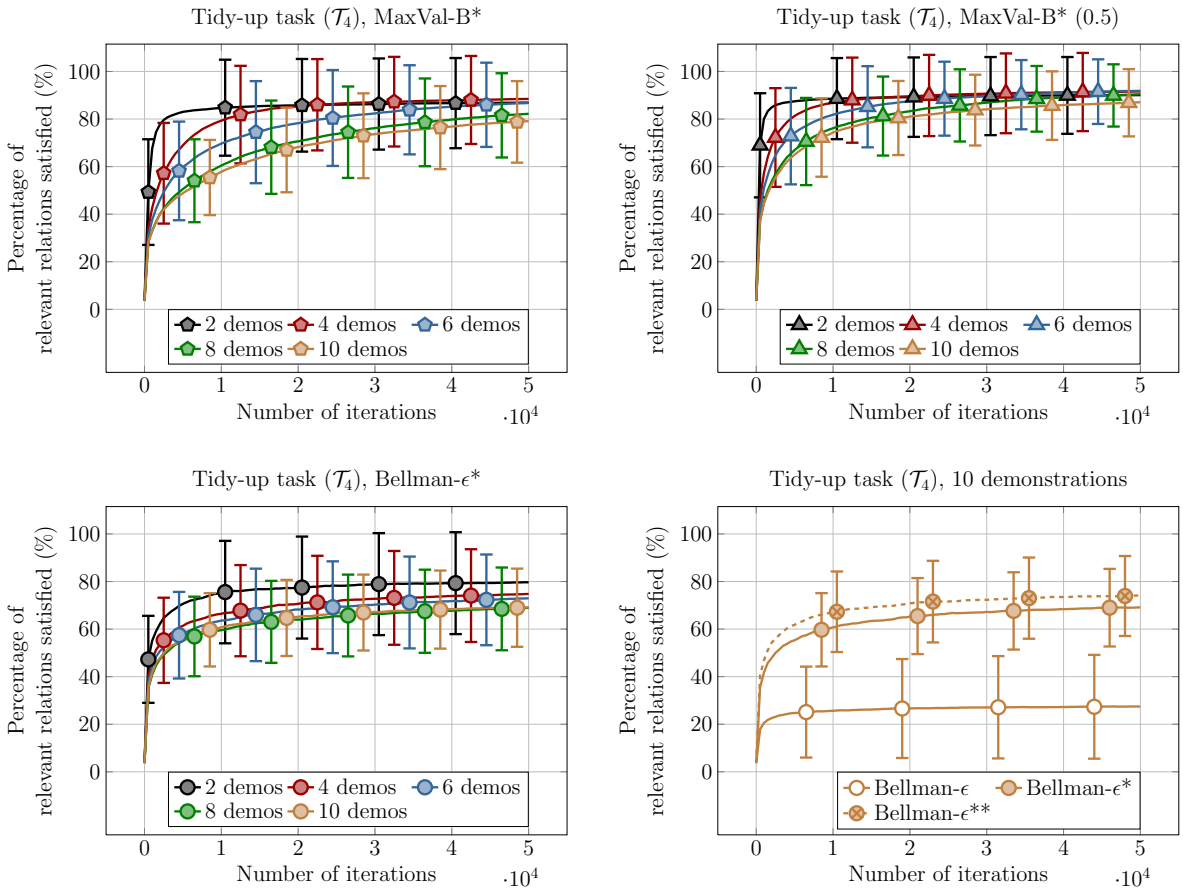


Figure 5.22.: Results for solving the tidy-up task ($\mathcal{T}_4$) after practicing the task and updating the action goal distributions. Top left: MaxVal-B* achieved 79.1%-88.4% of the relevant relations on average over all numbers of demonstrations, compared to 62.2%-86.6% for MaxVal-B before practicing the task (see Figure 5.14-bottom right for comparison). Top right: even when sampling only half the templates for each action, MaxVal-B* (0.5) achieved 87.1%-91.9%. The flexibility of our approach enables MaxVal-B* (0.5) to rely on a subset of the interpretations for each action to solve the task. Bottom left: Bellman-ϵ* achieved a significantly better performance compared to Bellman-ϵ before practicing the task (see Figure 5.18-left). Bottom right: results for ten demonstrations when using the noisy action models from the demonstrations (Bellman-ϵ), the updated models after practicing on 100 cases (Bellman-ϵ*), and the updated models after practicing further on 100 cases (Bellman-ϵ**). These variants satisfied 27.4%, 69.1%, and 74.1% of the relevant relations on average, respectively. Updating the action models is particularly beneficial when using Bellman updates, as this strategy incorporates action goal probabilities in the value estimates.

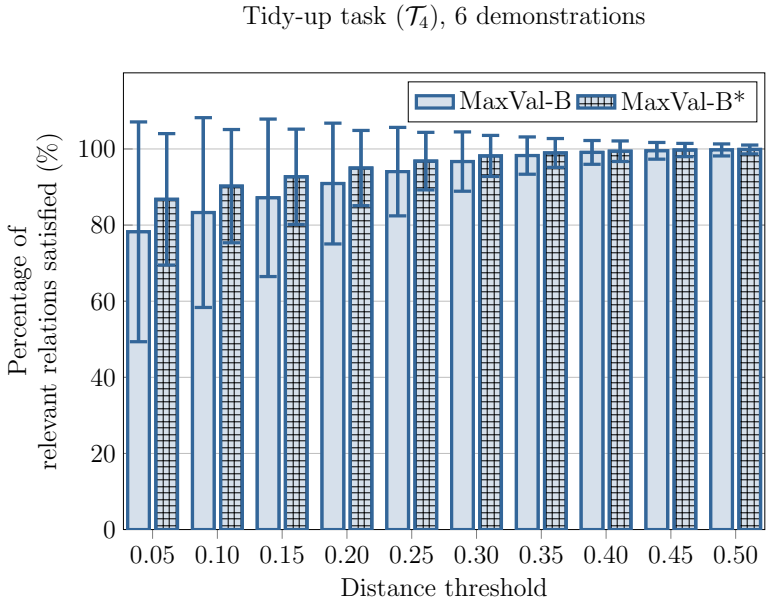


Figure 5.23.: Results for MaxVal-B and MaxVal-B* when solving the tidy-up task ($\mathcal{T}_4$) using six demonstrations and a discretization threshold of $\delta_{\max} = 0.05$ for the action models. We compared these variants with respect to how close the computed configurations are to the demonstrated ones. We incrementally increased the threshold for the maximum allowed distance between computed and demonstrated poses to consider a relation to be satisfied. Practicing the task resulted in satisfying more relations by MaxVal-B* compared to MaxVal-B for the same distance threshold, i.e., MaxVal-B* achieves a finer reproduction of the demonstrations.

numbers of demonstrations, compared to 62.2%-86.6% for MaxVal-B before practicing the task (see Figure 5.14-bottom right for comparison). The updated action models guided the search to more promising regions of the state space, achieving better solutions using less iterations. Practicing the task also resulted in a similar performance for MaxVal-B* for all numbers of demonstrations. On the other hand, the performance of MaxVal-B deteriorates with more demonstrations due to the increased branching factor. Figure 5.23 highlights the difference between MaxVal-B and MaxVal-B* (given six demonstrations) with respect to how close the computed configurations are to the demonstrated ones. For both variants, we used a discretization threshold of $\delta_{\max} = 0.05$ for the action goal distributions. When evaluating the solutions, we incrementally increased the threshold for the maximum allowed distance between computed and demonstrated poses to consider a relation to be satisfied. MaxVal-B* achieved a finer reproduction of the demonstrations for the same distance threshold. With a strict threshold of 0.05, MaxVal-B* satisfied 86.8% of the relevant relations on average

compared to 78.2% for MaxVal-B. When relaxing this threshold to 0.10, MaxVal-B* achieved a mean score of 90.2%. On the other hand, MaxVal-B achieves 91.0% only when considering a looser threshold of 0.20.

Furthermore, even when relying on only half the templates for each action, MaxVal-B* (0.5) achieved 87.1%-91.9% of the relevant relations (Figure 5.22-top right). Due to the reduced branching factor, it achieves this at a faster convergence rate than that of MaxVal-B*. This highlights the flexibility of our approach in leveraging different interpretations of each action when solving the task. This also demonstrates the advantage of practicing the task to update the relevance of action templates, as our approach is able to sample a subset of the templates based on how relevant they are for solving the task (Figure 5.19).

Moreover, our approach using Bellman-ϵ* achieved 68.9%-79.7% of the relevant relations on average (Figure 5.22-bottom left). Using the updated action goal distributions, Bellman-ϵ* achieved a significantly better performance compared to Bellman-ϵ before practicing the task, which only satisfied 27.4%-38.4% of the relevant relations (see Figure 5.18-left). Figure 5.22-bottom right highlights this for the case of ten demonstrations for three variants: using the noisy action models from the demonstrations (Bellman-ϵ), using the updated models after practicing on 100 cases (Bellman-ϵ*), and using the updated models after practicing further on 100 cases (Bellman-ϵ**). These variants are respectively able to achieve 27.4%, 69.1%, and 74.1% of the relevant relations on average. Practicing the task results in disambiguating the demonstrations by revealing the intended goals of each action. This is particularly beneficial when using Bellman updates, as this strategy incorporates action goal probabilities in the value estimates.

Additionally, we investigated the solution costs corresponding to the above variants, which we report in Figure 5.24. Before practicing the task, MaxVal-B computed plans that are between 10.1 (ten demonstrations) and 16.1 (two demonstrations) actions long on average (Figure 5.24-top left). After updating the action models, MaxVal-B* computed more efficient solutions that are between 10.9 (ten demonstrations) 12.8 actions long (two demonstrations) on average (Figure 5.24-top right). Whereas sampling only half the templates enabled MaxVal-B* (0.5) to achieve faster convergence rates, this also resulted in having to explore longer plans that are between 12.3 (ten demonstrations) and 13.3 (two demonstrations) actions long on average to solve the task (Figure 5.24-bottom left). Finally, by considering the goal probability of each action, Bellman-ϵ* computed more efficient plans consisting of 10.0 (ten demonstrations) to 11.3 (two demonstrations) actions on average (Figure 5.24-bottom right).

5.7.7.4. A Closer Look at Max-Value vs. Bellman Updates

In this section, we highlight the differences between the max-value and Bellman value update strategies of our approach. In addition to the percentage of relevant relations

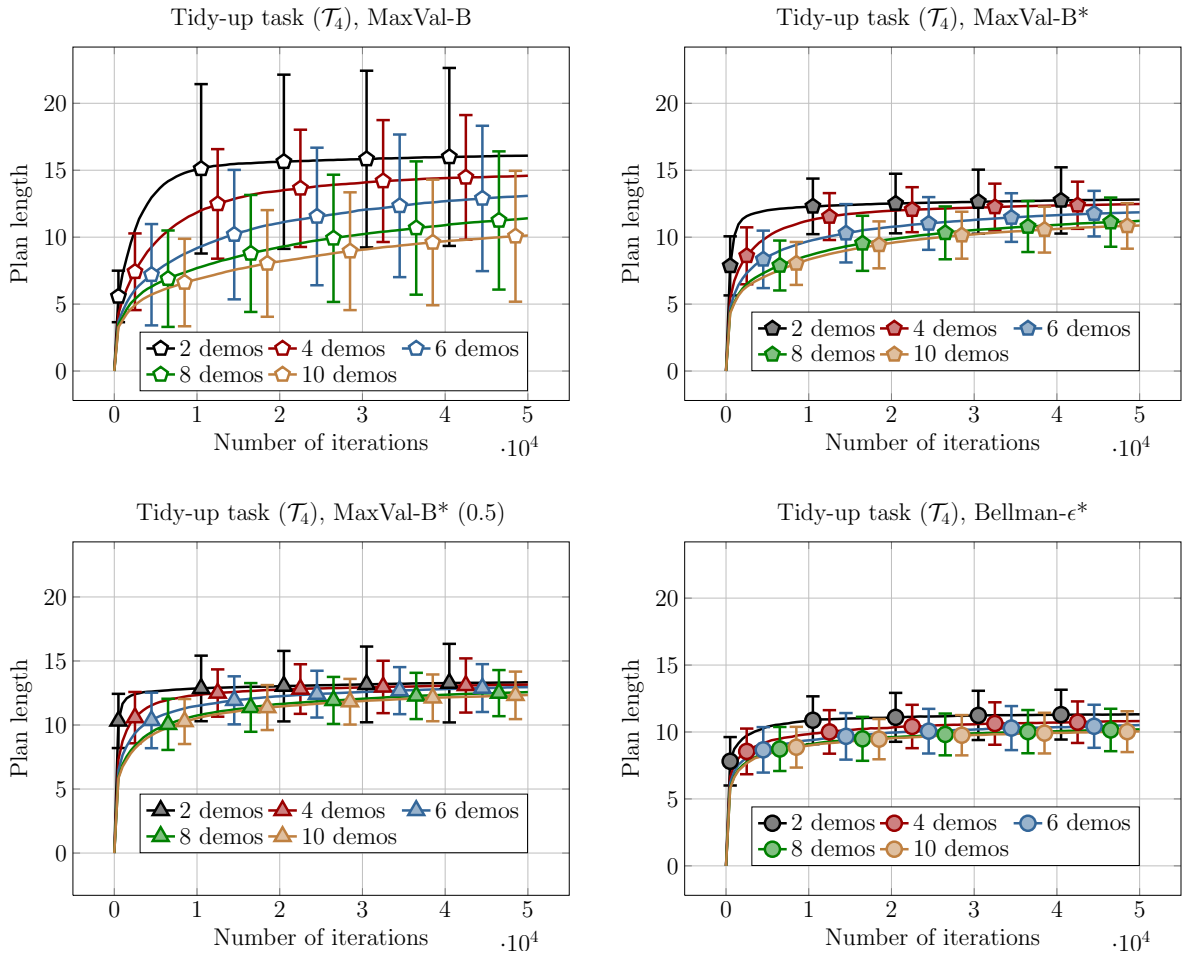


Figure 5.24.: Plan lengths computed by different variants of our approach for the tidy-up task ($\mathcal{T}_4$). Top left: before practicing the task, MaxVal-B computed plans consisting of 10.1-16.1 actions on average. Top right: after practicing, MaxVal-B* computed more efficient solutions consisting of 10.9-12.8 actions on average. Bottom left: by sampling only half the templates for each action, MaxVal-B* (0.5) achieved a faster convergence rate but at the expense of having to explore longer sequences of 12.3-13.3 actions on average. Bottom right: a Bellman-based strategy jointly considers the probability of each intermediate state and the intention likelihood of the final state, resulting in more efficient plans by Bellman-ϵ* (10.0-11.3 actions) compared to the max-value strategies.

satisfied and the costs of the computed plans, we shed light on the ability of those strategies to generalize the demonstrations by inspecting their tendency to satisfy *irrelevant* relations.

For this, we evaluated all variants of our algorithm (Table 5.1) when solving the

tidy-up task ($\mathcal{T}_4$) after learning from six demonstrations. As in the previous experiments, we report the results for solving the task starting in 100 random initial states, which we repeated 25 times using different sets of demonstrations in each case. Figure 5.25 shows the percentage of relevant relations satisfied in the final goal states. We report results for the max-value and Bellman strategies in the top and bottom row, respectively. We tested these strategies in combination with the Boltzmann (left) and ϵ-greedy (right) exploration strategies.

When relying on the noisy action models, i.e., before practicing the task, we observe a clear advantage of using max-value updates (MaxVal-B, MaxVal-ϵ) over Bellman updates (Bellman-B, Bellman-ϵ). With a budget of 50,000 iterations, max-value variants satisfied up to 79.0% of the relevant relations (MaxVal-ϵ) as opposed to 31.6% using Bellman updates (Bellman-ϵ). This highlights the degree of ambiguity in the demonstrations, resulting in action goal distributions that do not accurately reflect the intention of the teacher. Whereas max-value variants rely on these models to sample successor states during the search, they do not consider the probabilities of these states in the node value estimates. Accordingly, they are more robust to noise in the action models and are able to greedily optimize with respect to the intention likelihood of the final state. In contrast to this, the goal of the Bellman-based strategies is to maximize the *expected* value of the intention likelihood, and are therefore greatly affected by the noise in the action goal distributions.

After practicing the task on 100 cases and updating the action models accordingly, we observed an improvement for both value update strategies. The new action goal distributions capture the relevance of each template and vote more accurately compared to the prior models. This enables our algorithm to sample more useful states during the search, i.e., those that contribute to desirable final object arrangements. Accordingly, MaxVal-ϵ* achieved 93.4% of the relevant relations. More importantly, the updated goal state probabilities resulted in a clear improvement when using Bellman updates as Bellman-ϵ* satisfied 73.0% of the relevant relations (compared to 31.6% for Bellman-ϵ).

Additionally, relying on only half the action templates resulted in a slight advantage for MaxVal-B* (0.5), MaxVal-ϵ* (0.5), Bellman-B* (0.5), and Bellman-ϵ* (0.5) over their counterparts (MaxVal-B*, MaxVal-ϵ*, Bellman-B*, and Bellman-ϵ*, respectively) that consider all templates. This is due to the reduced branching factor when sampling less action templates resulting in satisfying more relations in less iterations. Furthermore, with respect to the exploration strategies, we observed on average a better performance for all variants when using ϵ-greedy exploration compared to Boltzmann exploration.

Overall, when inspecting the percentage of relevant relations satisfied in the final goal state only, we observed a better performance for max-value variants compared to using Bellman updates. This makes sense when considering the objective functions that these variants optimize. Intuitively, having selected a node to expand, a max-value approach asks the following question: "*How can I apply actions to satisfy more relations between*

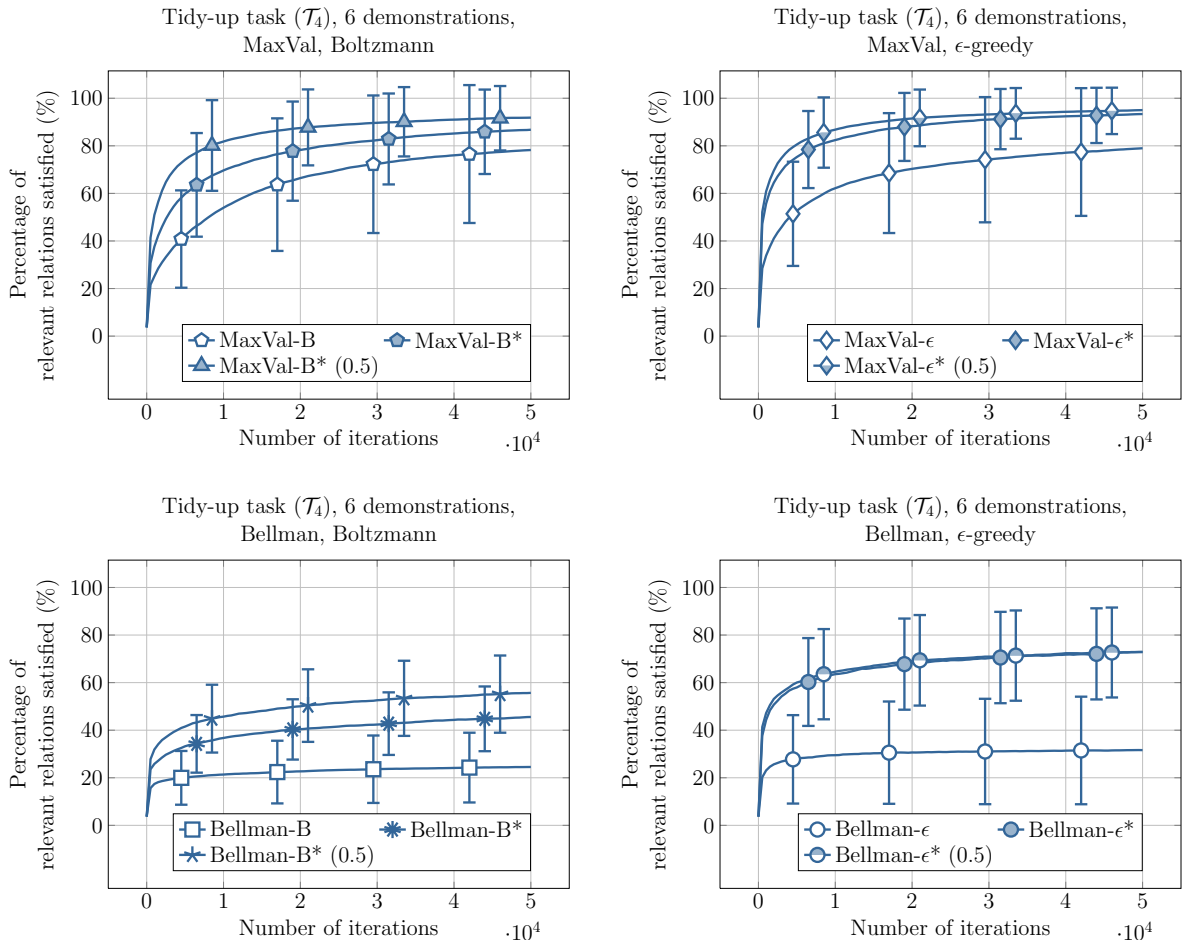


Figure 5.25.: Results of different variants of our algorithm for solving the tidy-up task ($\mathcal{T}_4$) after learning from six demonstrations. Before practicing the task, we observe a clear advantage of using max-value updates (MaxVal-B, MaxVal-ϵ) over Bellman updates (Bellman-B, Bellman-ϵ) since max-value variants greedily maximize the intention likelihood of the final state. In contrast, Bellman variants consider the probability of each action goal state and are thus more sensitive to noise in the action models. For both value update strategies, we observe an improvement in performance after practicing the task to update the action models. Whereas Bellman variants achieve less relevant relations in the final states, they tackle the harder problem of inferring the intention of the teacher for each step of the plan. Accordingly, they compute more efficient plans that satisfy less irrelevant relations compared to max-value strategies, see Figures 5.26 and 5.27.

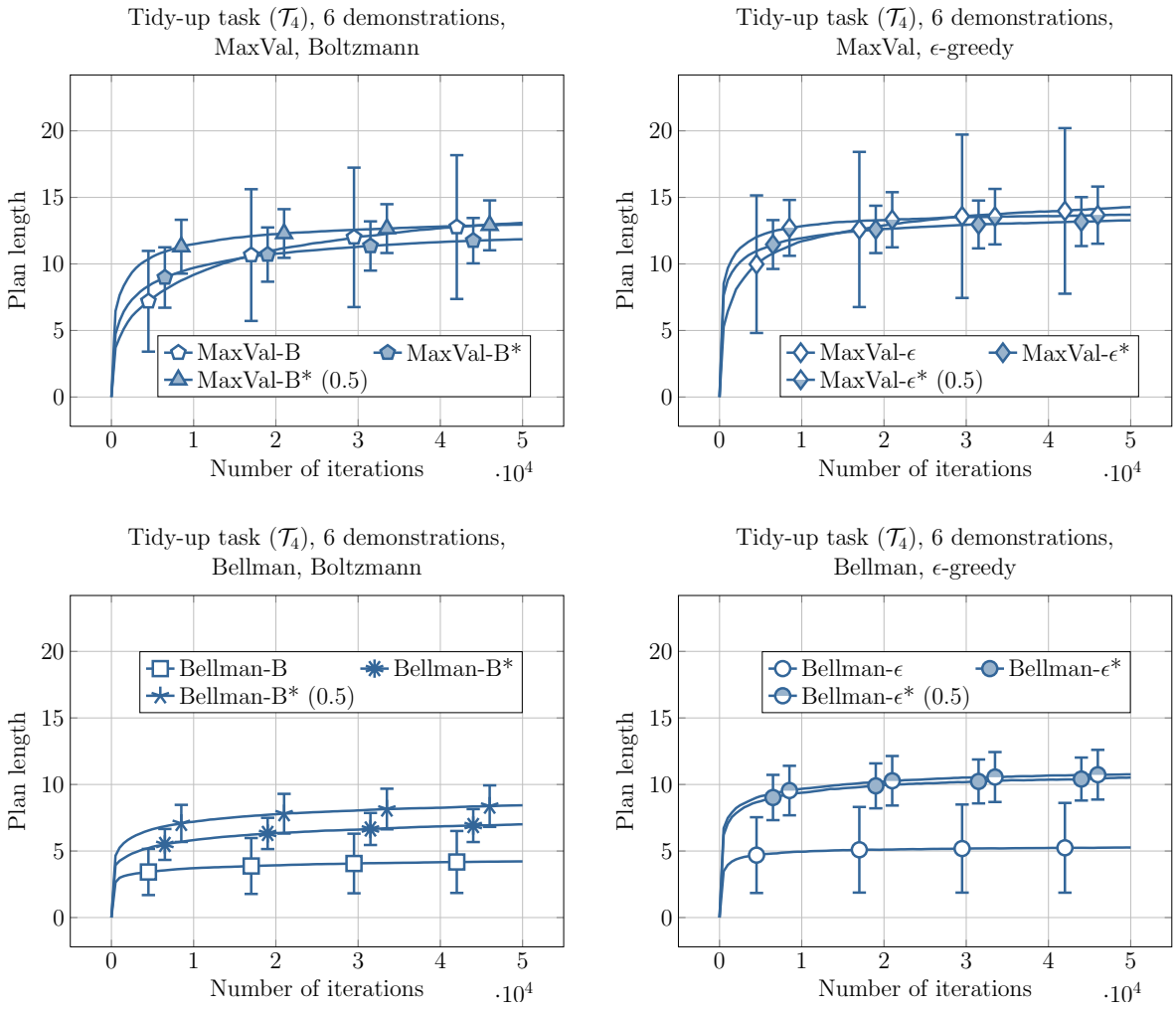


Figure 5.26.: Average plan lengths for the solutions computed by different variants of our algorithm for solving the tidy-up task ($\mathcal{T}_4$) after learning from six demonstrations. Before practicing the task, Bellman variants (bottom) are affected by noise in the action models and are unable to compute plans that are longer than 4.2-5.3 actions on average (i.e., reaching for and moving 2-3 objects). After practicing the task, Bellman-ϵ* required 10.5 actions on average to satisfy 73.0% of the relevant relations. Practicing the task also enabled max-value variants to satisfy more relevant relations (Figure 5.25-top) using less actions given the same number of iterations. By considering the probability of each intermediate state, Bellman variants compute more efficient solutions on average than their max-value counterparts, which only consider the intention likelihood of the final state, and are hence likely to apply more actions to satisfy even irrelevant relations, see Figure 5.27. Finally, sampling only 50% of the templates during search leads to a faster convergence rate but at the expense of computing longer plans.

the objects?" On the other hand, a Bellman strategy intuitively asks: "*Given the current arrangement, which moves would the teacher be more likely to make to satisfy more relations between the objects?*" Clearly, the latter problem is harder as it considers the intention of the teacher for each step of the solution, and not only for the final state.

This difference between the strategies is also reflected in the costs of their respective solutions as shown in Figure 5.26. Max-value variants computed longer sequences of actions on average with the same number of iterations. For example, MaxVal-ϵ* required 13.3 actions on average to satisfy 93.4% of the relevant relations. On the other hand, Bellman-ϵ* required 10.5 actions on average to satisfy 73.0% of the relevant relations. This translates to approximately three unsatisfied relations out of the 12 relevant ones for this task, i.e., the robot would need to reach for and move one more object to satisfy all relevant relations. In general, by considering the probability of each intermediate state, Bellman-based variants compute more efficient solutions than their max-value counterparts, which only consider the intention likelihood of the final state, and are hence likely to apply more actions.

This becomes more evident when we shed light on these variants with respect to their tendency to satisfy *irrelevant* relations. Our approach does not leverage negative training data, e.g., configurations (relations) that are labeled wrong (irrelevant) by the teacher. However, this information is implicitly encoded in the demonstrated actions in the sense that the teacher typically moves objects such that they achieve desirable relations with (a subset of) the stationary objects in the scene.

We illustrate this in Figure 5.27-top, which shows the frequency of satisfying relevant (blue) and irrelevant (red) relations by MaxVal-ϵ* and Bellman-ϵ* in this experiment. This task involves twelve relevant and nine irrelevant relations in total. As MaxVal-ϵ* does not consider goal probabilities in its value estimates, it greedily tends to satisfy both relevant and irrelevant relations to maximize the intention likelihood of the final state. Thus, it satisfied six irrelevant relations in 80%-90% of the cases and the other three in 90%-100% of the cases. In contrast to that, Bellman-ϵ* satisfied irrelevant relations in 40%-70% of the cases only. In other words, it is better at capturing the intention of the teacher with respect to the actions and hence at generalizing the demonstrations to improvise arrangements in which irrelevant relations are unsatisfied. In contrast to this, MaxVal-ϵ* tends to "copy" the demonstrated arrangements.

Finally, we repeated this experiment using action models we manually designed to reflect the ground truth intention of the teacher with respect to the template relevance weights and the voting distributions based on six task demonstrations. We refined these models by practicing the task using a ground truth intention likelihood model as well, i.e., with a weight of zero for irrelevant relations. Using the resulting action models, we solved the task using MaxVal-ϵ* and Bellman-ϵ* starting in 100 random states (given 75,000 iterations), but this time using the intention likelihood model learned using our approach, i.e., with the relation weights estimated from the six demonstrations

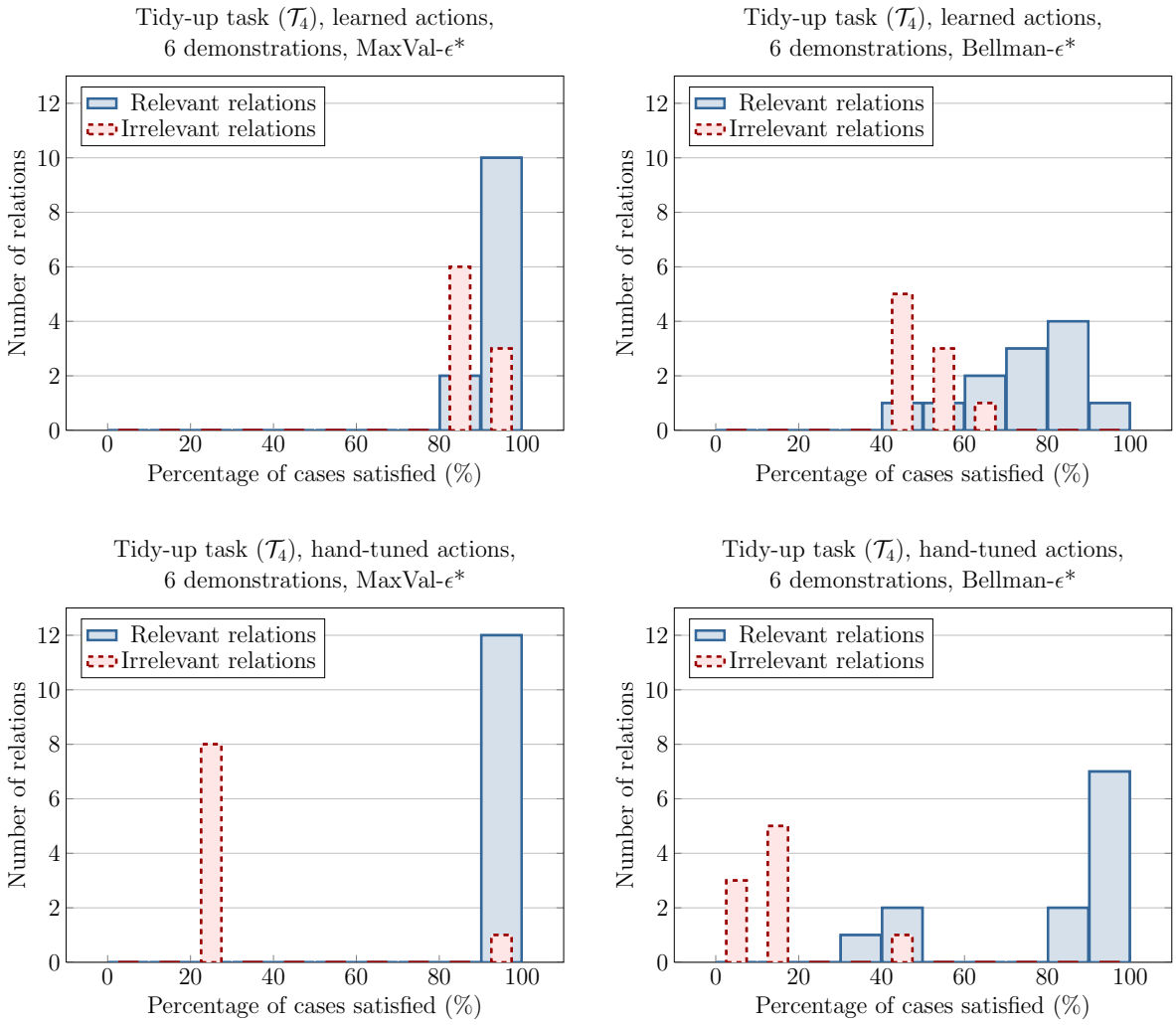


Figure 5.27.: Top: the frequency of satisfying relevant and irrelevant relations when solving 2,500 cases of the tidy-up task ($\mathcal{T}_4$) using MaxVal-ϵ* (left) and Bellman-ϵ* (right). This task involves twelve relevant and nine irrelevant relations in total. By greedily optimizing the intention likelihood of the final state, max-value variants tend to satisfy irrelevant relations in addition to relevant ones. In contrast, Bellman variants capture the intention of the teacher for each applied action and thus focus on satisfying relevant relations. Bottom: results for 100 cases after solving the task (given 75,000 iterations) using action models that we designed to satisfy relevant relations only. Even then, MaxVal-ϵ* applied 12.7 actions on average to satisfy most irrelevant relations in 20%-30% of the cases (as well as all relevant ones in 90%-100% of the cases). On the other hand, by considering the action goal probabilities, Bellman-ϵ* applied 10.8 actions on average and focused on reproducing the relevant relations (81% on average) while satisfying most irrelevant ones in only 0%-20% of the cases.

only. We show the resulting distribution of satisfied relations in Figure 5.27-bottom. Even with action models that have been designed to reproduce relevant relations only, MaxVal-ϵ* satisfied most irrelevant relations in 20%-30% of the cases. On the other hand, by considering the action goal probabilities, Bellman-ϵ* focused on reproducing the relevant relations (81% on average) and satisfied most irrelevant relations in only 0%-20% of the cases.

5.7.8. Planning Time

In this section, we briefly comment on the planning time of our approach when solving tasks in the experiments in Sections 5.7.5 and 5.7.7 above. Although we did not focus on optimizing our implementation of the TI algorithm in this work, we shed some light on how the planning time varied in our experiments for the different tasks and algorithm variants. We report the planning times after 50,000 iterations for the breakfast setting and the tidy-up tasks ($\mathcal{T}_1$ and $\mathcal{T}_4$) in this section and refer the reader to Appendix A.3 for the results for tasks $\mathcal{T}_2$ and $\mathcal{T}_3$. We averaged all results over 2,500 test cases. We performed all experiments on a machine with an Intel i7-2700K CPU at 3.50 GHz.

As expected, we observed an increase in planning time as the number of demonstrations increased. This is mainly due to our data-driven models, which result in larger branching factors when expanding nodes and more time to evaluate the intention likelihood given more teacher demonstrations. For example, as shown in Figure 5.28, MaxVal-B required 13.95 sec and 18.98 sec to run 50,000 iterations for the breakfast setting task given two and ten demonstrations, respectively. As expected, the planning time also increased with the number of objects (and hence actions and action templates) in a task. For example, MaxVal-B required 32.58-42.39 sec (two-ten demonstrations) to run 50,000 iterations for the tidy-up ($\mathcal{T}_4$).

Moreover, Figure 5.29 shows the planning times for the tidy-up ($\mathcal{T}_4$) when learning from six demonstrations. In general, updating the action models after practicing the task led to an increase in planning time. For example, Bellman-ϵ required 36.97 sec on average to run 50,000 iterations, which increased to 51.41 sec for Bellman-ϵ*. As we learn more accurate action goal distributions, we are able to focus the search on more relevant regions of the state space, which results in constructing deeper search trees. Accordingly, our approach is able to sequence more actions and satisfy more relations given the same number of iterations compared to using the noisy action models.

Finally, Figure 5.29 also shows how to reduce the planning time by relying on a subset (50%) of the action templates. For example, MaxVal-B* (0.5) required 38.07 sec on average compared to 48.97 sec for MaxVal-B*. As explained in Section 5.7.7, this enables our approach to achieve faster convergence rates (i.e., more relations satisfied in less iterations) but at the expense of computing longer plans.

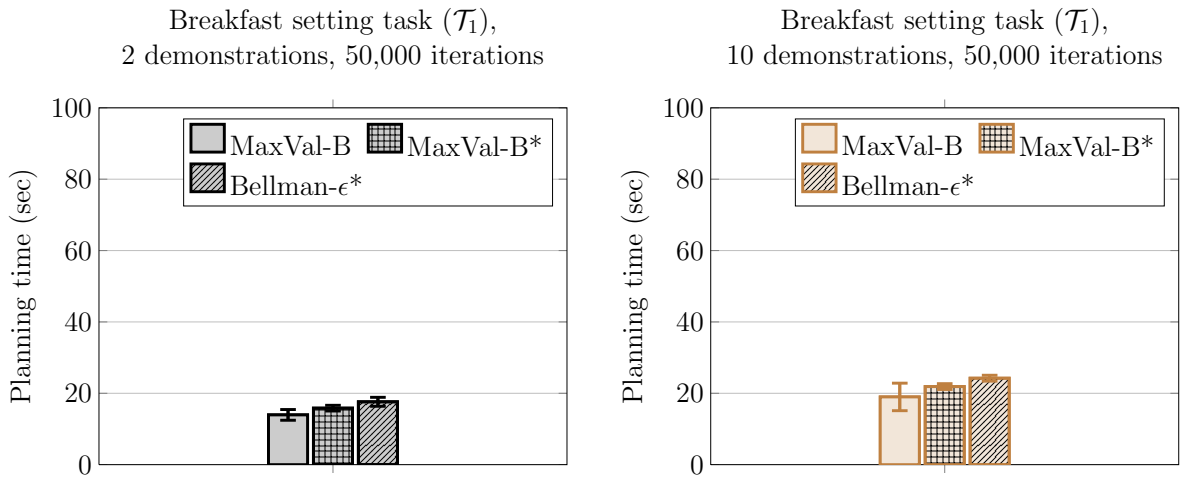


Figure 5.28.: The planning times of our approach after 50,000 iterations when solving the breakfast setting task ($\mathcal{T}_1$) given two (left) and ten (right) demonstrations (averaged over 2,500 cases). See Section 5.7.8.

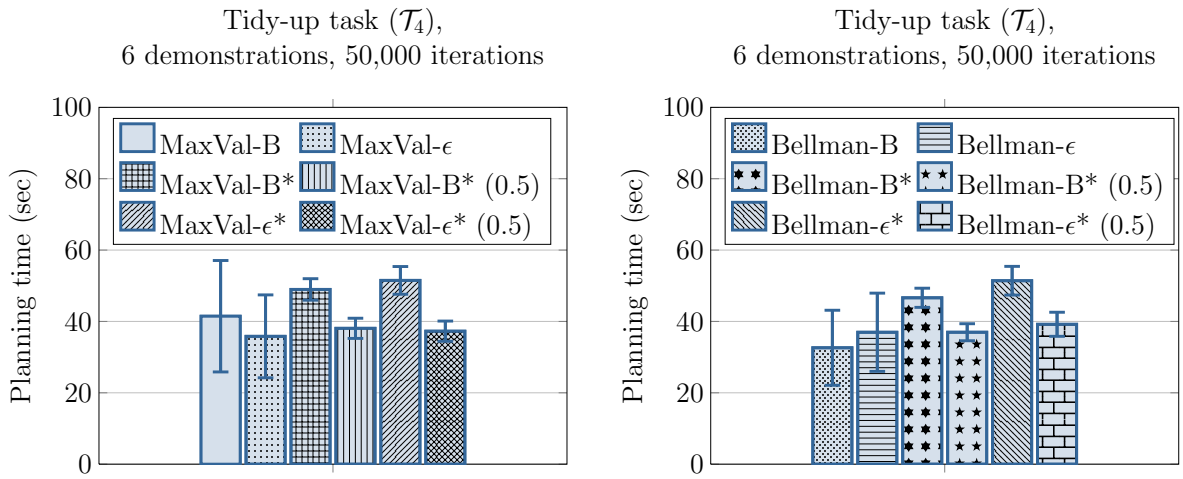


Figure 5.29.: The planning times of our approach after 50,000 iterations when solving the tidy-up task ($\mathcal{T}_4$) given six demonstrations (averaged over 2,500 cases). See Section 5.7.8.

5.7.9. The Advantage of Using Multiple Action Templates

The purpose of this experiment is to demonstrate the advantage of leveraging multiple interpretations for each action to tackle the ambiguity in the demonstrations. We demonstrate the following: *i)* by incorporating template-selection nodes, our proposed tree structure reduces the dimensionality of the problem by considering one action model at a time during planning, and *ii)* adopting multiple interpretations of an action tackles

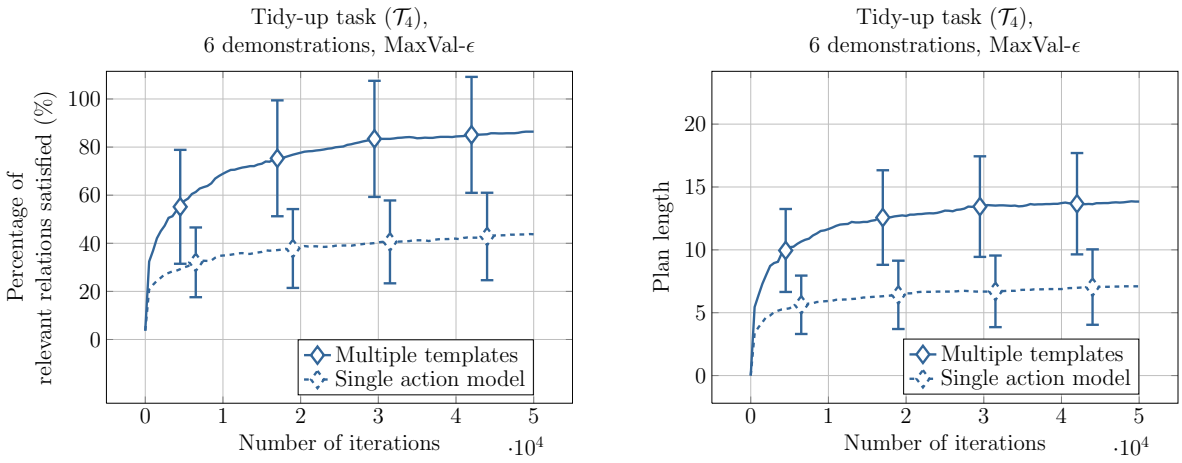


Figure 5.30.: We compared our approach (multiple templates) to using a single implicit model for each action based on one template that considers votes from all stationary objects. We show results when using six demonstrations to solve the tidy-up task ($\mathcal{T}_4$) in 100 random initial states. By considering several, simple action interpretations individually, our approach tackles the ambiguity in the action demonstrations while reducing the dimensionality of the problem, and thus achieves faster convergence rates compared to using a single, implicit model per action.

the ambiguity in the demonstrations by providing our algorithm with the flexibility to rely on different stationary objects when applying an action based on the current state.

Using six demonstrations for the tidy-up task ($\mathcal{T}_4$), we first used our approach (using MaxVal-ϵ) to solve the task in 100 random initial states. We compared this to a baseline that models each action based on one template that combines votes from all stationary objects in a single mixture distribution. This results in adding one (goal-selection) child per template-selection node to consider possible successor states of applying the corresponding action. We show the results in Figure 5.30 when using 50,000 iterations for both approaches. By relying on multiple, simple interpretations of each action, our approach (multiple templates) reduces the dimensionality of the problem and is able to satisfy 86.4% of the relevant relations by applying 13.8 actions after 39.66 sec on average. In contrast, sampling from a single mixture model based on all stationary objects increases the branching factor and enabled this baseline to sequence only up to 7.1 actions to satisfy 43.8% of the relevant relations after 30.97 sec on average. By considering several, simple action interpretations individually, our approach tackles the ambiguity in the action demonstrations while achieving faster convergence rates compared to a single, implicit action model.

Furthermore, we conducted the same experiment using the Bellman-ϵ variant of our algorithm but with the following baselines with respect to the action ($\mathcal{A}$) and intention

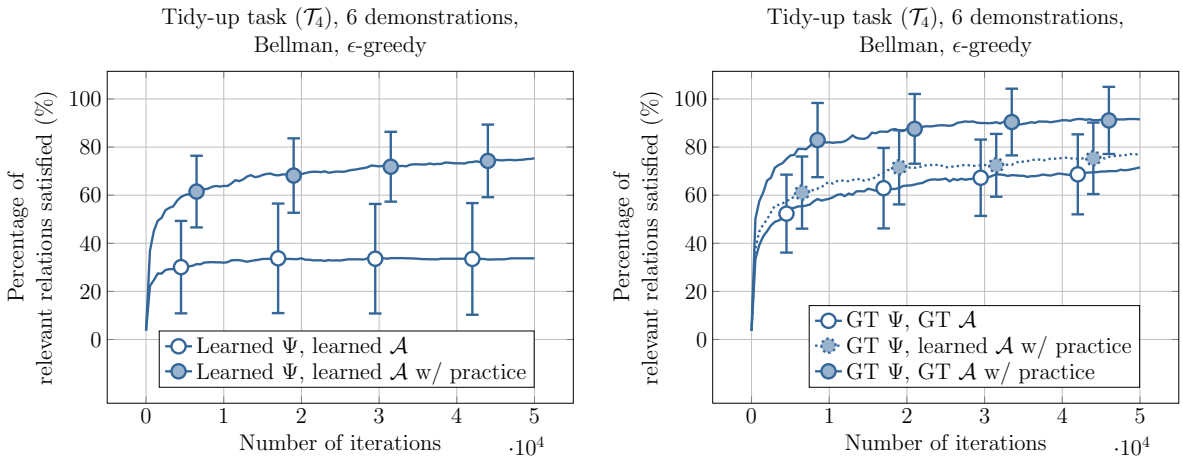


Figure 5.31.: Results for solving the tidy-up task ($\mathcal{T}_4$) in 100 random initial states using both learned and hand-designed models (see details in Section 5.7.9). Using learned action models (that consider all templates) after practicing is on par with using hand-designed models (that consider only relevant templates) before practicing. Our approach learns to use all stationary objects (even seemingly irrelevant ones) as needed, e.g., based on the sequence of applying actions or physical constraints.

likelihood models (Ψ):

- Learned Ψ, learned $\mathcal{A}$: all models learned from the demonstrations.
- Learned Ψ, learned $\mathcal{A}$ w/ practice: using the learned models after practicing the task to update the action models.
- GT Ψ, GT $\mathcal{A}$: using a ground truth (hand designed) intention likelihood model (irrelevant relations have a weight of zero) and ground truth action models (irrelevant templates/votes have a weight/probability of zero). We considered relevant templates as those that represent stationary objects with which moving objects have relations that are relevant for the task.
- GT Ψ, GT $\mathcal{A}$ w/ practice: using the ground truth models after updating the action models by practicing the task.
- GT Ψ, learned $\mathcal{A}$ w/ practice: using the ground truth intention likelihood model with the action models learned from the demonstrations, updated after practicing the task.

We report the percentage of satisfied relevant relations in Figure 5.31. As expected, we achieved the worst performance of 33.8% using the learned (noisy) models before

practicing the task (learned Ψ, learned $\mathcal{A}$) and the best performance of 91.5% using the designed models after practicing the task (GT Ψ, learned $\mathcal{A}$ w/ practice). However, it is interesting that using the learned action models after practicing the task (learned Ψ, learned $\mathcal{A}$ w/ practice achieved 75.3% and GT Ψ, learned $\mathcal{A}$ w/ practice achieved 76.9%) is on par with using the designed models before practicing the task (GT Ψ, GT $\mathcal{A}$ achieved 71.6%). In other words, our approach learns to use all stationary objects of an action by updating the models to reflect the usefulness of all objects and their votes for solving the task. This enables our algorithm to leverage all objects (even seemingly irrelevant ones) as needed, e.g., based on the sequence of applying actions or physical constraints. This results in a better generalization to states not demonstrated by the teacher compared to hand-designed action models.

5.7.10. Improvising Task Solutions Based on the Initial State

The purpose of this experiment is to demonstrate the ability of our approach to improvise task solutions based on the initial state without requiring the user to specify a goal.

5.7.10.1. Adapting to Clutter in the Scene

Figure 5.32 shows qualitative examples of solving the aligning blocks task ($\mathcal{T}_2$) in initial states including clutter caused by objects unrelated to the task. Whereas it is physically infeasible to achieve all relevant relations demonstrated by the teacher (e.g., satisfying object positions relative to the table), our approach improvised goal configurations that satisfy as many relations between the objects as possible. By formulating an optimization problem, we are able to adapt task solutions to new situations without requiring an explicit goal state. We refer the reader to Appendix A.4.1 for similar examples for $\mathcal{T}_4$.

5.7.10.2. Solving the Task with Some Objects Missing

In this experiment, we tested the ability of our approach to solve the tower building task ($\mathcal{T}_3$) given a subset of the task objects. We sampled 100 initial states and randomly removed two of the six blocks needed for constructing the tower in each of them. We solved these cases using our approach (Bellman-ϵ*) given models learned from two to ten demonstrations after practicing the task (see Section 5.7.6). We repeated this experiment 25 times using different subsets of demonstrations in each case.

Due to the missing blocks, it is not possible to satisfy all relevant relations demonstrated by the teacher. Depending on which blocks exist in the scene, it is possible to construct towers consisting of two to four blocks, which corresponds to two to six relations, respectively. We show the results in Figure 5.33 in which we report the percentage of relevant relations satisfied out of six theoretically possible ones. Our

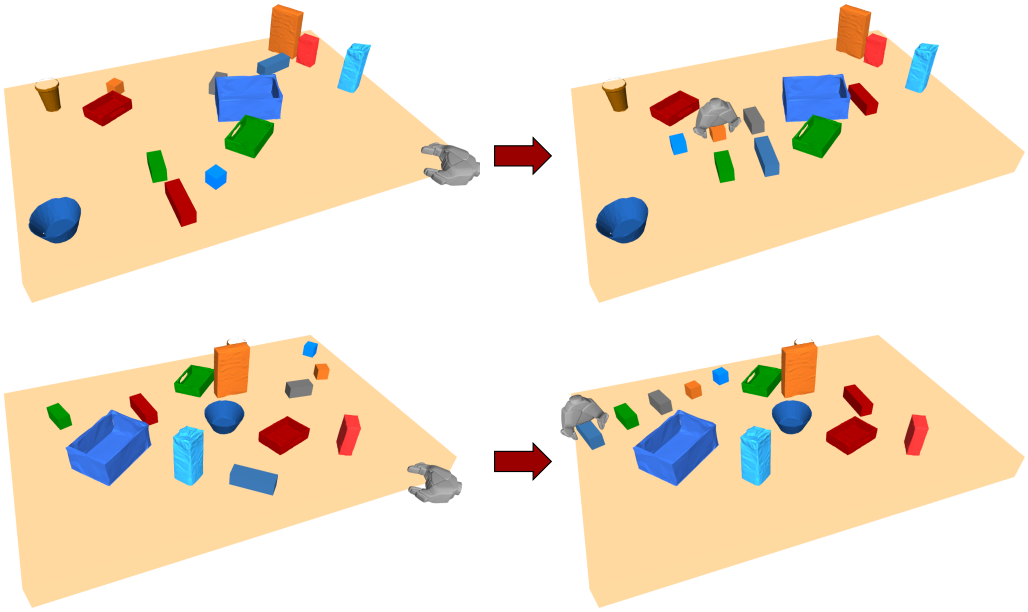


Figure 5.32.: Examples of solutions by our approach for the aligning blocks task ($\mathcal{T}_2$). Despite clutter caused by task-irrelevant objects, our approach improvised feasible solutions that satisfy as many pairwise relations as possible, e.g., by constructing two rows of blocks (top) or adapting the position and orientation of the row of blocks relative to the table (bottom).

approach is able to adapt to the missing objects in the scene to satisfy as many relations as possible in each case, achieving scores of 46.6%-48.2% by applying 4.6-4.8 actions on average. Figure 5.34 shows examples of cases from this experiment. We refer the reader to Appendix A.4.2 for examples of solving $\mathcal{T}_2$ in partially-solved initial states.

5.7.11. Real Robot Experiments

In the experiments above, we focused on evaluating our approach with respect to generalizing the teacher demonstrations to satisfy task-relevant relations. For this, we considered feasibility constraints in terms of stability and collisions (Section 2.2). In this section, we demonstrate the applicability of our approach on a real robot by additionally considering inverse-kinematic constraints of the robot's manipulator. Figure 5.35 shows qualitative examples in simulation in which the PR2 robot uses our approach to solve tasks $\mathcal{T}_2$-$\mathcal{T}_4$. Our approach enables the robot to satisfy physical constraints (e.g., some objects being unreachable) by improvising solutions that maximize the intention likelihood as much as possible.

Additionally, we implemented our approach on a real PR2 robot and performed experiments using four additional tasks. We show example runs of two tasks in Fig-

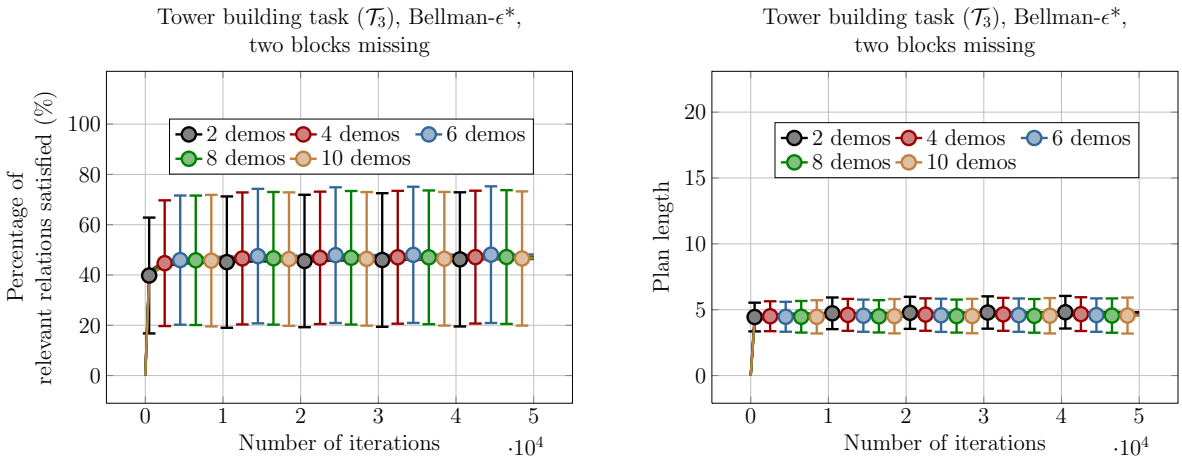


Figure 5.33.: Results for solving the tower building task ($\mathcal{T}_3$) in initial states with two (random) blocks missing. Our approach adapts by constructing feasible towers that satisfy a subset (46.6%-48.2%) of the possible pairwise relations without requiring explicit goal states to handle the new conditions.

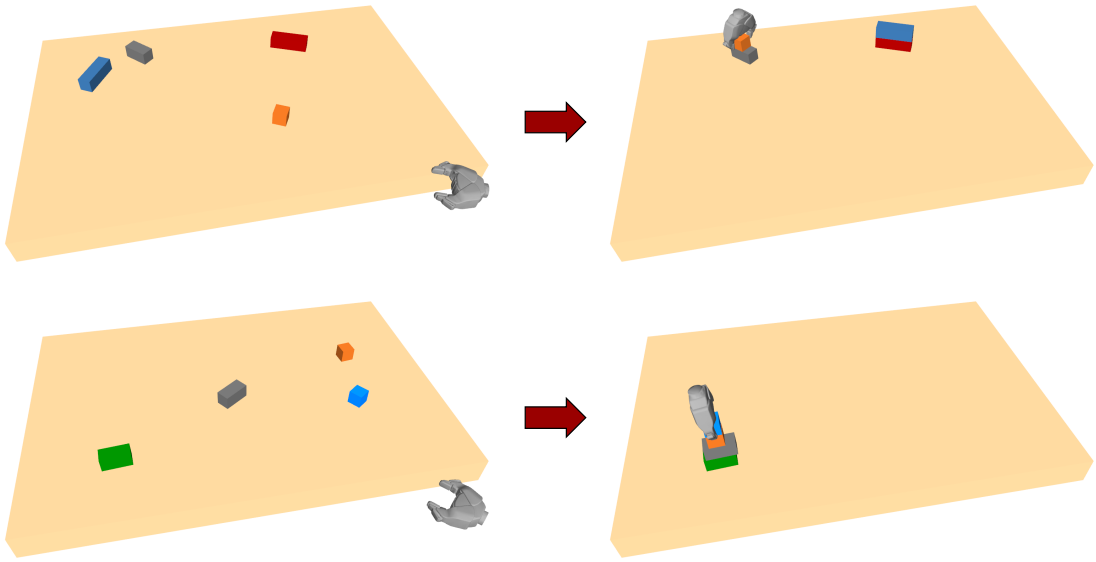


Figure 5.34.: Our approach adapts to missing blocks in the tower building task ($\mathcal{T}_3$) to compute feasible solutions that satisfy 33.3% (top) and 100% (bottom) of the relations between the four blocks in the scene, see Figure 5.33.

ures 5.36 and 5.37 and refer the reader to Appendix A.5 for additional examples. For perception, we relied on SimTrack (Pauwels and Kragic, 2015) to recognize the objects and compute their poses using a depth camera attached to the robot's head. We also used the camera to estimate the dimensions and pose of the table using a point cloud of

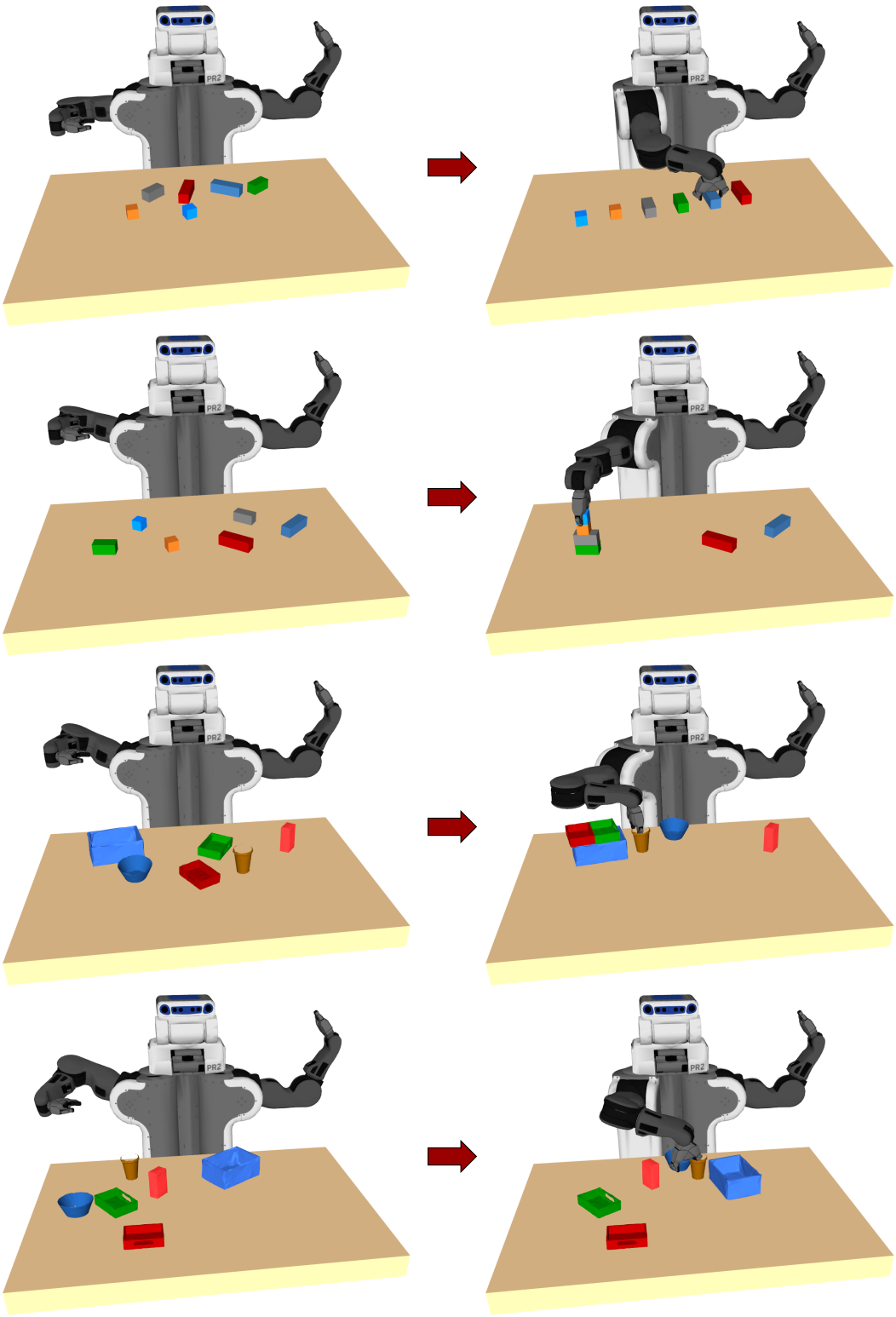


Figure 5.35.: Examples for solving $\mathcal{T}_2$ (first row), $\mathcal{T}_3$ (second row), and $\mathcal{T}_4$ (third and forth rows). Our approach enables a robot to generalize the demonstrations of the teacher to improvise feasible solutions that satisfy physical constraints (e.g., which objects are reachable). In the last example, the robot moved the blue box in order to arrange the kitchen-related objects.

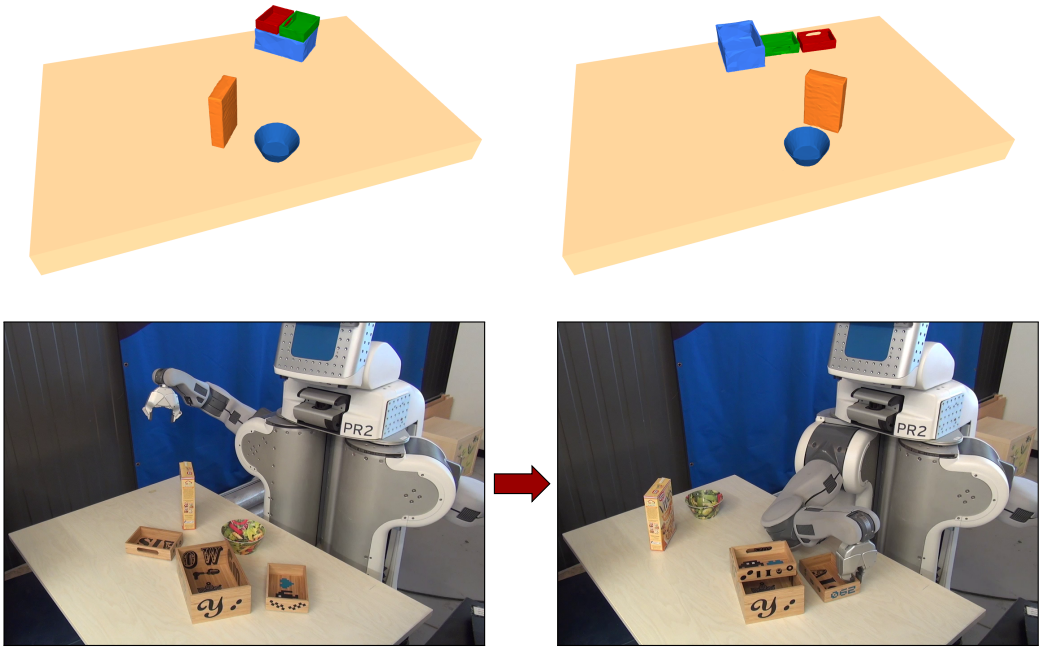


Figure 5.36.: Top: examples of demonstrated goals we provided to the robot. Bottom: using our approach, the robot was able to compute and execute a feasible plan to reproduce the demonstrated task by placing the box of cereals close to the bowl and arranging the toy boxes.

the scene. To execute the plans computed by our approach, we relied on the MoveIt! motion planning library (Sucan and Chitta, 2013).

We performed around 100 experimental runs over all tasks. In each run, we requested the robot to compute and execute a plan to reproduce one of the demonstrated tasks starting in an arbitrary initial state. We relied on max-value updates in these experiments. Note that plan monitoring and failure detection are outside the scope of our work. The only sources of failure we encountered in our experiments were due to errors in estimating the object poses (e.g., due to lighting conditions or occlusions), which may lead to an object slipping from the end-effector, or cause the motion planner to abort plan execution due to a false-positive collision detection.

Using our approach, the robot was able to improvise feasible solutions that reproduce demonstrated spatial relations between the objects based on the initial state. By considering feasibility constraints in planning, our algorithm enables the robot to compute sequences of actions leading to desirable object configurations, which the robot can execute in practice.

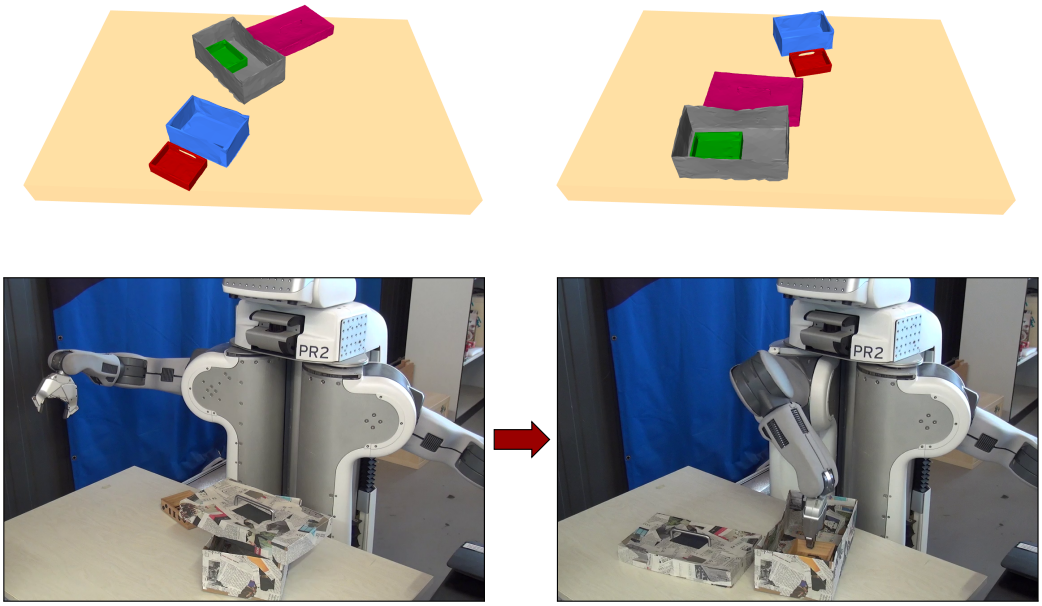


Figure 5.37.: Top: examples of demonstrated goals we provided to the robot. Bottom: using our approach, the robot was able to compute and execute a feasible plan to reproduce the demonstrated task by opening the big box in order to place the smaller one inside it.

5.7.12. Summary and Discussion of Results

In this section, we present a brief summary of our results, highlighting the characteristics of our approach. Our extensive evaluation on several tasks demonstrated the potential and flexibility of our approach and supported the claims we made at the beginning of Section 5.7, which we recap below.

- **Our intention likelihood model captures the relevance of pairwise relations to the task.** We do not assume to have semantic information about the objects or the goal of the task. Therefore, we adopt a data-driven model of the intention likelihood and express spatial relations as distributions over relative poses between pairs of objects. This allows us to reason about multiple modes for each relation (e.g., Figure 5.13). As demonstrated in Section 5.7.4, we are able to estimate the relevance of each relation for the task based on the consistency in the demonstrations. This is particularly challenging for tasks in which some relations are irrelevant as we do not rely on negative training examples for learning.

- **Our approach is able to generalize and sequence demonstrated actions.** The small number of demonstrations results in ambiguity with respect to the intention of the teacher for each action. Nonetheless, we demonstrated in Section 5.7.5

how our algorithm relies on the implicit action models to explore different interpretations for each action driven by the intention likelihood as a search heuristic. Accordingly, we were able to reproduce tasks from initial states not demonstrated by the teacher and to satisfy as many relations as possible in an anytime manner.

- **Our approach enables the robot to improve its performance by practicing the task in simulation.** In Section 5.7.6, we demonstrated how we use the plans computed by our algorithm to extract statistics that capture the usefulness of action templates and votes for solving the task. This allowed us to update the action goal distributions to more accurately reflect the intention of the teacher without requiring additional demonstrations. Using the updated action models, we showed in Section 5.7.7 a substantial improvement of our approach. Specifically, we were able to compute more efficient plans to satisfy more relevant relations using less iterations compared to before practicing the task.

- **Different variants of our algorithm have different characteristics.** The two value update strategies we explored in this work optimize different objective functions as explained in Section 5.5.3. In Sections 5.7.5 and 5.7.7.4, we demonstrated how max-value updates are more robust to noise in the action models compared to Bellman updates since max-value variants greedily maximize the intention likelihood of the final state. Therefore, they outperform Bellman variants when considering the final object configuration only. On the other hand, Bellman variants consider action goal probabilities and therefore optimize for the expected value of the intention likelihood. Therefore, their performance significantly improves after practicing the task and updating the action models as we demonstrated in Sections 5.7.7 and 5.7.7.4. Bellman variants also result in more efficient plans compared to max-value variants since they consider the intention of the teacher for each step of the plan and not only the final state. Accordingly, they tend to better generalize the demonstrations and satisfy less irrelevant relations compared to max-value updates. Finally, in Section 5.7.7 we demonstrated how we achieve more relations using less iterations when relying on only a subset of the action templates as this reduces the branching factor during search. However, this typically results in computing more expensive (longer) plans.

- **Our proposed tree structure results in efficiently tackling the ambiguity in the demonstrated actions.** By incorporating template-selection nodes, we demonstrated in Section 5.7.9 the ability of our algorithm to reduce the dimensionality of the problem by considering one action interpretation at a time during node expansion. We also demonstrated the flexibility of our approach in leveraging all stationary objects to apply actions as needed based on the current state rather than committing to a specific action model beforehand. Our proposed

tree structure also enabled us to gather statistics about action template selection rates to update action models from experience as shown in Section 5.7.6.

- **Our approach enables the robot to improvise solutions without requiring an explicit goal state.** Through our experiments in Sections 5.7.5 and 5.7.7, we demonstrated the ability of our approach to generalize teacher demonstrations and satisfy desirable relations between objects starting in arbitrary states. In Section 5.7.10, we additionally demonstrated how our algorithm enables a robot to adapt its behavior to varying conditions such as clutter or missing objects by improvising solutions that align with the intention of the teacher.

- **Our approach is applicable on a real robot.** By relying on out-of-the-box perception and motion planning solutions, we demonstrated in Section 5.7.11 that a real robot can use our approach to compute and execute physically feasible plans based on a small number of teacher demonstrations. Without requiring an explicit goal state, the robot was able to sequence several point-to-point actions to move objects relative to each other and achieve desirable arrangements based on the perceived initial state in each case.

5.8. Related Work

The approach we present in this chapter addresses learning action and task models from ambiguous demonstrations to enable a robot to reproduce the intention of a teacher without requiring an explicit goal state. Our approach overlaps with a large spectrum of prior work in the areas of learning from demonstrations, reinforcement learning, and planning. In this section, we present a thorough review of the most related works in this context in order to highlight our contributions.

Learning Tasks from Demonstrations There exists a plethora of works that address learning manipulation tasks which typically require executing sequences of actions. Here, we discuss approaches that tackle skill learning on the trajectory level based on demonstrated motions. In this context, the work of Calinon et al. (2007, 2009) provides a statistical framework for extracting spatial and temporal constraints and their relevance to each part of the task. For example, the robot learns to satisfy constraints relative to a ball while executing a reaching motion to grasp it, and subsequently switches to satisfying constraints relative to a basket during a placing motion. They encode actions probabilistically and generate motions using Gaussian mixture regression to robustly generalize the demonstrated trajectories. Analogously, Mühlig et al. (2009b) presented an approach that automatically selects task spaces (subsets of spatial constraints) for modeling demonstrated motions. In addition to statistical measures based on variances

in the demonstrated motions, they consider criteria such as attention to salient objects to select and blend different task spaces when reproducing that task.

Furthermore, several approaches addressed the problem of sequencing movement primitives when solving tasks. For example, Manschitz et al. (2014) use kinesthetic demonstrations to construct a representation of a task in terms of a sequence graph in which each node corresponds to a primitive. Accordingly, they train classifiers to map the current state modeled using sensory features to the next primitive to execute. In their later work, Manschitz et al. (2015) additionally address sequencing and synchronizing concurrent movement primitives, for example in the case of bi-manual manipulation. Related to that, Niekum et al. (2015) recently presented an approach that leverages a Bayesian nonparametric model to identify repeated structure in trajectories to segment teacher demonstrations. Accordingly, they construct a finite-state machine that captures the high-level structure of the task, and train classifiers to determine state transitions. They encode each learned skill using dynamic movement primitives transformed relative to the reference frame (e.g., one of the objects) with the highest consistency (density of end-points) across the demonstrations. Whereas the work of Niekum et al. enables a robot to learn a sequential task such as furniture assembling, Ewerton et al. (2015) recently presented an approach that models demonstrated skills as probabilistic movement primitives and learns multi-modal patterns of interaction between a teacher and a robot. Accordingly, their work enables a robot to collaborate with a human to solve an assembly task, e.g., by handing over or holding objects.

Moreover, the work of Figueroa et al. (2016) addresses segmenting demonstrations of complex sequential tasks to learn individual actions and extract the relevant constraints for each learned primitive. Additionally, they learn relevant object features to model the effects of primitives in the context of deformable objects, e.g., the shape and size of dough as a result of applying a rolling primitive. Related to that, Ureche et al. (2015) recently presented an approach to segment sequential tasks by detecting changes in task constraints (e.g., end-effector position, force, and stiffness), and to identify the relevant constraint for each stage of the task accordingly. Their method identifies the relevant reference frame for each step as the one in which the task variables are more consistent. By assuming that the flow of actions in the task is implicit in the demonstrations, their approach enables the robot to reproduce the demonstrated sequence of skills while transitioning between relevant task constraints. For example, the robot learns to satisfy position constraints relative to a target object while reaching for it, and to consider force and stiffness constraints while executing a grating motion. Moreover, Saveriano et al. (2015) recently presented an incremental learning approach that is able to prioritize and transition between end-effector and null-space task constraints. This enables the refinement of already-learned primitives through kinesthetic teaching, for example to adapt the position of the elbow to avoid an unforeseen obstacle.

In the context of human-robot interaction, Lee et al. (2009) presented an approach

that leverages the observed saliency of objects during the demonstrations to reason about the relevance of an object for a task. Furthermore, the approaches of Akgun et al. (2012) and Akgun and Thomaz (2016) rely on a non-expert user to intuitively specify keyframes for the robot to follow when reproducing sequential skills such as gestures or pouring. They enable the robot to smoothly execute the learned skills by modeling trajectories using splines. Recently, Alexandrova et al. (2015) proposed a flow-based visual programming language that enables a user to intuitively program a robot with the knowledge needed to solve a task. For instance, the user may specify a precondition for a manipulation skill (e.g., whether an object exists).

The above works focused on learning skills on the motion level, i.e., in cases where the dynamics of an action are relevant for successfully reproducing the demonstrations. Accordingly, they focus on extracting constraints to model trajectories that generalize the training data. In contrast to that, our work addresses learning point-to-point actions that are modeled only based on their endpoints. Therefore, our approach cannot account for trajectory-relevant constraints such as force and velocity profiles. On the other hand, the above works rely on simple heuristics, statistical measures, or a teacher to determine relevant frames of reference for learning and executing motion primitives. In contrast to that, our method encodes multiple interpretations of each action in terms of target reference frames based on the stationary objects in the scene. We leverage this flexibility while planning by selecting different models of the same action based on the situation rather than committing to a single model.

Moreover, as the above works focus on reproducing skills on the motion level, they enable a robot to achieve a specific task goal while generalizing to perturbations or changes in the initial state. On the other hand, we adopt a flexible, multi-modal intention likelihood model, and are thus able to achieve varying goal configurations by imitating spatial relations learned from the teacher. Finally, we rely on existing motion planners and controllers to execute each step of a plan. Accordingly, the above works can also be leveraged in our framework to reproduce demonstrated trajectories for each action, e.g., in the form of dynamic movement primitives as in our previous work (Abdo et al., 2013). This, however, is outside the scope of this work.

Leveraging Symbolic State Representations A common approach to solving complex manipulation tasks is to leverage symbolic representations of actions that allow efficient planning by abstracting continuous state spaces. In this context, there exist several techniques for integrated task and motion planning, which compute plans satisfying both symbolic conditions and geometric constraints (Dornhege et al., 2009; Plaku and Hager, 2010; Kaelbling and Lozano-Pérez, 2011, 2013; Srivastava et al., 2014; Gharbi et al., 2015; Lagriffoul and Andres, 2016). Whereas such approaches enable solving complex tasks, they place a large burden on experts to design the corresponding domains and specify the required mappings between useful logical predicates and continuous states.

Several researchers have addressed this by proposing methods for learning symbolic action models from teacher demonstrations to use for planning. In this context, a commonly used representation is based on the approach by Pasula et al. (2007), which learns probabilistic rules from training data consisting of state-action-outcome tuples. For this, they leverage a dictionary of basic symbolic predicates that they combine to construct action models maximizing the likelihood of the demonstrated action effects subject to a penalty on model complexity. Such rules have been successfully leveraged in planning to solve complex tasks or to learn models of articulated objects (Lang and Toussaint, 2010; Höfer et al., 2014). Alternatively, Wachter et al. (2013) segment demonstrations by detecting contacts between objects. They then match each segment to a manually-designed symbolic action from an existing library of object-action complexes that specifies the preconditions and effects of actions. Prior to that, Pardowitz et al. (2007) addressed the problem of decomposing a task such as setting a table into sequences of elementary skills with pre- and postconditions extracted from demonstrations and vocal comments by a teacher. Whereas their method focuses on learning hierarchical graphical representations of tasks, they do not address the problem of manipulating objects to reproduce the demonstrations. Related to that, Ekvall and Kragic (2006) presented an approach to decompose a task into different subtasks and extract constraints that determine the order of achieving each subtask. Their approach enables the robot to determine relevant features for each part of the task (e.g., relative vs. absolute positions) by analyzing the variance in the demonstrations. They represent the learned constraints in XML and leverage a high-level planner to reproduce the task based on the current state while relying on a low-level path planner for feasibility checks.

Recently, Paxton et al. (2016) presented a novel method for combined task and motion planning based on skill models grounded from expert teacher demonstrations. The approach assumes prior knowledge of the task in terms of a symbolic planning domain description and grounds actions from this domain based on the expert demonstrations. On the geometric level, the approach encodes actions probabilistically as dynamic movement primitives. Accordingly, the authors present a planning algorithm similar to Monte Carlo tree search to optimize sequences of actions that reproduce the task using trajectory simulations. The main idea is to iteratively update each action distribution to remain as close as possible to the prior based on the demonstrations while also satisfying constraints such as different obstacles and object shapes. The approach leverages a symbolic planner to translate the task description into a graph of possible actions and models transitions from symbolic states to actions as discrete probability distributions.

Related to that, Savarimuthu et al. (2017) recently presented a system for teaching robots the semantics of assembly tasks from teacher demonstrations. The proposed system enables learning assembly tasks on both a low, sensory-motor level and a high, semantic level based on object interactions. A mid-level representation encodes continuous, low-level signals as sequences of symbolic states that capture the task

topology and facilitates high-level planning and execution monitoring. Other approaches have focused on learning the mapping between symbolic predicates and geometric object configurations. For example, the approach by Dearden and Burbridge (2014) uses positive and negative training examples of predicates (e.g., `above`, `touching`) to learn classifiers for forward and backward mapping between symbolic states and geometric ones based on object poses. Accordingly, they present a planning algorithm that first computes a symbolic plan to achieve a given goal, which they then convert to a sequence of geometric states for the robot to achieve using the learned mappings.

Rather than relying on predefined predicates, other researchers proposed methods for learning symbolic action models by abstracting continuous demonstrations. For example, Ogawara et al. (2003) extract essential interactions between objects that appear consistently across all demonstrations and use them to model a task symbolically. On the motion level, they learn the demonstrated trajectories relative to those extracted interactions, thereby enabling them to generalize and reproduce the demonstrations. Moreover, other approaches extract relevant preconditions and effects of actions from continuous teacher demonstrations (Abdo et al., 2013; Cubek et al., 2015) or from experience in a reinforcement learning domain (Konidaris et al., 2014, 2015). By representing these conditions in the (probabilistic) planning and domain definition language, these methods are able to leverage existing high-level planners to sequence learned actions and solve tasks. Our previous work additionally enables the robot to model the motions associated with each symbolic action using dynamic movement primitives learned from the demonstrated trajectories (Abdo et al., 2013). Accordingly, the robot is able to select the best way to imitate each action on a low geometric level while executing symbolic plans.

The above works leverage existing or learned symbolic states that abstract continuous domains to achieve efficient, high-level planning. Symbolic domains additionally enable solving complex tasks while generalizing to changes in object instances and quantities. However, the corresponding planning paradigms assume that the user provides the goal of the task to the robot in the form of an explicit continuous or symbolic state. In contrast to that, a key contribution of our work is formulating task imitation as an optimization problem with respect to the intention likelihood. This enables the robot to improvise solutions based on the initial state and without constantly querying the user each time it has to solve the task. Additionally, our approach enables feasible planning by relying on clustering to discretize action goals, which is analogous to leveraging symbolic state abstractions in classical planning. From this perspective, our implicit action models are analogous to the noisy probabilistic rules proposed by Pasula et al. (2007). However, rather than learning these rules based on existing high-level relations, we target learning in contexts where such knowledge is unavailable a-priori, and are thus able to model arbitrary, multi-modal spatial relations in a continuous space.

Recently, Toussaint (2015) presented a novel approach for formulating sequential

manipulation tasks as an optimization problem in which the goal can be expressed as an objective function over desirable geometric states rather than an explicit symbolic state. Their method represents a first-order logic extension of a non-linear mathematical program subject to symbolic, kinematic, and geometric constraints. They rely on Monte Carlo tree search to first compute sequences of symbolic actions and states, which are then further-optimized subject to kinematic constraints. Analogous to our work, Toussaint (2015) tackles sequential manipulation tasks by optimizing a cost function without requiring the user to provide an explicit goal state. However, our work addresses learning models of the task and its actions from teacher demonstrations and focuses on inferring the intention of the teacher with respect to those models.

Segmenting Teacher Demonstrations For learning tasks, our approach assumes that the demonstrations are segmented either by the teacher or automatically using existing techniques. In our work, we adopted a heuristic that segments demonstrations based on which objects are observed to move in each part, and used the resulting segments to learn point-to-point action models. Whereas we do not consider trajectory segmentation in our work, we briefly discuss previous works that address segmenting demonstrations for learning skills. For example, several approaches leverage detected contacts between objects or the co-motion between the teacher's hands and salient objects in the scene to segment demonstrations (Gienger et al., 2010; Wachter et al., 2013; Wachter and Asfour, 2015). Recently, Baisero et al. (2015) presented a method based on linear-chain conditional random fields to decompose demonstrations into atomic actions by detecting concurrent interaction phases between pairs of objects.

Moreover, the work by Meier et al. (2012) segments demonstrations by identifying previously-learned movement primitives in observed trajectories. Related to that, Kulic et al. (2009) proposed an online approach for segmenting demonstrations by assuming that data generated from the same motion primitive originates from the same underlying distribution. Accordingly, their approach groups learned motions in a hierarchical fashion based on their similarities. Finally, other approaches addressed segmenting unstructured demonstrations to extract skills by relying on Bayesian nonparametric techniques (Niekum et al., 2015; Figueroa et al., 2016) or based on statistical changepoint detection (Konidaris et al., 2012; Ureche et al., 2015).

Reinforcement Learning As discussed in Section 5.4.2, the problem we address in this chapter shares similarities with Markov decision processes. This allowed us to formulate solving a task as an optimization problem and to leverage MCTS for planning. Additionally, our approach enables the robot to practice the task in simulation, thereby improving the action models it learned from ambiguous demonstrations. Therefore, we briefly discuss previous works that leveraged reinforcement learning techniques to learn, improve, and sequence skills.

In this context, Bentivegna et al. (2004) presented an approach to learn primitives corresponding to simple behaviors from teacher demonstrations. Rather than sequencing these primitives in a planning framework, their approach enables an agent to handle dynamic environments (e.g., playing an air hockey game) by selecting a primitive from its library to execute based on the current state. Their work enables the agent to improve with respect to selecting primitives and generating the corresponding sub-goals and motions by performing trials on its own.

In the context of learning manipulation skills, as discussed in Section 4.4, there exists a large body of work that leverages both imitation and reinforcement learning to learn policies initialized from teacher demonstrations (Kober and Peters, 2009, 2010; Stulp et al., 2011; Kalakrishnan et al., 2011; Da Silva et al., 2012; Finn et al., 2016). Reinforcement learning has also been leveraged to sequence simple primitives to acquire complex manipulation skills such as pendulum swinging and balancing or bi-manual manipulation (Neumann et al., 2009; Kroemer et al., 2015).

Moreover, Konidaris et al. (2012) presented an approach for constructing skill trees from demonstrations in the context of hierarchical reinforcement learning. Their work enables a robot to segment demonstrations into skills via changepoint detection, and to assign each skill an abstraction from a library of predefined abstractions. Accordingly, the approach merges skill chains from multiple demonstrations into a skill tree and determines which trajectory segments are instances of the same policy. Using a policy learning technique, the robot is also able to improve the learned skills. Related to that, Grave and Behnke (2013) leveraged interactive learning from demonstration and reinforcement learning to acquire manipulation primitives and sequence them for solving tasks. Grave and Behnke (2014) additionally decompose tasks into several levels of abstraction in a hierarchical reinforcement learning framework and define rewards for each task level. Accordingly, they propose a Bayesian exploration strategy to balance the optimization of expected values and the risk from exploring unknown actions.

The above works address imitation learning of skills by allowing a robot to improve policies based on observed rewards, which are typically manually designed by an expert as cost functions that encourage desirable behaviors. However, engineering such functions is not always feasible or straightforward in complex, stochastic domains. Inverse reinforcement learning addresses this problem by recovering reward functions based on teacher demonstrations (Ng et al., 2000; Abbeel and Ng, 2004; Ziebart et al., 2008; Herman et al., 2016). Moreover, other approaches proposed methods to learn distributions of manipulation trajectories that match teacher demonstrations (Englert et al., 2013) or that model joint trajectories of multiple agents in the context of socially-compliant navigation (Kuderer et al., 2012). Recently, Duan et al. (2017) presented a novel imitation learning approach that does not assume knowledge of task rewards. They aim to accelerate learning new tasks based on demonstrations of related tasks. By observing pairs of demonstrations sampled from different tasks, they train a neural

network that predicts an action based on the current state. Accordingly, the network is able to imitate a new task at test time based on a single demonstration.

Whereas the above works considered reinforcement learning in purely continuous domains, other researchers leveraged relational knowledge about tasks in the framework of relational reinforcement learning (Džeroski et al., 2001; Kersting et al., 2004). In this context, techniques for solving relational MDPs have enabled solving manipulation tasks in stochastic domains using noisy rules learned from demonstrations (Lang and Toussaint, 2010) and computing high-level plans for assembly tasks in the context of interactive learning from demonstrations (Mollard et al., 2015).

Indeed, leveraging teacher demonstrations in the context of reinforcement learning has enabled robots to learn complex manipulation skills and to improve their performance by trial and error. A robot can accordingly learn to act to maximize its expected rewards while applying actions in stochastic environments. By drawing inspiration from MDPs, our approach enables a robot to solve sequential manipulation tasks based on ambiguous teacher demonstrations. Rather than optimizing policies with respect to rewards received when interacting with the environment, we focus on inferring the intention of the teacher, and therefore aim to maximize the (expected) value of the intention likelihood for the task. At the same time, our work enables the robot to adapt its behavior based on physical constraints while computing plans that align with the demonstrations. As in MDPs, our work models actions probabilistically. However, a major difference to MDPs is our assumption that stochasticity stems from the latent intention of the teacher, and is not due to the dynamics of the environment.

Furthermore, our work assumes task goals to be modeled in terms of the spatial relations between objects. From this perspective, our proposed intention likelihood function can be seen as a reward received for achieving a particular goal state. We adopt a data-driven model of this function by modeling spatial relations as mixture distributions and computing state likelihoods using kernel density estimation. Previous works have also leveraged mixtures of Gaussian to model spatial relations in the context of reasoning about typical object placements in indoor environments (Welke et al., 2013; Kunze et al., 2014; Toris et al., 2015). The approach of Joho et al. (2012) adopts a hierarchical nonparametric Bayesian model for scenes consisting of different object constellations. Their method can be used to sample missing objects and their poses to complete partial scenes based on previously seen constellations. Note that in general, our framework is agnostic to the specific intention likelihood model used, making it possible to leverage richer models of object configurations such as the one proposed by Joho et al. (2012). However, our work focuses on enabling a robot to reproduce and adapt demonstrated configurations based on noisy action models learned from a teacher and without prior knowledge about the task. In the spirit of reinforcement learning, by practicing the task in simulation, we use plans leading to states with large values to update the action models and disambiguate the teacher demonstrations.

Finally, leveraging relational reinforcement learning based on logical descriptions of objects and relations allows scaling to large domains and generalizing to different instances and numbers of objects. However, our work targets scenarios in which non-experts teach tasks for which we have no prior semantic knowledge about the objects or their relations. Accordingly, our work enables learning and planning solely based on geometric relations in a continuous space. Whereas our approach does not consider the rich state descriptions and abstractions of relational domains, our flexible intention likelihood model allows for solving the task even if some of the objects are missing.

Monte Carlo Tree Search We model the problem of solving sequential tasks as an optimization problem that involves decision-making on multiple levels. This enables us to leverage MCTS to compute plans while reasoning about the intention of the teacher in a probabilistic framework (Coulom, 2006). MCTS is a powerful framework that has been successfully used to address a large spectrum of problems including MDPs, probabilistic planning, interplanetary trajectory planning, and games (Kocsis and Szepesvári, 2006; Keller and Eyerich, 2012; Hennes and Izzo, 2015; Silver et al., 2016). For a comprehensive overview of MCTS-based algorithms and applications, we refer the reader to the survey by Browne et al. (2012).

In the context of finite-horizon MDPs, Keller and Helmert (2013) recently proposed Trial-based Heuristic Tree Search (THTS), a framework for generating a large number of algorithms by specifying a few ingredients, thereby allowing for the combination of both heuristic-based and Monte Carlo-based search. As we base our algorithm on MCTS and rely on the intention likelihood heuristic to initialize values, our algorithm can be seen as a variant of THTS. However, our goal is to enable robots to solve manipulation tasks in a way that is robust to ambiguity in teacher demonstrations. Therefore, our approach extends the classical MCTS tree structure of decision and chance nodes to allow decision making on three levels: actions, action templates, and goals. This allows us to efficiently explore the space of action interpretations and goal states while improvising solutions that satisfy physical constraints. Other approaches proposed hierarchical MCTS structures that incorporate state abstractions or macro actions in the context of solving partially-observable MDPs (Vien and Toussaint, 2015; Bai et al., 2016).

Furthermore, in this work, we focused on developing a practical solution for the problem of solving tasks learned from demonstrations. We explored the use of max-value and Bellman update criteria in order to maximize the intention likelihood or the expectation thereof, respectively. In our experiments, we highlighted the differences between those variants with respect to the computed solutions and their robustness to the noise in the action goal distributions. To trade off exploration and exploitation during search, we considered the Boltzmann exploration and ϵ-greedy criteria. A thorough investigation of other value-update and selection strategies is outside the scope of this

work. For examples of other strategies, see Keller and Helmert (2013); Schulte and Keller (2014); Khandelwal et al. (2016).

Finally, in the context of robotics, MCTS has recently been used for improving robot soccer policies (Riccio et al., 2016), active object recognition through tactile sensing (Zhang et al., 2017), and multi-robot task allocation (Kartal et al., 2016). In the context of manipulation planning, Toussaint (2015) recently presented an approach that uses MCTS to compute symbolic plans that are then optimized with respect to geometric and kinematic constraints. Additionally, Toussaint et al. (2016) proposed a novel approach for modeling concurrent activities in relational domains, e.g., for human-robot collaboration when solving assembly tasks. They leverage Monte Carlo planing to estimate optimal decisions and reason about future decisions such as anticipating the activities of other agents.

5.9. Conclusion

In this chapter, we presented *teach and improvise*, a novel approach to learning sequential manipulation tasks from demonstrations. Our work leverages Monte Carlo tree search and formulates solving the task as an optimization problem. In this way, we go beyond existing solutions that require the user or an expert to provide an explicit goal state to solve the task. Instead, our algorithm enables the robot to improvise feasible solutions that depend on the initial state by maximizing the intention of the teacher with respect to relevant spatial relations in an anytime fashion. Moreover, our work does not assume prior semantic knowledge about the objects or the task and thus models spatial relations as distributions over relative poses in a data-driven manner. To address the ambiguity in the demonstrations and reduce the dimensionality of the problem, we leveraged the implicit action models we proposed in Chapter 4 and extended the standard MCTS tree structure to allow reasoning about multiple interpretations of each action while searching for promising solutions.

Through extensive experimental evaluation, we demonstrated the ability of our approach to generalize a small number of demonstrations to solve the task starting in arbitrary initial states. Additionally, we demonstrated how our approach enables the robot to improve its performance based on its own experience in solving the task and without requiring additional teacher demonstrations. This allows the robot to update the learned action models and disambiguate the original teacher demonstrations. Accordingly, by relying on Bellman value updates, our approach enables the robot to consider the intention of the teacher with respect to both action and task goals in a probabilistic framework. Finally, we demonstrated the applicability of our approach on a real robot that leveraged our algorithm to compute and execute feasible sequences of actions to achieve goal states that align with the intention of the teacher.

6. Distance Metric Learning for Generalizing Pairwise Spatial Relations to New Objects

So far in this thesis, the techniques we presented either relied on predefined spatial relations for solving a task (e.g., an object being on a shelf as in Chapter 3), or modeled relations as distributions over relative poses between objects (Chapters 4 and 5). Whereas these models enable a robot to solve tasks using the objects accounted for in its prior knowledge or those used by the teacher in the demonstrations, they limit the robot's ability to generalize relations to new objects of varying shapes and sizes. For an autonomous robot to operate effectively in human-centered environments, it should be able to learn arbitrary spatial relations in a lifelong manner and generalize them to new objects. For example, having learned to place a toy inside a basket, the robot should be able to generalize this concept using a spoon and a cup. However, the wide spectrum of everyday objects and arbitrary spatial relations renders it challenging for an expert to pre-program the robot with this knowledge. In this chapter, we address this problem by introducing a novel method from the perspective of distance metric learning. Our approach enables a robot to reason about the similarity between pairwise spatial relations, thereby enabling it to use its previous knowledge when presented with a new relation to imitate. We show how this makes it possible to learn arbitrary spatial relations from non-expert users using a small number of examples and in an interactive manner. Our extensive evaluation with real-world data demonstrates the effectiveness of our method in reasoning about a continuous spectrum of spatial relations and generalizing them to new objects.

Understanding spatial relations is a crucial faculty of autonomous robots operating in human-centered environments. We expect future service robots to undertake a

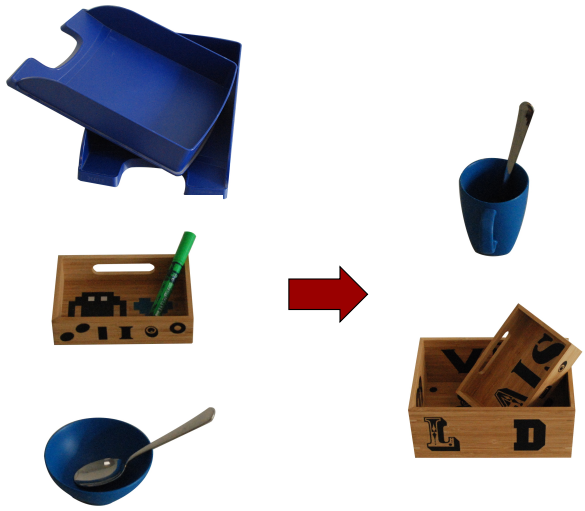


Figure 6.1.: We present a novel method based on distance metric learning to reason about the similarity between pairwise spatial relations. Our approach uses demonstrations of a relation given by non-expert teachers (left) to generalize this relation to new objects of different shapes and sizes (right).

variety of everyday tasks such as setting the table, tidying up, or assembling furniture. In this context, a robot should be able to reason about the best way to reproduce a spatial relation between two objects, e.g., by placing an item inside a drawer, or aligning two boxes side by side. So far in this thesis, we leveraged spatial relations to enable a robot to learn models of actions and tasks that align with the intention or preferences of its users. In Chapter 3, we relied on existing models of spatial relations (e.g., `on`(*box*, *shelf*)) to learn preferred object arrangements. However, this requires an expert to provide the robot with knowledge of how to satisfy such relations for each object. Alternatively, without prior semantic knowledge of objects, we modeled relations as distributions over relative poses between pairs of objects based on teacher demonstrations (Chapters 4 and 5). However, this model limits the robot's ability to reproduce demonstrated actions and tasks to the objects used by the teacher.

Therefore, our goal in this chapter is to enable a robot to learn new spatial relations from its users and generalize them to new objects of different shapes and sizes, see Figure 6.1. This is a crucial requirement for a service robot to handle the rich spectrum of objects in our everyday environments and reason about desirable spatial relations between them in order to solve tasks according to the preferences of its users. In this context, it is highly impractical for an expert to pre-program a robot with the knowledge it needs to handle all potential objects and relations in the real world, e.g., in the form of a library of predicates. Instead, we aim for a lifelong learning approach that enables

non-expert users to teach new spatial relations to robots in an intuitive manner.

One way to do this is to provide a robot with several examples using different objects in order to learn a model for a new relation such as `inside`. On the one hand, this may require generating large amounts of data to learn the new relation, which is impractical in setups in which a robot learns from a non-expert teacher. On the other hand, this requires learning a new model for each relation individually, making it hard for the robot to reuse its knowledge from previous relations.

In this chapter, we address this problem from the perspective of distance metric learning and focus on learning relations between pairs of objects. We present a novel method that allows the robot to reason about how similar two relations are to each other. For training, our approach relies only on partially labeled data describing how similar pairs of relations are to each other, and which the robot accumulates from non-expert users over time and in an interactive manner. Accordingly, we formulate the problem of reproducing a relation using two new objects as one of minimizing the *distance* between the reproduced relation and the teacher demonstrations. More importantly, our approach enables the robot to use a few teacher demonstrations as queries for retrieving *similar* relations it has seen before, thereby leveraging prior knowledge to bootstrap imitating the new relation. Therefore, rather than learning a finite set of individual relation models, our method enables reasoning on a continuous spectrum of relations.

Concretely, we make the following contributions: *i)* we present a novel approach from the perspective of distance metric learning to address the problem of learning pairwise spatial relations and generalizing them to new objects, *ii)* we introduce a novel descriptor that encodes pairwise spatial relations based only on the object geometries and their relative pose, *iii)* we demonstrate how our method enables bootstrapping the learning of a new relation by relating it to similar, previously-learned relations, *iv)* we present an interactive learning method that enables non-expert users to teach arbitrary spatial relations from a small number of examples, and *v)* we present an extensive evaluation of our method based on real-world data we gathered from different user demonstrations.

6.1. Problem Formulation

In this section, we formalize the problem we address in this chapter: learning pairwise spatial relations from non-expert teachers and generalizing them to new objects.

6.1.1. Object and State Representation

We consider the problem of learning spatial relations between pairs of objects. We denote the pair consisting of objects o_k and o_l by $p_{(k,l)} = \{o_k, o_l\}$. In this chapter, we assume to have no semantic knowledge about objects such as their type, e.g., box. Instead, we aim to learn relations based on object geometries and assume to have a 3D

model of each object o_k in the form of a point cloud $\mathbf{P}_k$. Additionally, we model the state using the 3D poses of objects in $SE(3)$ where $\mathbf{T}_k$ denotes the pose of o_k relative to the world frame and ${}^k\mathbf{T}_l$ denotes the pose of o_l relative to o_k. We assume the world frame to be specified such that the $-z$-axis aligns with the gravity vector $\mathbf{g}$. Accordingly, we define the state $\mathbf{s}$ in terms of the poses and point cloud models of all objects $\mathcal{O}$ in the environment. In this work, we rely on existing perception and segmentation techniques to compute the object poses based on their point clouds.

6.1.2. Pairwise Relations

We consider learning pairwise spatial relations between objects. Accordingly, we rely on *scenes* that consider only two objects in the state. We express a scene consisting of o_k and o_l as a tuple $\mathbf{d} := \langle p_{(k,l)}, \mathbf{s} \rangle$. This enables us to model the spatial relation between the pair of objects based on their 3D models ($\mathbf{P}_k$ and $\mathbf{P}_l$) and their relative pose ${}^k\mathbf{T}_l$ in the state $\mathbf{s}$. Furthermore, we assume that one of the objects (o_k) is labeled as the *reference object*, and therefore express the scene using the pose of o_l relative to o_k.

Given a scene $\mathbf{d}$, we rely on a descriptor function $f_{\mathcal{R}}$ to express the relation between the two objects in it as a K-dimensional feature vector $\mathbf{r}$, i.e., $f_{\mathcal{R}}(\mathbf{d}) = \mathbf{r} \in \mathbb{R}^K$. Moreover, our goal is to enable the robot to reason about how *similar* two scenes are with respect to the pairwise relations in them. We capture the similarity between $\mathbf{d}_1$ and $\mathbf{d}_2$ using a distance function dist that computes the distance $\text{dist}(\mathbf{r}_1, \mathbf{r}_2) \geq 0$ between the two scenes with respect to their feature vectors $\mathbf{r}_1$ and $\mathbf{r}_2$.

6.1.3. The Problem

The problem we address in this chapter is threefold as follows.

6.1.3.1. Representing Relations

First, we seek a descriptor $f_{\mathcal{R}}$ that enables us to capture the underlying spatial relation in a scene based only on the geometries (point clouds) of the objects, their relative poses, and the direction of gravity $\mathbf{g}$.

6.1.3.2. Learning the Distance between Relations

Given $f_{\mathcal{R}}$, we aim to learn a distance metric dist for computing the distance between two scenes. For this, we rely on training data $\mathcal{D} = \{\mathbf{d}^{(1)}, \ldots, \mathbf{d}^{(N)}\}$ consisting of N demonstrated scenes. Each scene can involve different objects and relations. Additionally, we assume to have a symmetric similarity matrix $\mathbf{Y}$ of size $N \times N$ with unit diagonal values. The value $y_{i,j}$ in the i-th row and j-th column of $\mathbf{Y}$ captures the degree of similarity between scenes $\mathbf{d}^{(i)}$ and $\mathbf{d}^{(j)}$ in $\mathcal{D}$. In this work, we consider binary similarities

$y \in \{0, 1\}$, such that 0 represents dissimilar relations and 1 means that the relations in both scenes are identical. Note that we do not assume $\mathbf{Y}$ to be completely specified, i.e., some entries may be missing. Therefore, we aim for a method that can learn with partially-labeled data with respect to scene similarities. Given $\mathcal{D}$ and $\mathbf{Y}$, our goal is to learn a *distance metric* dist that captures the distance between scenes $\mathbf{d}_1$ and $\mathbf{d}_2$ based on their features $\mathbf{r}_1 = f_{\mathcal{R}}(\mathbf{d}_1)$ and $\mathbf{r}_2 = f_{\mathcal{R}}(\mathbf{d}_2)$. We learn this metric such that $\text{dist}(\mathbf{r}_1, \mathbf{r}_2)$ is "small" if $\mathbf{d}_1$ and $\mathbf{d}_2$ represent similar relations, and "large" if they represent dissimilar relations.

6.1.3.3. Generalizing a Relation to New Objects

Given $f_{\mathcal{R}}$ and a distance metric dist, our goal is to learn a new, arbitrary relation from a teacher. We assume the teacher provides a small set of demonstrations $\mathcal{D}' = \{\mathbf{d}^{(1)}, \ldots, \mathbf{d}^{(N')}\}$ of the new relation, where $1 \leq N' \ll N$. Given two new objects o_k and o_l, the robot has to "imitate" the demonstrated relation in $\mathcal{D}'$ by computing the pose ${}^k\mathbf{T}_l$ of o_k relative to o_l such that the resulting scene $\mathbf{d}^* = \langle p_{(k,l)}, \mathbf{s} \rangle$ is close to the demonstrations with respect to the corresponding features.

Concretely, we seek the pose ${}^k\mathbf{T}_l^*$ to solve a problem of the form:

$$\begin{aligned} &\text{minimize} \quad \mathcal{L}(\mathcal{R}', \mathbf{r}^*) \\ &\text{subject to} \quad \mathbf{s}^* \in \mathcal{S}_{\text{f}}, \end{aligned} \tag{6.1}$$

where $\mathcal{R}' = \{\mathbf{r}^{(1)}, \ldots, \mathbf{r}^{(N')}\}$ is the set of features for the demonstrated scenes $\mathcal{D}'$, $\mathbf{r}^* = f_{\mathcal{R}}(\mathbf{d}^*)$ is the feature vector of the test scene $\mathbf{d}^*$, and $\mathcal{L}$ is a loss function describing the distance between the demonstrations and the test scene based on dist. Additionally, $\mathbf{s}^*$ is the state in which the pose of o_l relative to o_k is ${}^k\mathbf{T}_l^*$, and $\mathcal{S}_{\text{f}}$ is the set of physically feasible states. In this work, we focus on computing the desired pose between the two objects and do not consider the problem of manipulating the objects to achieve this pose. Therefore, we consider $\mathcal{S}_{\text{f}}$ as the set of scenes in which there is no penetration between o_k and o_l (see Section 2.2).

6.2. Proposed Feature Representation

In this section, we present our model for $f_{\mathcal{R}}$ and propose a descriptor for modeling pairwise spatial relations, thereby addressing Section 6.1.3.1. We model relations based only on the spatial interaction between their point clouds $\mathbf{P}_k$ and $\mathbf{P}_l$ given the direction of the gravity vector $\mathbf{g}$. In this work, we do not address the correspondence problem between scenes and assume the teacher specifies the reference object.

We rely on the directions of the vectors between the points of the objects as a signature of the underlying relation between them. Defining these directions purely based on

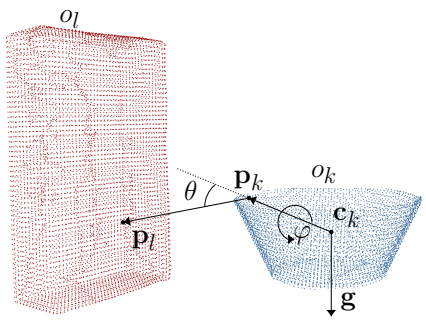


Figure 6.2.: Visualization of the descriptor computation for the spatial relation between objects o_k and o_l. We consider the gravity vector $\mathbf{g}$ at the centroid $\mathbf{c}_k$ of the reference object o_k. We compute angles θ and φ based on direction vectors involving all points $\mathbf{p}_k$ and $\mathbf{p}_l$ on o_k and o_l, respectively.

a fixed (world) reference frame is sub-optimal as this results in a descriptor that is affected by translations and rotations of the scene. At the same time, computing a local reference frame using one of the objects (e.g., using PCA) introduces the challenge of ensuring consistency and reproducibility of the axes across different scenes.

We address this problem by computing angles between direction vectors between points on both objects and the centroid of the reference object o_k, see Figure 6.2. This is analogous to methods for computing rotationally-invariant descriptors for 2D images such as RIFT (Lazebnik et al., 2005). Additionally, we aim to encode the distances between the objects and generalize with respect to their sizes. Accordingly, we propose a descriptor that is based on three histograms as follows:

$$f(\mathbf{d}) := [\mathbf{h}_\theta \quad \mathbf{h}_\varphi \quad \mathbf{h}_d]^\top. \tag{6.2}$$

With the first histogram $\mathbf{h}_\theta$, we capture the angular relation between the two objects regardless of how the scene is oriented in the global reference frame. We construct $\mathbf{h}_\theta$ as a distribution over the angle θ between vectors $(\mathbf{p}_k - \mathbf{c}_k)$ and $(\mathbf{p}_l - \mathbf{p}_k)$ based on all points $\mathbf{p}_k \in \mathbf{P}_k$ and $\mathbf{p}_l \in \mathbf{P}_l$, i.e.,

$$\theta = \arccos\left(\frac{(\mathbf{p}_k - \mathbf{c}_k)^\top(\mathbf{p}_l - \mathbf{p}_k)}{\|\mathbf{p}_k - \mathbf{c}_k\|_2 \quad \|\mathbf{p}_l - \mathbf{p}_k\|_2}\right), \tag{6.3}$$

where $\mathbf{c}_k$ is the centroid of the reference object o_k, see Figure 6.2.

For the same relative pose ${}^k\mathbf{T}_l$, $\mathbf{h}_\theta$ provides a unique signature of the scene that is invariant to its translation or rotation in the global reference frame. However, in the context of everyday manipulation tasks, it is typically useful to also reason about spatial constraints with respect to the world frame, e.g., a supporting surface such as a table. For example, this enables the robot to disambiguate scenes in which the two objects are on top of each other from those in which they are next to each other for the same ${}^k\mathbf{T}_l$.

We achieve this disambiguation using the second histogram $\mathbf{h}_\varphi$, which is a distribution over the angle φ around the vector $(\mathbf{p}_k - \mathbf{c}_k)$, see Figure 6.2. We take this as the angle between two planes. The first plane is defined by the two vectors $(\mathbf{p}_k - \mathbf{c}_k)$ and $\mathbf{g}$, whereas the second is defined by the two vectors $(\mathbf{p}_k - \mathbf{c}_k)$ and $(\mathbf{p}_l - \mathbf{p}_k)$. We compute φ as the angle between the respective normal vectors $\mathbf{n}_1$ and $\mathbf{n}_2$ of those planes, i.e.,

$$\mathbf{n}_1 = \frac{(\mathbf{p}_k - \mathbf{c}_k) \times \mathbf{g}}{\|(\mathbf{p}_k - \mathbf{c}_k) \times \mathbf{g}\|_2}, \quad \mathbf{n}_2 = \frac{(\mathbf{p}_k - \mathbf{c}_k) \times (\mathbf{p}_l - \mathbf{p}_k)}{\|(\mathbf{p}_k - \mathbf{c}_k) \times (\mathbf{p}_l - \mathbf{p}_k)\|_2}, \quad \varphi = \arccos(\mathbf{n}_1^\top \mathbf{n}_2). \tag{6.4}$$

We populate $\mathbf{h}_\varphi$ by computing φ using all points $\mathbf{p}_k \in \mathbf{P}_k$ and $\mathbf{p}_l \in \mathbf{P}_l$. As the direction of $\mathbf{g}$ is fixed, rotating the scene while maintaining ${}^k\mathbf{T}_l$ results in changes in φ, i.e., the discriminative behavior we seek. On the other hand, $\mathbf{h}_\theta$ and $\mathbf{h}_\varphi$ are invariant to translations or rotations around $\mathbf{g}$.

Whereas $\mathbf{h}_\theta$ and $\mathbf{h}_\varphi$ encode the relation with respect to the relative rotation between the two objects, we encode the desired distance between them using the third histogram $\mathbf{h}_d$. We compute $\mathbf{h}_d$ as a distribution over the Euclidean distance $\|\mathbf{p}_k - \mathbf{p}_l\|_2$ between points $\mathbf{p}_k \in \mathbf{P}_k$ and $\mathbf{p}_l \in \mathbf{P}_l$. Rather than doing so using all $|\mathbf{P}_k|\,|\mathbf{P}_l|$ pairs of points $\{\mathbf{p}_k, \mathbf{p}_l\}$, we consider the subset of pairs with the smallest 10% distances over all pairs, as this is indicative of how close the two objects are and is more robust to differences in object sizes.

Finally, we discretize both $\mathbf{h}_\theta$ and $\mathbf{h}_\varphi$ with a bin resolution of 20 deg, and discretize $\mathbf{h}_d$ with a resolution of 6 cm. We normalize all histograms using the number of points used to compute them such that $f_\mathcal{R}(\mathbf{d})$ is independent of the object point cloud densities.

6.3. Distance Metric Learning

In this section, we discuss how we learn a metric dist that models the similarities between relations given the feature representation $\mathbf{r} = f_\mathcal{R}(\mathbf{d})$ above, i.e., the problem in Section 6.1.3.2. For this, we leverage a popular metric learning technique originally introduced by Weinberger and Saul (2009) to improve the performance of k-NN classification: large margin nearest neighbor (LMNN).

We follow the terminology of Weinberger and Saul and define the set of *target neighbors* $\mathcal{R}_i^+$ for an example $\mathbf{r}_i$ as the k nearest neighbors of $\mathbf{r}_i$ that are labeled as similar, i.e. $y_{i,j} = 1$ for $\mathbf{r}_j \in \mathcal{R}_i^+$. We determine the target neighbours based on Euclidean distance and they define a region around $\mathbf{r}_i$. We refer to all examples $\mathbf{r}_k$ within this region that are not similar to $\mathbf{r}_i$ (i.e., $y_{i,k} = 0$) as *impostors* $\mathcal{R}_i^-$. The original LMNN formulation identifies $\mathcal{R}_i^+$ and $\mathcal{R}_i^-$ by assuming training data that is labeled with pre-specified classes. In our context, we achieve this based on the similarity labels y without requiring class labels to be specified by the teacher.

In the general form, LMNN learns a metric dist_ϕ parametrized by ϕ by minimizing a loss function with two objectives: *i)* for each training relation $\mathbf{r}_i$, pull target neighbors

$\mathbf{r}_j \in \mathcal{R}_i^+$ close, and *ii)* push impostors $\mathbf{r}_k \in \mathcal{R}_i^-$ away such that they are further than target neighbors $\mathbf{r}_j$ by at least a large margin ζ (Weinberger and Saul, 2009), i.e.,

$$\underset{\phi}{\text{minimize}} \sum_{\substack{\mathbf{r}_i \in \mathcal{D}, \\ \mathbf{r}_j \in \mathcal{R}_i^+}} \overbrace{\text{dist}_\phi(\mathbf{r}_i, \mathbf{r}_j)^2}^{\substack{\text{pull a similar} \\ \text{neighbor } \mathbf{r}_j \text{ close}}} + \lambda \sum_{\mathbf{r}_k \in \mathcal{R}_i^-} \underbrace{\left[\zeta + \text{dist}_\phi(\mathbf{r}_i, \mathbf{r}_j)^2 - \text{dist}_\phi(\mathbf{r}_i, \mathbf{r}_k)^2\right]_+}_{\substack{\text{push a dissimilar neighbor } \mathbf{r}_k \\ \text{further than } \mathbf{r}_j \text{ by at least } \zeta}}, \tag{6.5}$$

where $[d]_+ = \max(0, d)$ is the hinge loss and λ is a constant that controls the trade-off between the two objectives. In this work, we consider three LMNN-based methods for learning a dist_ϕ parametrized by ϕ, which we summarize below.

LMNN The standard LMNN parametrizes dist_ϕ using a linear mapping $\mathbf{L} \in \mathbb{R}^{K \times K}$ that transforms points to a space that better captures their distances, i.e., $\phi(\mathbf{r}) = \mathbf{L}\mathbf{r}$. This corresponds to learning a Mahalanobis metric such that

$$\begin{aligned} \text{dist}_\phi(\mathbf{r}_i, \mathbf{r}_j) &:= \|\mathbf{L}(\mathbf{r}_i - \mathbf{r}_j)\|_2 \\ &= \sqrt{(\mathbf{r}_i - \mathbf{r}_j)^\top \mathbf{M} (\mathbf{r}_i - \mathbf{r}_j)}, \end{aligned} \tag{6.6}$$

where $\mathbf{M} = \mathbf{L}^\top \mathbf{L}$. Weinberger and Saul proved that, by parametrizing the distance with a positive semidefinite $\mathbf{M}$, Eq. (6.5) is a convex problem that can be expressed in semidefinite form, enabling them to efficiently compute the globally-optimal solution with a custom solver (see Weinberger and Saul (2009) for details).

χ^2-LMNN This approach also adopts a linear mapping $\phi(\mathbf{r}) = \mathbf{L}\mathbf{r}$, but was introduced by Kedem et al. (2012) specifically for data represented by histograms. Hence, χ^2-LMNN computes dist_ϕ using the χ^2 distance instead of the Euclidean distance as follows:

$$\begin{aligned} \text{dist}_\phi(\mathbf{r}_i, \mathbf{r}_j) &:= \chi^2(\mathbf{L}\mathbf{r}_i, \mathbf{L}\mathbf{r}_j) \\ &= \frac{1}{2} \sum_{k=1}^{K} \frac{\left([\mathbf{r}'_i]_k - [\mathbf{r}'_j]_k\right)^2}{[\mathbf{r}'_i]_k + [\mathbf{r}'_j]_k}, \end{aligned} \tag{6.7}$$

where $\mathbf{r}' = \mathbf{L}\mathbf{r}$ and $[\mathbf{r}']_k$ denotes the k-th dimension of $\mathbf{r}'$.

GB-LMNN Gradient-boosted LMNN (GB-LMNN) goes beyond the linear LMNN approach as it is able to model arbitrary, non-linear mappings $\phi(\mathbf{r})$ of the input space:

$$\text{dist}_\phi(\mathbf{r}_i, \mathbf{r}_j) := \|\phi(\mathbf{r}_i) - \phi(\mathbf{r}_j)\|_2. \tag{6.8}$$

The special case of $\phi(\mathbf{r}) = \mathbf{L}\mathbf{r}$ corresponds to the linear LMNN metric above. To consider a wide spectrum of possible non-linear mappings, the approach of Kedem

et al. (2012) uses gradient-boosted regression trees to model ϕ. The idea is to learn ϕ as an ensemble of T simple decision trees h, i.e., $\phi = \phi_0 + \alpha \sum_{t=1}^{T} h_t$ such that GB-LMNN minimizes Eq. (6.5) directly in function space. In each iteration, we add the tree that greedily minimizes the loss by approximating the negative gradient of the loss with respect to the training data. For this, Kedem et al. use a linear metric $\mathbf{L}$ computed by LMNN as an initialization ϕ_0, and learn the trees using the parallel boosted regression trees approach (see Kedem et al. (2012) for details).

6.4. Reproducing a New Relation from a Few Demonstrations

In this section, we present our approach for imitating a new relation from a small number of teacher demonstrations (Section 6.1.3.3). We assume that the robot is already equipped with a set of relation scenes $\mathcal{D} = \{\mathbf{d}^{(1)}, \ldots, \mathbf{d}^{(N)}\}$ and a (partially-filled) $N \times N$ matrix $\mathbf{Y}$ consisting of their similarity labels as in Section 6.1.3.2. These are either provided by an expert beforehand, or are accumulated by the robot when learning previous relations over time. Using $\mathcal{D}$ and $\mathbf{Y}$, we assume the robot has already learned a prior distance metric dist_{ϕ_0} parametrized by ϕ_0 as described in Section 6.3. This is done offline and without knowledge of the new relation.

We now consider a teacher providing the robot with a small set of demonstrations $\mathcal{D}'$ of size N' for a new, arbitrary relation. The teacher can use different pairs of objects, such that all scenes in $\mathcal{D}'$ are equally valid ways of achieving this relation, i.e., $y_{i,j} = 1$ for all $\mathbf{d}^{(i)}, \mathbf{d}^{(j)} \in \mathcal{D}'$. Given two objects o_k and o_l and their respective models $\mathbf{P}_k$ and $\mathbf{P}_l$, our goal is to compute a pose ${}^{k}\mathbf{T}_l^*$ such that the resulting scene $\mathbf{d}^*$ corresponds to the intention of the teacher for the new relation, i.e., minimizing $\mathcal{L}$ in Eq. (6.1).

Given a metric dist_{ϕ^*}, there are different ways to model $\mathcal{L}$ to express the distance between the features $\mathbf{r}^* = f_{\mathcal{R}}(\mathbf{d}^*)$ of the test scene and the features $\mathcal{R}'$ of the demonstrations $\mathcal{D}'$. In general, as $|\mathcal{R}'| \geq 1$, Eq. (6.1) is a multi-objective optimization problem seeking to minimize the distance between $\mathbf{r}^*$ and all $\mathbf{r}' \in \mathcal{R}'$. In such settings, it is typically challenging to satisfy all objectives. Minimizing the (mean) distance to $\mathcal{D}'$ can thus lead to sub-optimal solutions that "average" the demonstrated scenes. Instead, we consider each demonstration to represent a mode of the target relation and seek the best solution with respect to any of them as follows:

$$\mathcal{L}(\mathcal{R}', \mathbf{r}^*) := \min_{\mathbf{r}' \in \mathcal{R}'} \text{dist}_{\phi^*}(\mathbf{r}', \mathbf{r}^*). \tag{6.9}$$

6.4.1. Interactive Local Metric Learning

One way to obtain dist_{ϕ^*} in Eq. (6.9) is to use the prior metric dist_{ϕ_0}. However, we learned this metric using a set of previous relations $\mathcal{D}$ and their similarities. Therefore,

it is not directly clear if dist_{ϕ_0} is able to generalize to novel relations.

We answer this question using an interactive approach. For each demonstration in $\mathcal{D}'$, we use dist_{ϕ_0} to retrieve the Q nearest neighbor examples corresponding to scenes from the database $\mathcal{D}$. We query the teacher with these examples and ask her to indicate whether they align with her intention ($y = 1$) for the new relation or not ($y = 0$). In our experiments, we achieved this by means of a graphical user interface visualizing $Q = 8$ nearest neighbors per query. Let $\mathcal{D}_{NN} \subset \mathcal{D}$ be the set of all nearest neighbor scenes for $\mathcal{D}'$. We measure the confidence in the ability of dist_{ϕ_0} to generalize to the new relation as the ratio ϵ_{NN} of scenes in $\mathcal{D}_{NN}$ for which the teacher indicated a similarity to the new relation (i.e., $y = 1$). ϵ_{NN} values larger than a threshold ϵ_* indicate that we are able to relate the new relation to ones the robot has seen in the past. Therefore, we use dist_{ϕ_0} to compute Eq. (6.9), i.e., $\phi_* = \phi_0$. We empirically set ϵ_* to 77% in our experiments.

On the other hand, $\epsilon_{NN} < \epsilon_*$ indicates that the new relation is far in the mapped feature space from (target neighbor) relations in $\mathcal{D}$. We address this by learning a new *local* metric dist_{ϕ_*} using the set of scenes $\mathcal{D}^* = \mathcal{D}' \cup \mathcal{D}_{NN}$ and labels $\mathbf{Y}^*$ of size $N^* \times N^*$, where $N^* = |\mathcal{D}^*|$. This is a smaller problem (compared to learning the prior metric dist_{ϕ_0}) in which $\mathbf{Y}^*$ is completely specified. We set the similarity $y_{i,j}$ to 1 for all $\mathbf{d}^{(i)}, \mathbf{d}^{(j)} \in \mathcal{D}'$. For rows i and columns j corresponding to scenes $\mathbf{d}^{(i)} \in \mathcal{D}'$ and $\mathbf{d}^{(j)} \in \mathcal{D}_{NN}$ (or vice versa), we set $y_{i,j}$ to the similarity labels obtained from querying the teacher. We use the transitivity property to set the similarity between $\mathbf{d}^{(j)}, \mathbf{d}^{(k)} \in \mathcal{D}_{NN}$. For example, if the teacher labeled $y_{i,j} = 0$ and $y_{i,k} = 0$, we set $y_{j,k} = 1$.

Finally, we highlight two main advantages of leveraging the previous relations $\mathcal{D}$ and prior metric dist_{ϕ_0}. Firstly, we enable the robot to consider whether its previous knowledge is sufficient to reproduce the new relation or not. Secondly, even for new relations that are significantly different from previously-known ones, we are able to augment the teacher's demonstrations $\mathcal{D}'$ with additional training data $\mathcal{D}_{NN}$ consisting of target neighbors and impostors retrieved from $\mathcal{D}$ without requiring the teacher to demonstrate them.

6.4.2. Sample-Based Pose Optimization

Given the metric dist_{ϕ_*} to model Eq. (6.9), we present our approach for solving Eq. (6.1) to compute ${}^k\mathbf{T}_l^*$ for reproducing the new relation using o_k and o_l. In this work, we simplify this problem by assuming that the reference object o_k is stationary and therefore only reason about desirable poses of o_l relative to it. Due to the discretization in computing our descriptor $f_{\mathcal{R}}$, we cannot rely on gradient-based methods as our loss function is piecewise constant.

We address this using a sample-based approach. We discretize the space of poses by searching over a grid of translations ${}^k\mathbf{t}_l$ of o_l relative to o_k. For each translation, we sample rotations ${}^k\mathbf{R}_l$ uniformly. We use the resulting ${}^k\mathbf{T}_l$ to transform $\mathbf{P}_l$ and compute

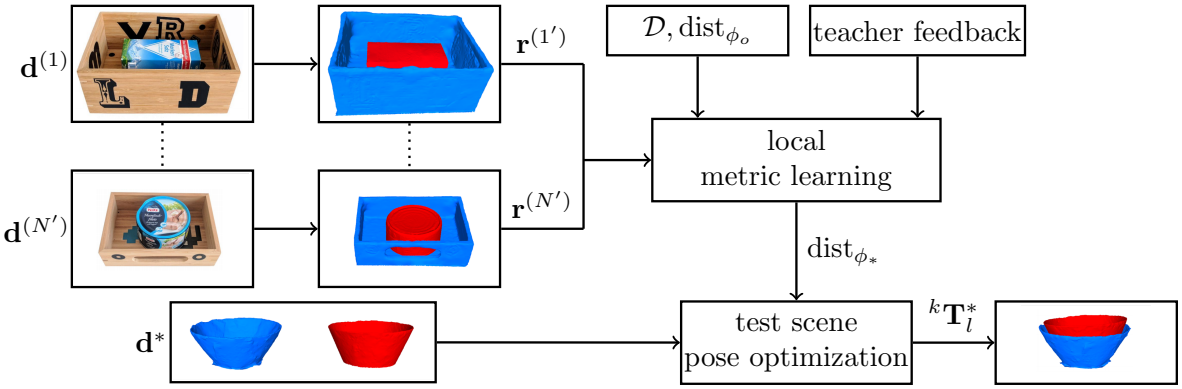


Figure 6.3.: Overview of our interactive approach for learning to reproduce a new relation. Given a small number of demonstrations $\mathbf{d}^{(1)}, \ldots, \mathbf{d}^{(N')}$ by a teacher and two objects in the test scene $\mathbf{d}^*$, we aim to compute a pose ${}^k\mathbf{T}_l^*$ transforming $\mathbf{d}^*$ in order to imitate the demonstrated relation and generalize the intention of the teacher. Our approach enables the robot to leverage its prior knowledge of scenes $\mathcal{D}$ representing other relations and the distances between them based on a metric dist$_{\phi_0}$. Accordingly, we retrieve examples from $\mathcal{D}$ to complement the training examples given by the teacher. We use this to learn a local metric dist$_{\phi_*}$ in order to reproduce the new relation.

$\mathcal{L}$ based on the corresponding feature value $\mathbf{r}^*$ of the scene. Whenever we find a new local minima during optimization, we check for collisions between the two objects and reject infeasible solutions. Finally, we take ${}^k\mathbf{T}_l^*$ as the feasible pose minimizing $\mathcal{L}$ over all sampled poses. We implemented this process efficiently by parallelizing the grid search over translations. Figure 6.3 shows an overview of our method for reproducing a new relation.

6.5. Experimental Evaluation

In this section, we present the experimental evaluation of our approach. Through our experiments, we demonstrate the following: *i)* our proposed descriptor is able to capture different spatial relations and to generalize to the shapes and sizes of the objects, *ii)* using distance metric learning, we are able to capture the similarities between scenes even for relations not encountered before, *iii)* our interactive learning method enables non-expert users to teach new relations based on a small number of examples, and *iv)* we outperform several baselines that do not learn a metric based on the similarities between scenes.

Figure 6.4.: Examples of the scenes we used for evaluation. We recorded a set of 546 scenes and manually provided ground truth labels for their similarities.

6.5.1. Baselines

In our experiments, we used three variants of LMNN-based metrics: vanilla (linear) LMNN, χ^2-LMNN, and GB-LMNN, which we learned as in Section 6.3. We compared those learned metrics to a variety of standard distance metrics. This includes the Euclidean, χ^2, Bhattacharyya, and the correlation distances, as well as the Kullback-Leibler divergence (KL) and the Jensen-Shannon divergence (JS).

6.5.2. Dataset

We recorded 3D models of 26 household objects and used SimTrack (Pauwels and Kragic, 2015) to detect them and compute their poses in a scene using a Kinect camera. Using this setup, we recorded a set of demonstrations $\mathcal{D}$ consisting of 546 scenes, see Figure 6.4 for examples. For the purpose of evaluation, we manually labeled the similarities $\mathbf{Y}$ between all scenes.

6.5.3. Nearest Neighbor Classification

In this experiment, we evaluated the ability of distance metric learning to relate scenes based on the similarities of their relations. We formulated this as a k-NN classification problem with $k = 5$ and used 15 random splits for evaluation. For each split, we used 75% of the data for training and 25% for testing. We considered a success if at least three out of five of the retrieved nearest neighbors were similar to the test example.

The results are shown in Table 6.1. LMNN-based metrics outperform the baselines, i.e., the learned metrics can better capture the distances between scenes. We achieved the highest success rate of 87.6% using GB-LMNN. Additionally, by directly computing the Euclidean distance in the original feature space, we are able to achieve a success rate of 82.32%. This demonstrates that our proposed feature descriptor is suitable for encoding arbitrary spatial relations. Figure 6.5 shows a qualitative example of the nearest neighbors of a test scene using both the Euclidean distance and LMNN.

Table 6.1.: Performance of different methods for retrieving at least three out of five target neighbors of scenes (see Section 6.5.3). We achieved a success rate of 82.32% by directly computing the Euclidean distance in the original feature space based on our proposed descriptor. GB-LMNN resulted in the highest success rate by learning a non-linear metric capturing relation similarities.

Method	Accuracy(%)
Euclidean	82.32 ± 2.56
KL	82.61 ± 3.10
Correlation	82.66 ± 2.43
χ^2	82.81 ± 3.20
Bhattacharyya	83.26 ± 3.15
JS	83.30 ± 3.14
χ^2-LMNN	86.46 ± 2.84
LMNN	86.52 ± 1.98
GB-LMNN	$\mathbf{87.60 \pm 1.94}$

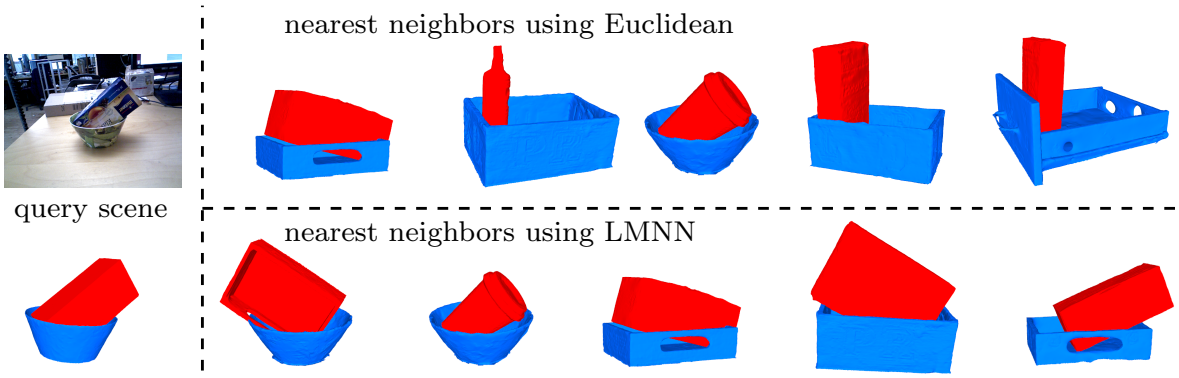


Figure 6.5.: Example of the five nearest neighbors of a query scene using Euclidean distance (top) and LMNN (bottom). LMNN better captures the distances between relations.

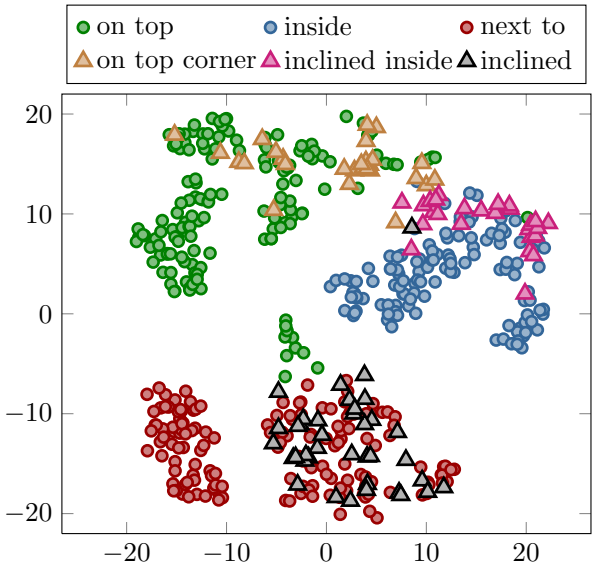


Figure 6.6.: The t-SNE visualization of six relations mapped using LMNN. We trained this metric using three of the six relations only (upper row of Figure 6.4), which we depict as circles. The metric is able to capture the semantic similarity between those relations and the test relations (lower row of Figure 6.4), which we depict as triangles.

6.5.4. Distance to New Relations

In this qualitative experiment, we investigated the ability of a learned metric to capture the similarities between relations that were not used for training. We trained LMNN with data from three relations (upper row of Figure 6.4), which can be semantically described as "on top", "inside", and "next to". We used the learned metric to map all six relations in Figure 6.4 to the new space and visualized the data using t-SNE, a popular non-linear embedding technique for visualizing high dimensional data (Maaten and Hinton, 2008). We show this in Figure 6.6. This qualitatively illustrates the separation between the three relations used for training the metric. Moreover, the metric is able to capture the semantic similarity between the relations used for training and the new ones, which we denote by "on top corner", "inclined inside", and "inclined" (bottom row of Figure 6.4).

6.5.5. Generalizing a Relation to New Objects

In this experiment, we evaluated our approach for reproducing a demonstrated relation using two new objects, see Section 6.1.3.3. We recorded 30 demonstrations for each of five new relations. We then selected two new objects that were not used in the demonstrations and evaluated our method's ability to generalize each of the five relations to the new objects. In each case, we provided our method with $|\mathcal{D}'| = 5$ examples. Using those

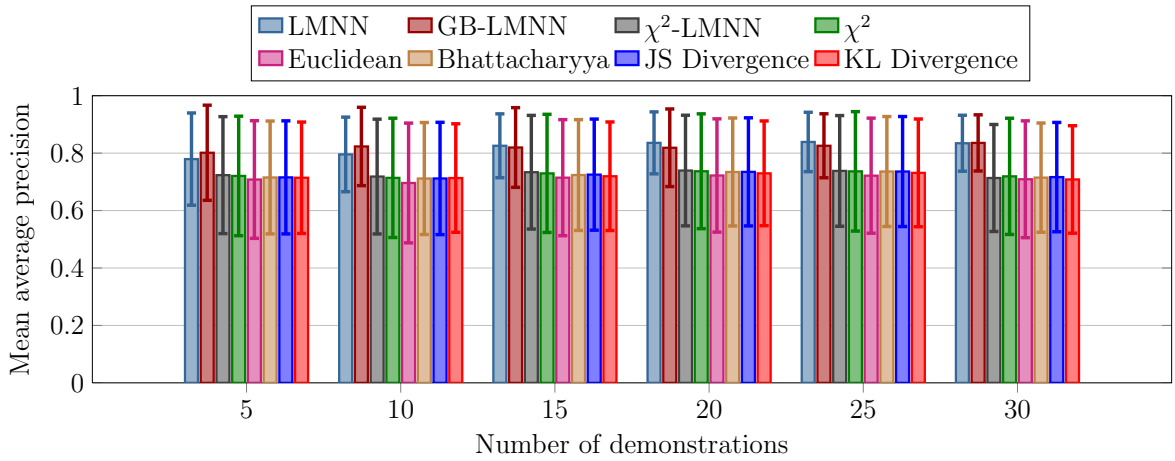


Figure 6.7.: Results for generalizing a relation to two new objects (see Section 6.5.5). GB-LMNN and LMNN outperform the other metrics in identifying the correct ways of reproducing a relation with two new objects. In each case, we evaluated the generalization given five new demonstrations, which we then added to the set of prior demonstrations $\mathcal{D}$ before relearning the prior metric. Our approach enables reusing previous demonstrations when querying the teacher to learn a new relation. Accordingly, GB-LMNN achieves a mean average precision of 0.82 after having seen five demonstrations in the past. This demonstrates the ability of our approach to use previous demonstrations when generalizing a new relation in a lifelong learning manner. Note that the x-axis shows the cumulative number of demonstrations over time.

examples, we retrieved nearest neighbor queries $\mathcal{D}_{NN}$ from $\mathcal{D}$ as described in Section 6.4. As we aimed for a quantitative evaluation in this experiment, we implemented an "oracle" that provides the binary labels for the queries automatically, and used this to learn a local metric as in Section 6.4.1.

To facilitate evaluation and comparison between the baselines, we provided our method with a set of 75 poses ${}^k\mathbf{T}_l$ between the new objects rather than sampling poses randomly. In each case, only 15 of the 75 poses were correct ways of imitating the relation in question using the new objects, whereas the rest corresponded to other relations. To evaluate the ability of our approach to recognize the correct poses, we computed the features $\mathbf{r}^*$ of the test scene for each pose and the corresponding loss $\mathcal{L}$, i.e., the minimum distance between $\mathbf{r}^*$ and the features $\mathcal{R}'$ of the five demonstrations, see Eq. (6.9). Accordingly, we sorted the poses according to $\mathcal{L}$. Ideally, the correct poses should be in the top 15 positions as they result in minimizing the distance between the test scene and the demonstrations. Therefore, we used the mean average precision of the computed ranking of poses as an evaluation measure.

After each such test, we added the five demonstrations $\mathcal{D}'$ to the database $\mathcal{D}$ and

extended $\mathbf{Y}$ with the new labels from the nearest neighbor queries. We used this to re-learn the prior metric dist_{ϕ_0}. We did this six times, each with five new demonstrations, until the 30 demonstrations have been used. We repeated the whole experiment 50 times using different random orders of providing five demonstrations. We report the results in Figure 6.7 averaged over all runs and five relations.

The metrics we learned with GB-LMNN and LMNN outperformed the other metrics in their ability to identify the correct ways of reproducing a relation with two new objects. Although in each case we computed Eq. (6.9) based on five new demonstrations only, our approach enables those metrics to reuse demonstrations added to $\mathcal{D}$ from the previous tests to learn the local metric. Accordingly, GB-LMNN achieved a mean average precision of 0.82 after having seen at least five demonstrations in the past. This demonstrates the ability of our approach to use previous demonstrations when generalizing a new relation in a lifelong learning manner.

6.5.6. Interactive Learning of a New Relation

We conducted a small survey to evaluate our approach for learning a new relation interactively. We asked nine different teachers to provide demonstrations of 50 relations. Each teacher provided five demonstrations per relation using different objects they chose, resulting in 250 recorded scenes in total. We used only three demonstrations $\mathcal{D}'$ to learn each relation. In each case, we queried the teacher with nearest neighbor examples from $\mathcal{D}$ (based on a prior LMNN metric we trained on $\mathcal{D}$) and used the result to generalize the three demonstrations $\mathcal{D}'$ and reproduce one of the test scenes we left out as in Section 6.4. We computed the best pose for reproducing the relation using the sample-based approach in Section 6.4.2.

We showed the reproduced relations to their corresponding teachers in a 3D visualization environment and asked them to score the quality of the result with 0, 0.5, or 1, where 0 represents unsuccessful and 1 represents successful. The teachers scored results from different baselines shown in random order and without knowing which baselines they were scoring.

We show the mean scores in Table 6.2. The reproduced scenes using both LMNN and GB-LMNN metrics were judged to be the best by the teachers, achieving an average score of 0.72 and 0.71 respectively. Figure 6.8 illustrates one such generalization from our experiments. The results confirm that our approach enables non-expert users to teach arbitrary spatial relations to a robot from a small number of examples.

As a final qualitative evaluation, we carried out a small survey in which we asked six participants to judge the quality of scenes generated by our sample-based approach in Section 6.4.2. For this, we selected 30 scenes computed by LMNN, and which were scored with either 0.5 or 1 in the experiment above. We asked the six participants to judge whether the relations in those scenes were demonstrated by a human or generated

Table 6.2.: Mean scores for reproducing 50 relations from nine different teachers. For each relation, the teacher scored the result with 0 (unsuccessful), 0.5, or 1 (successful), see Section 6.5.6.

Method	Score
Euclidean	0.49
Bhattacharyya	0.54
KL	0.56
JS	0.56
χ^2	0.60
χ^2-LMNN	0.63
LMNN	**0.72**
GB-LMNN	0.71

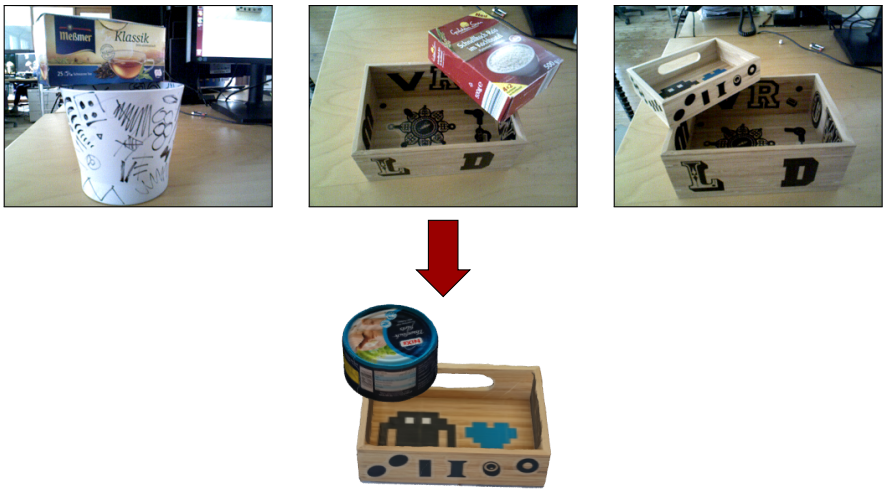


Figure 6.8.: Top: three demonstrations of a relation by a teacher in our survey, see Section 6.5.6. Bottom: the result of reproducing the relation with two new objects using our approach.

by a computer, whereas in fact they were all computed using our approach. Despite the fact that we do not consider physics checks (e.g., scene stability) or optimize computed scenes to make them more realistic, the participants judged 63.05% of the scenes to be produced by a human. This highlights the ability of our approach to reproduce relations using different objects and compute high-quality scenes.

6.6. Related Work

In this chapter, we presented a novel method to tackle the problem of learning and generalizing pairwise spatial relations. In this section, we review the most related works in the literature in the context of learning and leveraging spatial relations for robotic tasks and in the field of distance metric learning.

Learning and Using Spatial Relations Spatial relations have played a large role in the context of robot manipulation. A large body of works has focused on leveraging predefined relations in the form of symbolic predicates for solving tasks, as in the case of combined task and motion planning (Dornhege et al., 2012; Kaelbling and Lozano-Pérez, 2013). Such works rely on an expert to provide relations as high-level state abstractions, as well as the mapping between these relations and low-level (geometric) states, to reason about the feasibility of plans.

On the other hand other approaches have focused on learning the mapping between existing symbolic predicates and low-level state representations in terms of geometric object configurations. In this context, Dearden and Burbridge (2014) proposed an approach to learn classifiers for forward and backward mapping between symbolic states and geometric ones based on object poses. For training, they rely on positive and negative examples of predicates (e.g., `above`, `touching`), and are thus able to leverage the learned mappings for high-level planning to satisfy both symbolic and geometric constraints. Related to that, Guadarrama et al. (2013) learn models of predefined prepositions by training a multi-class logistic regression model using data gathered from crowdsourcing. As opposed to those works, we do not address grounding existing symbols that represent spatial relations. Instead, we propose an approach to lifelong learning of relations and their similarities on a continuous spectrum without requiring an expert to specify explicit concepts or classes.

Instead of leveraging existing relations, other works have addressed learning symbols or preconditions and effects of actions to abstract continuous states for the purpose of high-level planning (Abdo et al., 2013; Jetchev et al., 2013; Konidaris et al., 2014; Ahmadzadeh et al., 2015). Moreover, Welke et al. (2013) presented an approach to infer and ground spatial symbols based on a robot's experience in exploring an environment or using human knowledge extracted via natural language processing of text. This enables the robot to learn distributions that model typical locations (e.g., fridge) and associated spatial relations in an environment that can be used for high-level task planning. By learning such symbolic representations, the above works are able to scale to large domains, for example by grounding a relation using all applicable objects of the same type in the environment. In contrast to these works, we do not address learning state abstractions that can be used to achieve desired goals using high-level planning. Instead, our approach encodes relations on a geometric level and without relying on

semantic information about objects, and is thus able to imitate a relation using new objects by minimizing the distance between scenes in feature space.

Moreover, related to our work is the interactive approach by Kulick et al. (2013) for learning relational symbols from a teacher. They use Gaussian Process classifiers to model symbols based on properties such object poses and dimensions, and enable a robot to query the teacher with examples to increase its confidence in the learned models. They demonstrate the applicability of such learned symbols in the domain of relational reinforcement learning. Similarly, our method enables a robot to learn a relation by interacting with a teacher. However, we address this problem from the perspective of metric learning, thereby allowing the robot to re-use previous demonstrations of other relations in a lifelong manner. Moreover, using our proposed descriptor for pairwise relations, we achieve a great flexibility by generalizing the same relation to objects of varying shapes and sizes.

Similar to our work, other approaches addressed learning and recognizing relations using low-level features such as object geometries. For example, Rosman and Ramamoorthy (2011) proposed encoding a scene topologically by constructing a contact point network based on regions of contact between objects. They leverage these networks to obtain a feature representation of a scene based on the displacement between contact points, and accordingly rely on labeled training data to train (SVM) classifiers to identify relations (e.g., `adjacent`) in new scenes. Related to that, Fichtl et al. (2014) presented an approach that extracts surface patches of objects in segmented images and thus constructs several histograms that encode the relative distances and orientations between those patches. This representation is then used to train random forest-based classifiers to recognize relations and affordance concepts in images. In contrast to these works, our approach does not require training data that is labeled according to predefined relation categories. Instead, we learn a metric that applies to a variety of relations. This allows us to learn a new, arbitrary relation based on a small number of interactions with a non-expert teacher and the subjective similarity labels she provides.

Furthermore, rather than learning relations, other works leveraged existing spatial relations and 3D object models to tackle a variety of manipulation and perception problems. In the context of scene interpretation, the work of Ersen et al. (2014) relies on predefined relation predicates for failure detection during task execution, e.g., by recognizing a failed stacking of blocks based on the observed value of the `on` predicate. Similar to our work, the approach of Zampogiannis et al. (2015) models spatial relations based on the geometries of objects given their point cloud models. However, they manually define a variety of common relations and focus on extracting the semantics of manipulation actions through temporal analysis of spatial relations between objects. Other works have used predefined spatial relations to learn models of typical object configurations or placements that can be leveraged for object search (Kunze et al., 2014; Lorbach et al., 2014) or to reason about user preferences when organizing objects (Abdo

et al., 2016). Related to that, other researchers relied on the geometries of objects and scenes to reason about preferred object placements (Jiang et al., 2012b) or likely places to find an object (Aydemir and Jensfelt, 2012). Finally, Kroemer and Peters (2014) presented an approach that uses 3D object models to extract contact point distributions for predicting object interactions e.g., when stacking or grasping objects.

Distance Metric Learning Our approach leverages distance metric learning in order to reason about the similarity between relations. Specifically, we built our work on the paradigm of large margin nearest neighbor (LMNN), a widely-used metric learning approach originally introduced by Weinberger and Saul (2009) to improve the performance of k-NN classification. Metric learning is a popular paradigm in the machine learning and computer vision communities, and has been successfully used to tackle problems related to face recognition (Guillaumin et al., 2009) as well as image retrieval and segmentation (Hoi et al., 2008; Xiang et al., 2008). We refer the reader to the surveys by Bellet et al. (2013) and Kulis et al. (2013) for a more comprehensive overview. Furthermore, metric learning is closely related to the problems of embedding learning and dimensionality reduction for mapping high-dimensional points to low-dimensional spaces that preserve point similarities and distances (Roweis and Saul, 2000; Hadsell et al., 2006; Maaten and Hinton, 2008).

To encode relations, we proposed a novel descriptor based on object point cloud models and used both linear (Weinberger and Saul, 2009) and non-linear (Kedem et al., 2012) metric learning approaches to map scenes from the original feature space to one that reflect the similarities between the corresponding spatial relations. Rather than relying on designed features, several researchers have leveraged recent advances in deep learning to train neural networks that are able to learn the similarities between images in an end-to-end fashion (Chopra et al., 2005; Yi et al., 2014; Hoffer and Ailon, 2015).

Finally, our approach represents a novel solution to the problem of imitating a desired spatial relation by using a learned metric to retrieve previously-learned relations that are similar to it. In the context of robotics, metric learning has also been used to address object instance recognition (Lai et al., 2011b) and to retrieve best-match images of places for loop closure detection (Shahid et al., 2016). To address the problem of grasp selection, Herzog et al. (2014) recently proposed a local shape descriptor to encode grasp regions of object parts. Accordingly, they introduce distance metrics that enable a robot to retrieve promising grasp hypothesis by matching perceived point clouds to a library of learned grasp templates.

6.7. Conclusion

In this chapter, we presented a novel approach to tackle the problem of learning pairwise spatial relations and generalizing them to different objects. Our work is based on distance metric learning and enables a robot to reason about the similarity between pairwise spatial relations. In this work, we assumed to have no semantic knowledge about objects. Therefore, we introduced a novel descriptor to encode relations based on the geometries and poses of the involved objects. By learning a distance metric using this representation, our method is able to bootstrap learning a new relation from a small number of teacher demonstrations by reasoning about its similarity to previously-encountered ones. The learned metric also enables us to formulate the problem of generalizing a relation to new objects as one of computing object poses to minimize the distance between two scenes. Accordingly, our approach allows for reasoning about a continuous spectrum of relations in a lifelong learning scenario. This goes beyond existing approaches that learn a model (e.g., a classifier) for each relation individually, and that are therefore limited to reasoning about a finite number of relations. Finally, our evaluation with real-world data demonstrates the effectiveness of our approach in reasoning about the similarity between relations and its ability to reproduce arbitrary relations to new objects by learning interactively from non-expert teachers.

7. Conclusion

In this thesis, we presented several contributions to the fields of service robotics and learning from demonstrations. Our contributions revolved around the challenge of enabling robots to solve manipulation tasks in human-centered environments. Specifically, we considered learning goal-oriented actions and tasks that are modeled based on desirable spatial relations between objects. The novel approaches we presented in this thesis all share the same underlying philosophy: *i)* to act effectively and intelligently, service robots should consider the preferences of their users, *ii)* to handle the wide spectrum of everyday objects and tasks in domestic environments, service robots should rely less on expert design as a source of knowledge, and more on lifelong learning through interacting with their users and environments, and *iii)* when faced with new situations, service robots should be able to adapt their behavior and improvise solutions rather than constantly querying their users.

We first tackled the challenge of learning user-preferred task goals in the context of organizing or sorting objects. Typically, each user has her own preferred way of solving such tasks, making it challenging for an expert to pre-program a robot with the knowledge needed to attend to all users. We proposed a novel solution to this problem by leveraging the paradigm of collaborative filtering to enable a robot to tailor its behavior to the preferences of a specific user when organizing objects in containers. Our contribution is the first to bridge the gap between the worlds of personalized recommendations and service robotics and enables robots to learn patterns of user preferences with respect to grouping pairs of objects when tidying up. We learn these preferences in a lifelong manner from a corpus of data gathered from different users and environments. Using the learned preference models, a robot is able to autonomously predict the preferred way to solve the task for its user based on partial knowledge of her preferences or a small number of observations of the environment. Accordingly, the robot is able to organize the objects by maximally satisfying the preferences of the user without constantly querying her, and to adapt to the number of containers available for tidying up. Additionally, we addressed the challenge of handling objects for which there is no data in the system by presenting a novel method that complements collaborative filtering with expert knowledge mined from the Web. We extensively evaluated our approach on two tidy-up scenarios using data we gathered from over 1,200 users. Our results demonstrate the effectiveness of our approach in predicting user preferences and its ability to improve its performance over time given more user data in the system.

The above contribution enables a robot to infer preferred task goals in the form of object arrangements and to subsequently achieve those goals by relying on existing task and motion planners for manipulating the objects. However, this limits the robot to handling objects for which an expert has preprogrammed action models, for example in the form of pick-and-place planning operators. Therefore, we additionally tackled the challenge of enabling a robot to acquire new action models by learning from demonstrations provided by non-expert teachers. Specifically, we addressed learning point-to-point actions that are modeled based on spatial constraints between the objects before and after applying an action. As we aim to learn these actions using a small number of demonstrations, a key challenge in this context is inferring the intention of the teacher with respect to which features (spatial constraints) are relevant for modeling a new action. This knowledge is crucial to successfully generalize the action and to sequence several actions when planning. We tackled this issue from two perspectives. Firstly, we proposed a model-selection approach by which the robot leverages prior expert knowledge about manipulation actions to bootstrap learning the new action. Accordingly, the robot is able to maintain several valid interpretations, or *templates*, of the teacher demonstrations and to apply the best template based on the current state. Secondly, we proposed a novel approach that does not require the availability of such prior expert knowledge and instead learns *implicit action models* in a data-driven manner based on a mixture of interpretations. This allows the robot to construct a multi-modal distribution over likely goal states when reproducing the intention of the teacher in new states. Using our techniques, the robot avoids the need to commit to a single model of the action beforehand, thereby achieving flexibility and generalization. Equally important, in contrast to the majority of existing solutions for learning actions, our contributions eliminate the need for an expert to specify the relevant spatial constraints for modeling each new action. Through experiments with real-world data, we demonstrated the effectiveness of our method and its ability to successfully reproduce an action in arbitrary initial states.

Furthermore, a key contribution of this thesis is the *teach-and-improvise* framework (TI), a novel approach to the problem of simultaneously inferring action and task models from a small number of teacher demonstrations. Whereas the above contributions focused on either learning semantically preferred task goals or new action models individually, TI enables a robot to acquire models of both in order to reproduce sequential manipulation tasks without assuming prior semantic knowledge about the task or the involved objects. By drawing similarities between inferring the intention of the teacher and solving Markov decision processes, we formulated solving a task as an optimization problem and accordingly presented a novel algorithm that builds on Monte Carlo tree search to generalize and sequence learned actions and achieve desirable goals in an anytime manner. Our algorithm leverages implicit action models learned from the teacher and tackles the ambiguity in the demonstrations through decision making

on three levels: which action to apply in the current state, which action model to use, and which goal state to achieve. Our work eliminates the need for the user or an expert to provide an explicit goal state to solve a task. Instead, TI reasons about multiple modes in desirable spatial relations to *improvise* feasible solutions that align with the intention of the teacher with respect to each applied action and the final goal state in a probabilistic framework. This enables the robot to autonomously adapt the solution of the task based on the initial state and physical constraints without querying the user in each new situation. Moreover, we proposed a novel approach that allows the robot to update the learned action models and improve its performance with time by practicing the task in simulation. We evaluated our approach extensively in simulation and with a real robot and demonstrated its ability to generalize a small number of demonstrations to achieve desirable goals starting in arbitrary states.

At the heart of the above approaches are spatial relations between everyday objects. Specifically, we adopted pairwise spatial relations as the building blocks for modeling preferred task goals or relevant constraints for applying actions. For this, we either relied on existing expert models, e.g., in the form of high-level predicates for planning, or, when lacking prior semantic knowledge, constructed multi-modal distributions of relations based on relative poses between objects. However, this limits the robot to solving tasks using the objects that are supported by its prior knowledge as programmed by an expert or demonstrated by a teacher. In other words, this renders the robot unable to reproduce learned tasks and actions using new objects of different shapes and sizes. The standard solutions to this problem require the user or an expert to provide the robot with large amounts of labeled training data to learn a model for each new relation. This restricts the robot to reasoning about a limited number of relations. We addressed this problem by presenting a novel approach to learning and generalizing pairwise spatial relations from the perspective of distance metric learning. We proposed a novel descriptor that encodes a spatial relation geometrically based on 3D models of the objects and their relative poses. Using this representation, we leveraged a popular distance metric learning technique to learn a metric capturing the similarities between spatial relations. For this, we relied on training data describing how similar relations are to each other, which the robot accumulates by interacting with non-expert users over time. Accordingly, the learned metric allows the robot to bootstrap learning a new relation by relying on a small number of examples to retrieve similar ones from its database. The metric also enables the robot to imitate the relation using new objects by minimizing the distance between the resulting relation and the desired one. Therefore, our contribution goes beyond existing approaches by allowing the robot to reason about a continuous spectrum of spatial relations. We demonstrated the effectiveness of our approach in generalizing arbitrary relations to new objects through extensive evaluation with real-world data acquired from different users.

In summary, we proposed in this thesis several contributions that enable robots to

continuously acquire models of spatial relations, actions, and tasks by interacting with non-expert users and their environments. Our solutions provided, for the first time, a principled way of reasoning about and attending to the preferences of end-users with respect to everyday manipulation tasks. Moreover, our work extends the state of the art by allowing robots to reason about different valid ways of applying actions or solving tasks and to adapt their behavior as needed without requiring users to constantly provide explicit goals for planning. This flexibility is crucial in the context of dynamic, domestic environments. By placing the user at the center of our solutions, the contributions of this thesis relieve experts from the tremendous burden of pre-programming robots with the knowledge to handle all possible situations, thereby bringing us closer to deploying intelligent service robots in human-centered environments.

Outlook

There are several ways by which future research can extend the scope and capabilities of the approaches we proposed in this thesis. In general, our collaborative filtering approach for learning user preferences is agnostic to the specific relations or goals being considered. Whereas we focused on organizing objects in containers in this work, it would be straightforward to extend our system to other everyday tasks such as cleaning, setting a table, or elderly care. By leveraging expert-defined relations or goals based on prior knowledge of these tasks, our approach can be used to extract patterns of preferences for these domains by interacting with users over time. Moreover, we considered learning user preferences on a semantic level that does not require defining features to model objects. However, it would be beneficial to extend our recommender system to consider low-level geometric or visual features to achieve a more fine-grained prediction of user preferences. By observing different environments over time, this would allow a robot to extract patterns such as placing large objects at the back of a shelf or grouping objects based on color. This results in a hybrid recommender system combining aspects of collaborative filtering with content-based information.

Moreover, in this thesis we addressed learning point-to-point actions from segmented teacher demonstrations and focused on inferring the relevant features or spatial constraints for modeling the action before and after applying it. To execute the action in the learned feature space, we relied on existing motion planning and control solutions. However, it would also be possible to use the teacher demonstrations to learn preferred motions to reproduce the action on the trajectory level. This could be achieved using existing techniques for encoding trajectories as dynamic movement primitives. Having inferred the suitable template or reference frames for applying an action using our work, one would then transform the learned motion model to the corresponding frame to reproduce the intention of the teacher with respect to both the goal state and the

trajectory to achieve it. This would extend the scope of our work to tasks in which the dynamics of an action are relevant for its success such as when opening a door or moving a glass of water.

Furthermore, our TI approach is currently able to reproduce a task using the same objects demonstrated by the teacher or a subset thereof. This is due to our model of pairwise spatial relations in the form of distributions over relative poses between two objects. On the other hand, our distance metric learning approach enables a robot to generalize a pairwise spatial relation to objects of different shapes and sizes. Therefore, incorporating the learned metric in the TI framework would allow the robot to solve a task using objects that were not demonstrated by the teacher. This could be achieved by modifying the action goal distributions and the intention likelihood model of the task to consider the distances between (current and desirable) relations under the learned metric rather than the distances between poses.

Finally, our proposed teach-and-improvise approach assumes to have no expert or semantic knowledge of the task and thus aims to maximize the intention of the teacher with respect to spatial relations on a geometric level. On the other hand, our collaborative filtering approach leverages known object types and existing high-level relations to scale to several objects of the same type and abstract away from object poses when reasoning about user preferences. Combining both worlds would allow our TI approach to scale to larger domains by leveraging high-level state abstractions when maximizing the intention of a teacher. In this context, it would be interesting to investigate techniques for learning symbolic abstractions from non-expert users in a lifelong manner and without requiring experts to design or provide such knowledge beforehand. This would allow the robot to bootstrap inferring the relevant relations for a new task based on preference models learned from previous users on a high level and at the same time to acquire new relations and geometric models from the teacher demonstrations. To solve tasks using compact relational representations, a promising domain of research is relational reinforcement learning, which combines aspects of reinforcement and relational learning.

Appendices

A. Additional Teach-and-Improvise Experiments

In this appendix, we present additional experimental results and visualizations related to our teach-and-improvise approach in Chapter 5.

A.1. Updating Action Models from New Experiences

We qualitatively investigate our approach for updating action models using the robot's experience in solving the task (see experiment in Section 5.7.6). For this, we used the plans we computed in the experiment in Section 5.7.5 (using the noisy action models learned from the teacher). Using our approach in Section 5.6, we updated the relevance of each action template and the corresponding voting distribution to reflect the usefulness of each template and vote for solving the task, respectively.

Figure A.1 shows an example of the average template selection rates for two actions in the tower building task ($\mathcal{T}_3$) based on all computed plans in Section 5.7.5 when learning from six task demonstrations. Despite ambiguity in the teacher demonstrations and initializing all template relevance likelihoods uniformly, our approach was able to select the action templates (stationary objects) that are most useful for applying those actions more frequently. Specifically, our method selected the stationary blocks that typically appear beneath the moving block of each action since the corresponding votes result in stacking the blocks and constructing the tower. For example, for the action of moving the dark blue block (second block from the bottom in the tower), our approach selected template γ_1^2 in 93.9% of the cases. This corresponds to the red block (bottom of the tower), since its votes result in stacking the two blocks (Figure A.1-top). Similarly, for the action of moving the top-most block, our approach relied on the blocks that typically appear beneath it in the goal states (Figure A.1-bottom). Moreover, we selected the template corresponding to the table in 0% of the cases, which is consistent with the irrelevance of the poses of the blocks relative to the table in this task.

Figure A.2 shows an example of updating the goal distribution for the action of moving the end-effector in the breakfast setting task ($\mathcal{T}_1$) using our approach and after solving the task 100 times starting in random initial states. The resulting distribution (Figure A.2-bottom) is more peaked around goal states in which the end-effector achieves

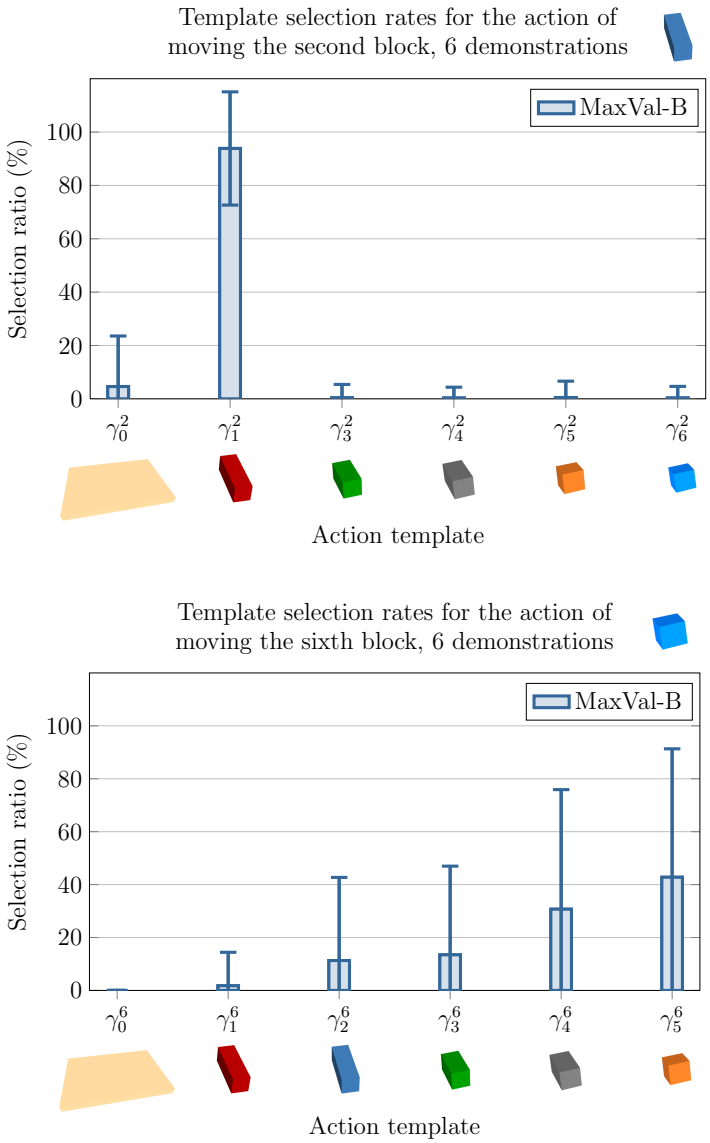


Figure A.1.: Average action template selection rates for two actions in the tower building task ($\mathcal{T}_3$). We extracted these statistics from plans computed using our approach when solving the task starting in 2,500 random initial states. Despite initializing the template relevance likelihoods uniformly for all actions, our approach is able to leverage templates that are more useful for solving the task. The votes corresponding to these templates encourage stacking the moving block in each case by placing it relative to blocks that typically appear beneath it in the tower (see Figure 5.8). Using these selection rates, our approach updates the learned action goal distributions to reflect the usefulness of each action template and associated votes for solving the task and without requiring additional teacher demonstrations.

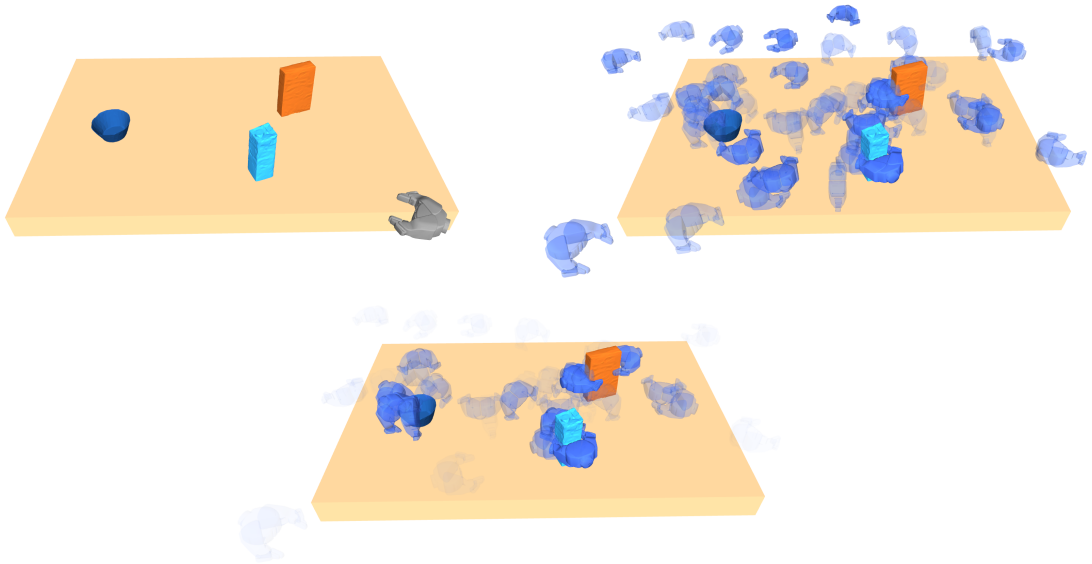


Figure A.2.: An example of the action for moving the end-effector in the breakfast setting task ($\mathcal{T}_1$). Top left: a random initial state. Top right: the corresponding distribution of goal poses for the end-effector. The teacher moved the end-effector to reach for different objects and at different time steps of the task demonstrations. This introduces ambiguity in the intention of the teacher for this action. Bottom: after solving the task 100 times in simulation, we update this distribution to reflect the intention of the teacher more accurately. The result is a higher likelihood of moving the end-effector to reach for the relevant objects in the scene.

grasping poses relative to the other objects. This captures the intention for the action more accurately compared to the noisy distribution learned directly from the ambiguous demonstrations (Figure A.2-top right).

A.2. Solving Tasks Using the Updated Action Models

In this section, we present detailed experimental results for solving the aligning blocks and tower building Tasks ($\mathcal{T}_2$-$\mathcal{T}_3$) after allowing our approach to practice those tasks in simulation and to update the action models accordingly, see Section 5.7.7.

A.2.1. Results for the Aligning Blocks Task ($\mathcal{T}_2$)

We evaluated our approach for solving the aligning blocks task ($\mathcal{T}_2$) starting in 100 random initial states using the updated action models. We repeated this 25 times using different sets of demonstrations in each case. Using MaxVal-B*, our approach is able to

satisfy between 93.6% and 99.1% of the relevant relations for this task, see Figure A.3-top left. The updated action goal distributions result in high success rates for all numbers of demonstrations. In contrast, MaxVal-B (before updating the action models) was only able to satisfy 94% of the relations on average when learning from two demonstrations, and its performance decreased with more demonstrations due to the increased branching factor when sampling from noisy action models (see Figure 5.14-top right).

Moreover, Figure A.4-top right shows the results of our approach using MaxVal-B* (0.5), i.e., using the updated action models with a max-value strategy but relying on only 50% of the templates of each action during node expansion. Despite sampling only half of the possible templates for each action based on their relevance likelihoods, our approach is able to satisfy between 98% and 99.6% of the relevant relations for this task. By selecting between the sampled templates based on the resulting values, our algorithm leverages different interpretations of each action based on the current state when applying it.

When using a Bellman update strategy, our approach (Bellman-ϵ*) was able to satisfy between 88.1% and 92.3% of the relations on average, see Figure A.3-bottom left. Compared to a max-value strategy, Bellman-ϵ* seeks to maximize the expected value of the solution, thereby applying actions based on the probability of their goal states, and not only based on the intention likelihood of the final goal state. Therefore, MaxVal-B* is more robust to the (remaining) noise in the action models, and is able to satisfy more relations than Bellman-ϵ* given the same number of iterations.

Additionally, Figure A.3-bottom right highlights the differences between those variants of our approach when learning from six demonstrations of this task. When using the updated action models (MaxVal-B*, MaxVal-B* (0.5), and Bellman-ϵ*), the performance of our approach improves compared to that before practicing the task (MaxVal-B). By considering only the intention likelihood of the final goal state, MaxVal-B* is less sensitive to noise in the action goal distributions, and therefore outperforms Bellman-ϵ* with respect to the relations satisfied in the final state given the same number of iterations. When sampling only half the available action templates during node expansion, MaxVal-B* (0.5) is still able to solve the task and satisfy 99.6% of the relations. Due to the decreased branching factor at template-selection nodes, it also exhibits a faster convergence rate compared to MaxVal-B*.

Figure A.4 shows the corresponding plan lengths computed by our approach for this task. Before practicing the task MaxVal-B (Figure A.4-top left) computed plans consisting of 17.5 actions on average when learning from two demonstrations and satisfied 94% of the relevant relations. With more demonstrations, this strategy suffers from an increasing branching factor when sampling from the noisy action models and was able to sequence up to 13.3 actions to satisfy only 81.6% of the relations. On the other hand, using the updated action models, MaxVal-B* required between 13.7 and 14.3 actions to satisfy 93.6% and 99.1% of the relations using two and ten demonstrations, respectively.

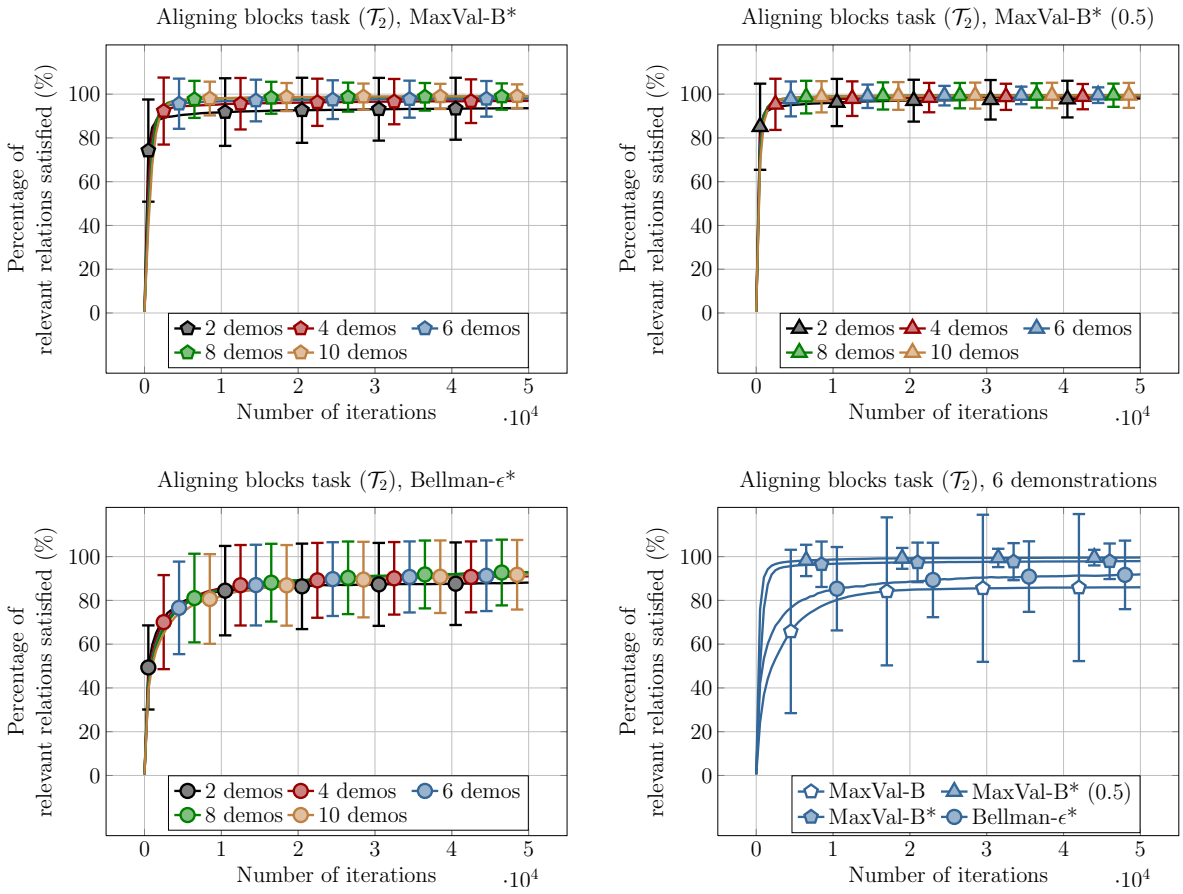


Figure A.3.: Results of our approach for solving the aligning blocks task ($\mathcal{T}_2$) after practicing the task and updating the action goal distributions (compare to Figure 5.14-top right). Top left: using MaxVal-B*, our approach is able to satisfy between 93.6% and 99.1% of the relevant relations for this task. Top right: despite sampling only half of the possible templates for each action based on their relevance likelihoods, MaxVal-B* (0.5) is able to satisfy between 98% and 99.6% of the relevant relations by leveraging different interpretations of each action. Bottom left: Bellman-ϵ* was able to satisfy between 88.1% and 92.3% of the relations on average. Bottom right: performance of different variants when learning from six demonstrations. When using the updated action models (MaxVal-B*, MaxVal-B* (0.5), and Bellman-ϵ*), the performance improves compared to that before practicing the task (MaxVal-B). By considering only the intention likelihood of the final goal state, MaxVal-B* outperforms Bellman-ϵ* with respect to the relations satisfied in the final state but at the expense of computing more expensive plans, see Figure A.4. When sampling only half the available action templates during node expansion, MaxVal-B* (0.5) is still able to solve the task and satisfy 99.6% of the relations.

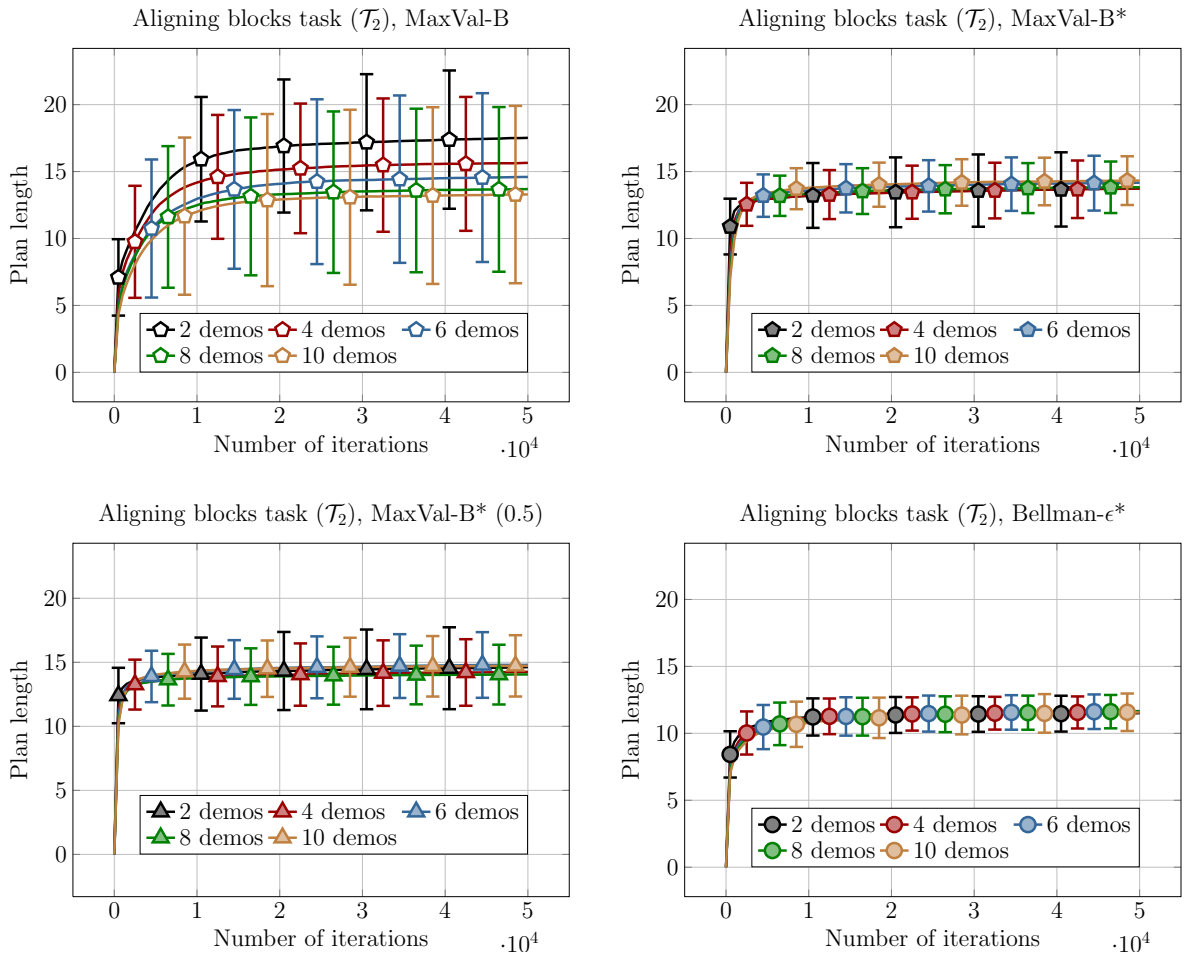


Figure A.4.: The average plan lengths when solving the aligning blocks task ($\mathcal{T}_2$). Top left: before practicing the task MaxVal-B required 17.5 actions on average when learning from two demonstrations and satisfied 94% of the relevant relations. With more demonstrations, this strategy suffers from an increasing branching factor when sampling from the noisy action models and was able to sequence up to 13.3 actions to satisfy only 81.6% of the relations (see Figure 5.14-top right). Top right: using the updated action models, MaxVal-B* required between 13.7 and 14.3 actions to satisfy 93.6% and 99.1% of the relations using two and ten demonstrations, respectively. Bottom left: since MaxVal-B* (0.5) considers only half the templates for each action during node expansion, this method requires applying more actions (between 14.1 and 14.8) to solve the task compared to MaxVal-B*. Bottom right: as Bellman-ϵ* maximizes the expected likelihood of the solution, this strategy computed more efficient plans (11.5 - 11.7 actions) compared to a max-value strategy.

Moreover, since MaxVal-B* (0.5) considers only half the templates for each action during node expansion, this method requires applying more actions to solve the task compared to MaxVal-B*, and computed plans ranging between 14.1 and 14.8 actions on average, see Figure A.4-bottom left. Finally, as Bellman-ϵ* maximizes the expected likelihood of the solution, this strategy computed more efficient plans of 11.5 - 11.7 actions on average (Figure A.4-bottom right). In other words, a Bellman-based approach is less likely to apply actions that result in goal states with low probabilities.

A.2.2. Results for the Tower Building Task ($\mathcal{T}_3$)

We evaluated our approach for solving the tower building task ($\mathcal{T}_3$) starting in 100 random initial states using the updated action models. We repeated this 25 times using different sets of demonstrations in each case. We show the results in Figure A.5. Using MaxVal-B*, our approach was able to compute plans to stack the blocks and achieve between 60% and 65.7% of the desired relations for this task (Figure A.5-top left). In comparison, before practicing the task, MaxVal-B satisfied between 48% and 52.2% of the relations on average (Figure 5.14-bottom left).

When allowing our algorithm to sample only 50% of the templates for each action during node expansion, MaxVal-B* (0.5) was also able to solve the task to satisfy between 56.2% and 63.5% of the relevant relations (Figure A.5-top right). Despite approximating each action's goal distributions using only half the stationary objects in the scene, MaxVal-B* (0.5) achieves a similar performance to that of MaxVal-B*. This demonstrates that the learned action template relevance distributions after practicing the task (Figure A.1) do in fact reflect the useful ways of applying each action. Our approach is thus able to adapt and sample a subset of the templates in each iteration based on their relevance.

Additionally, when using a Bellman criterion, Bellman-ϵ* was able to compute solutions that satisfy between 59.2%-64.7% of the relations (Figure A.5-middle left). When learning from ten task demonstrations, MaxVal-B*, MaxVal-B* (0.5), and Bellman-ϵ* achieved a similar success rate of 65.6%, 62.8%, and 64.7%, respectively. For this, MaxVal-B* and MaxVal-B* (0.5) required 10 and 10.3 actions on average, respectively. On the other hand, Bellman-ϵ* computed more efficient plans consisting of only 9.12 actions on average. By optimizing the expected intention likelihood, the Bellman-based approach is less likely to satisfy irrelevant relations, and therefore tends to favor actions with a high goal probability. For instance, 5.5% of the actions applied by MaxVal-B* in the solution plans corresponded to the action of moving the lower-most block in the tower before stacking blocks on it. On the other hand, this action constituted only 2% of the plans computed by Bellman-ϵ* on average since the exact position of the tower on the table is irrelevant for the task.

Furthermore, we highlight the fact that the teacher demonstrations for this task did

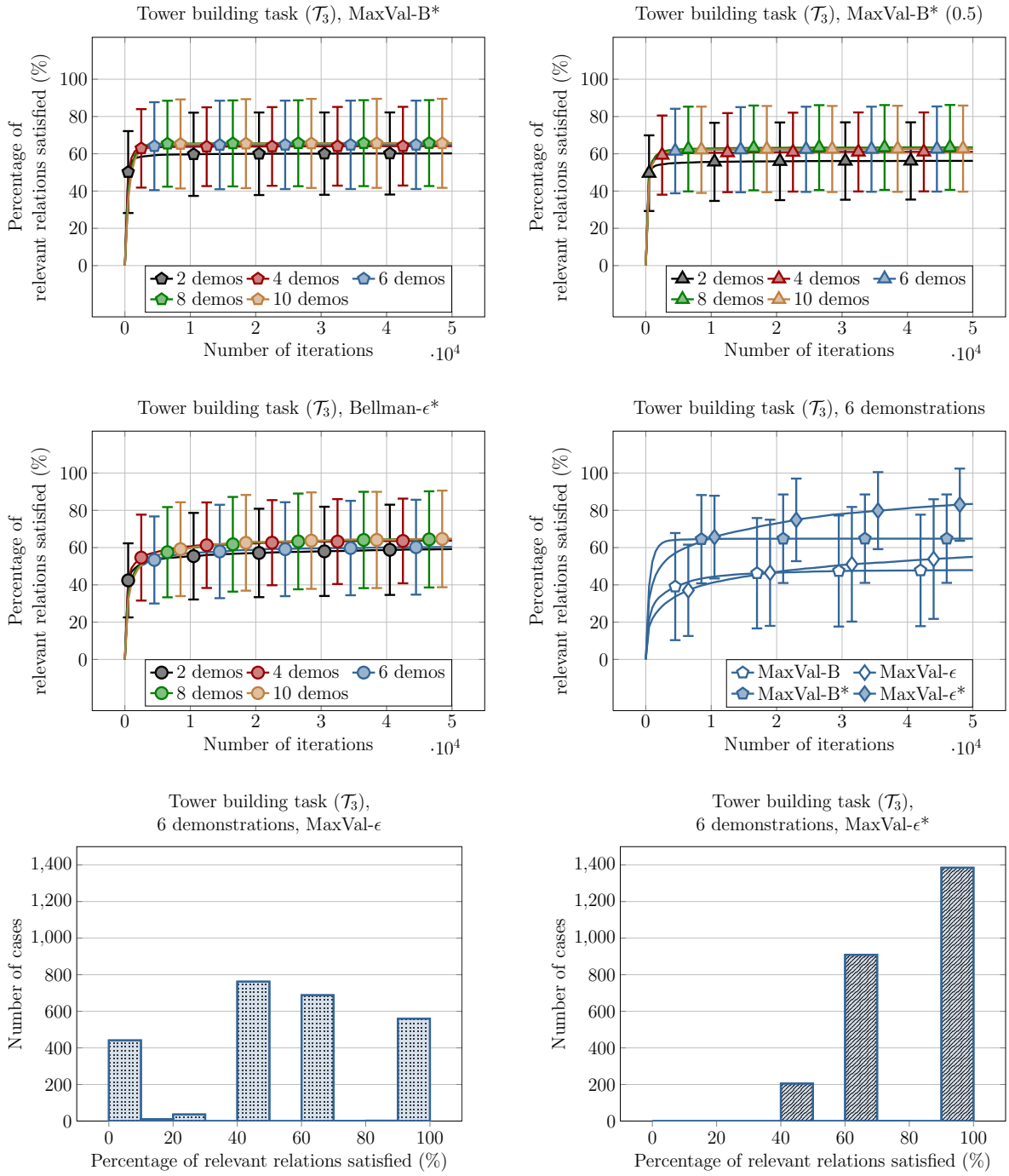


Figure A.5.: Results for the tower building task ($\mathcal{T}_3$) after practicing the task and updating the action goal distributions. See Section A.2.2 for details. Bottom row: the distribution of satisfied relations using MaxVal-ϵ over 2,500 test cases when learning from six demonstrations before (left) and after (right) practicing the task. MaxVal-ϵ constructed towers satisfying 60%-70% and 90%-100% of the relations in 27.6% and 22.4% of the cases, respectively. Using the updated action models, MaxVal-ϵ* achieved 60%-70% and 90%-100% of the relations in 36.4% and 55.4% of the cases, respectively.

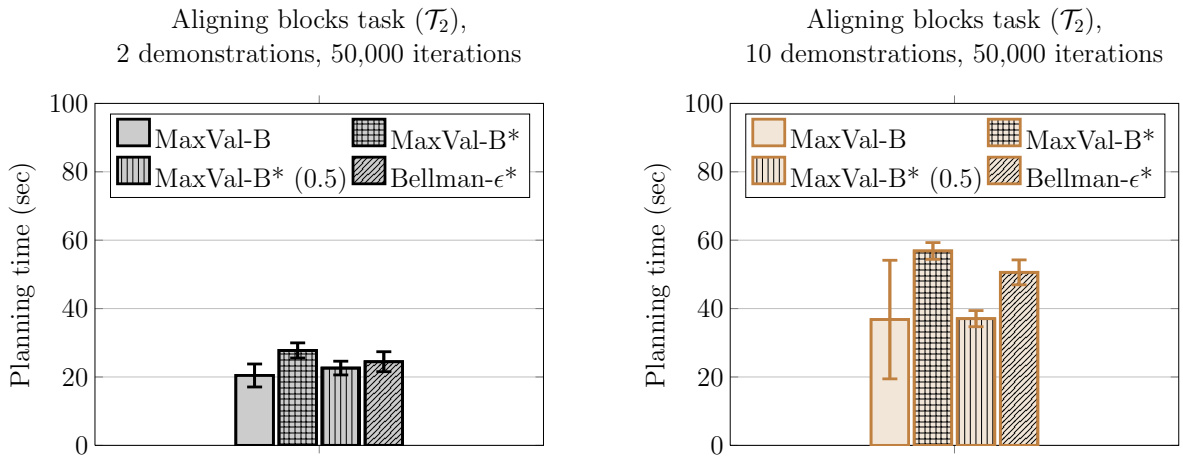


Figure A.6.: The planning times of our approach after 50,000 iterations when solving the aligning blocks task ($\mathcal{T}_2$) given two (left) and ten (right) demonstrations.

not include actions for unstacking blocks by moving them back to the table for example. This makes it challenging to recover from suboptimal solutions and explore alternative solutions during the tree search. Nonetheless, the above three variants of our approach computed solutions that satisfy between 56.2% and 65.7% of the relevant relations over all numbers of demonstrations on average. This corresponds to constructing 1-2 towers of various heights, see Figure 5.15 for examples. For this task, we achieved the best performance with respect to the number of satisfied relations when using a max-value strategy and an ϵ-greedy exploration criterion. We show this in Figure A.5 for the case of six demonstrations, in which MaxVal-ϵ* was able to satisfy 83.4% of the relations on average using the updated action models.

Finally, the bottom row of Figure A.5 shows the distribution of satisfied relations using MaxVal-ϵ over the 2,500 cases we tested before (left) and after (right) practicing the task for the case of six demonstrations. MaxVal-ϵ constructed towers satisfying 60%-70% and 90%-100% of the relations in 27.6% and 22.4% of the cases, respectively. Using the updated action models, MaxVal-ϵ* achieved 60%-70% and 90%-100% of the relations in 36.4% and 55.4% of the cases, respectively.

A.3. Planning Time

In this section, we report selected planning times (after 50,000 iterations) of our approach for the aligning blocks and tower building tasks ($\mathcal{T}_2$ and $\mathcal{T}_3$), which we show in Figures A.6 and A.7, respectively. These correspond to the experiments in Sections 5.7.5 and 5.7.7 and are averaged over 2,500 cases for all tasks and approach variants. See the discussion in Section 5.7.8.

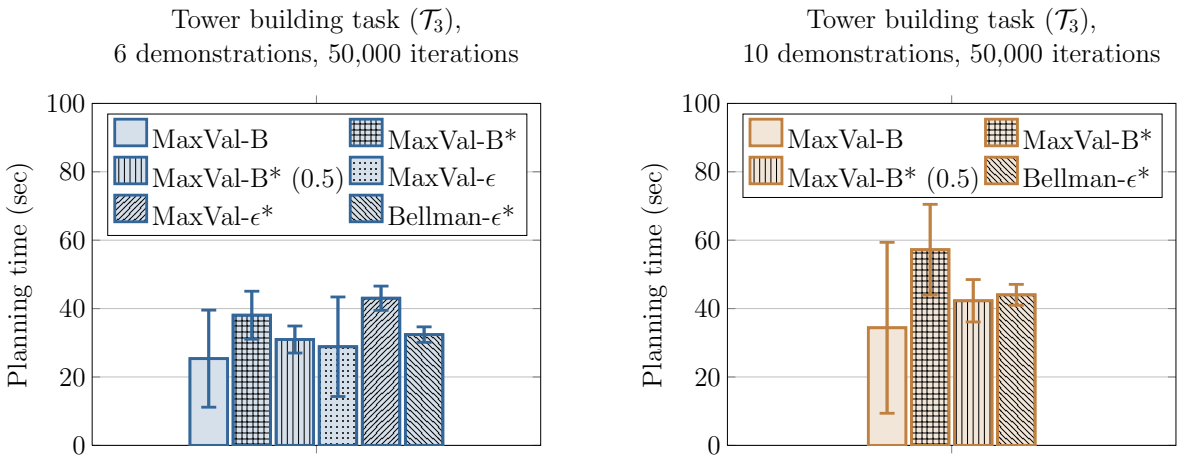


Figure A.7.: The planning times of our approach after 50,000 iterations when solving the tower building task ($\mathcal{T}_3$) given six (left) and ten (right) demonstrations.

A.4. Improvising Task Solutions Based on the Initial State

In this section, we demonstrate the ability of our approach to improvise task solutions based on the initial state without requiring the user to specify a concrete goal to achieve (see Section 5.7.10).

A.4.1. Adapting to Clutter in the Scene

Figure A.8 shows qualitative examples of solving the tidy-up task ($\mathcal{T}_4$) in initial states including clutter caused by objects unrelated to the task. Here, it physically infeasible to achieve all relevant relations demonstrated by the teacher (e.g., satisfying object positions relative to the table). By formulating an optimization problem, our approach improvised feasible configurations that satisfy as many pairwise relations as possible.

A.4.2. Starting from a Partially-Solved Configuration

We tested our approach for solving the aligning blocks task ($\mathcal{T}_2$) in starting states that represent partially-solved arrangements. We generated 100 initial states in which the blocks are aligned as in the demonstrations and then randomly changed the poses of two blocks. We used our approach (Bellman-ϵ*) to solve the task in these cases using models learned from two to ten demonstrations after practicing the task. We repeated this 25 times using different demonstrations in each case. We report the results in Figure A.9. Our approach was able to achieve 97.2% - 97.5% of the relations by applying only 7.4 - 7.9 actions on average. As the initial states already represent

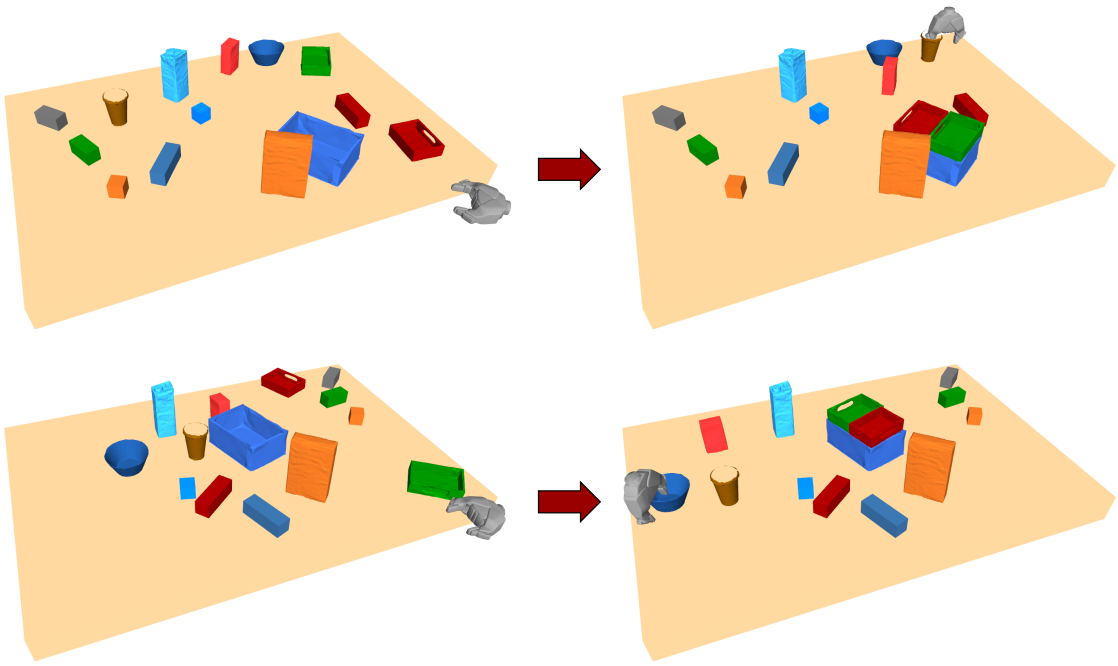


Figure A.8.: Examples of solutions by our approach for the tidy-up task ($\mathcal{T}_4$). Despite clutter caused by task-irrelevant objects, our approach improvised feasible solutions that satisfy as many relations as possible by achieving partial object arrangements on different positions on the table.

good solution initializations, our approach was able to quickly converge to desirable goal configurations, see Figure A.10 for an example. In comparison, when starting in completely random initial states, Bellman-ϵ* required 11.5 - 11.7 actions to achieve 88.1% - 92.3% of the relations on average (see Figure A.3-bottom left and Figure A.4-bottom right).

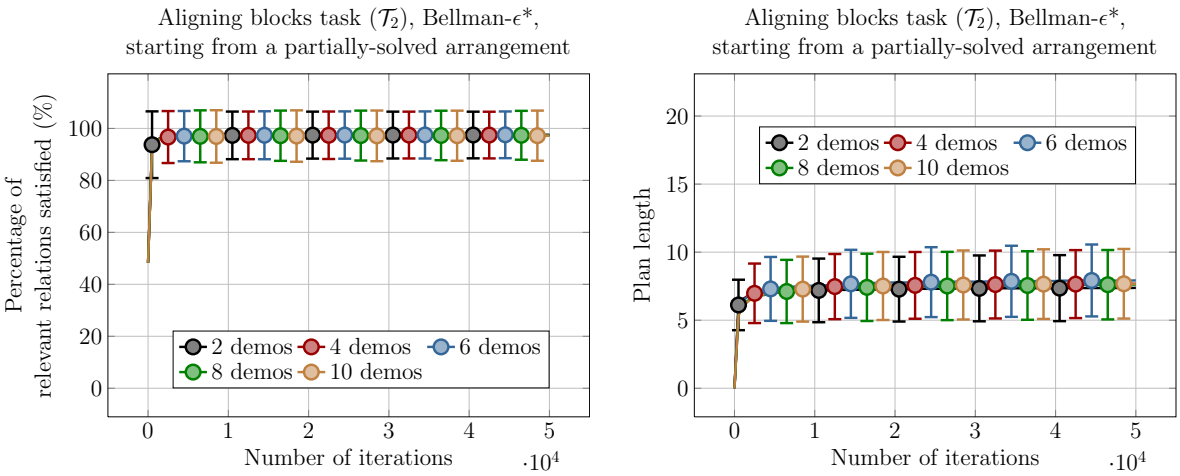


Figure A.9.: Results for solving the aligning blocks task ($\mathcal{T}_2$) in partially-solved states in which four of the six blocks are already aligned in a row on the table. As the initial states represent good solution initializations, our approach was able to quickly converge to desirable goal configurations to achieve 97.2% - 97.5% of the relations using 7.4 - 7.9 actions on average (compare to Bellman-ϵ* in Figures A.3 and A.4 when starting in random states).

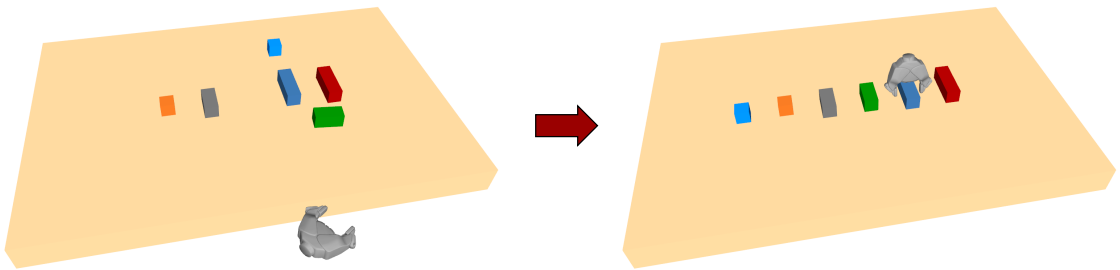


Figure A.10.: Example case of solving the aligning blocks task ($\mathcal{T}_2$) starting with four blocks already aligned on the table, see Figure A.9.

A.5. Real Robot Experiments

We implemented and tested our teach-and-improvise approach on a real PR2 robot (see Section 5.7.11 for details). We relied on out-of-the-box solutions for perception and motion planning to estimate the state and execute the plans computed by our algorithm, respectively. Figures A.11 and A.12 show examples of two tasks we considered. By incorporating feasibility constraints in planning, our algorithm enables the robot to compute sequences of actions leading to desirable object configurations, which the robot can execute in practice.

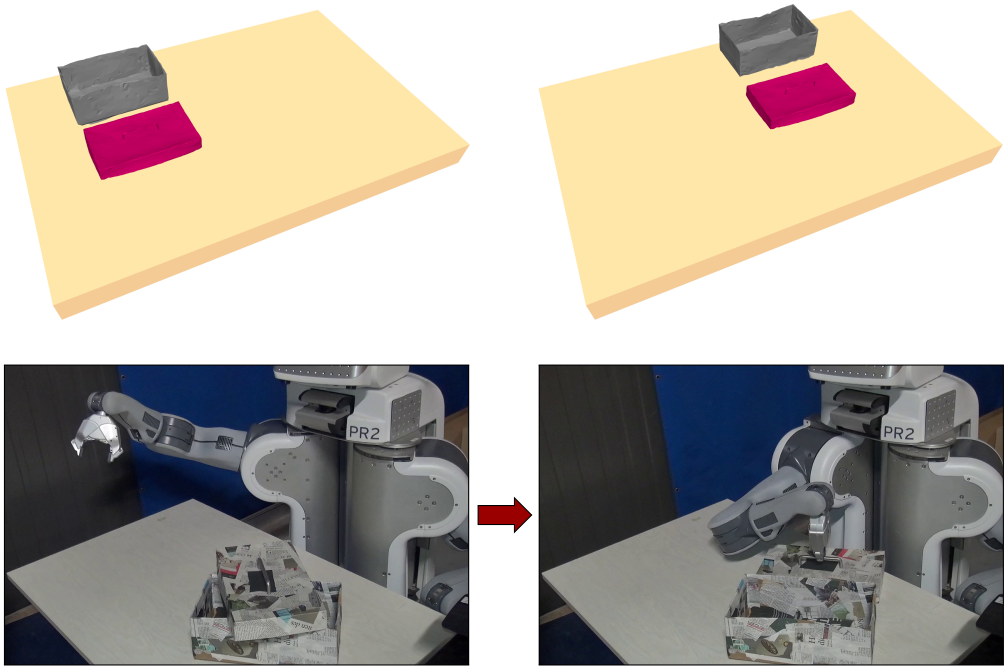


Figure A.11.: Top: examples of demonstrated goals we provided to the robot. Bottom: using our approach, the robot was able to compute and execute a feasible plan to open the box and align the lid next to the box on the table.

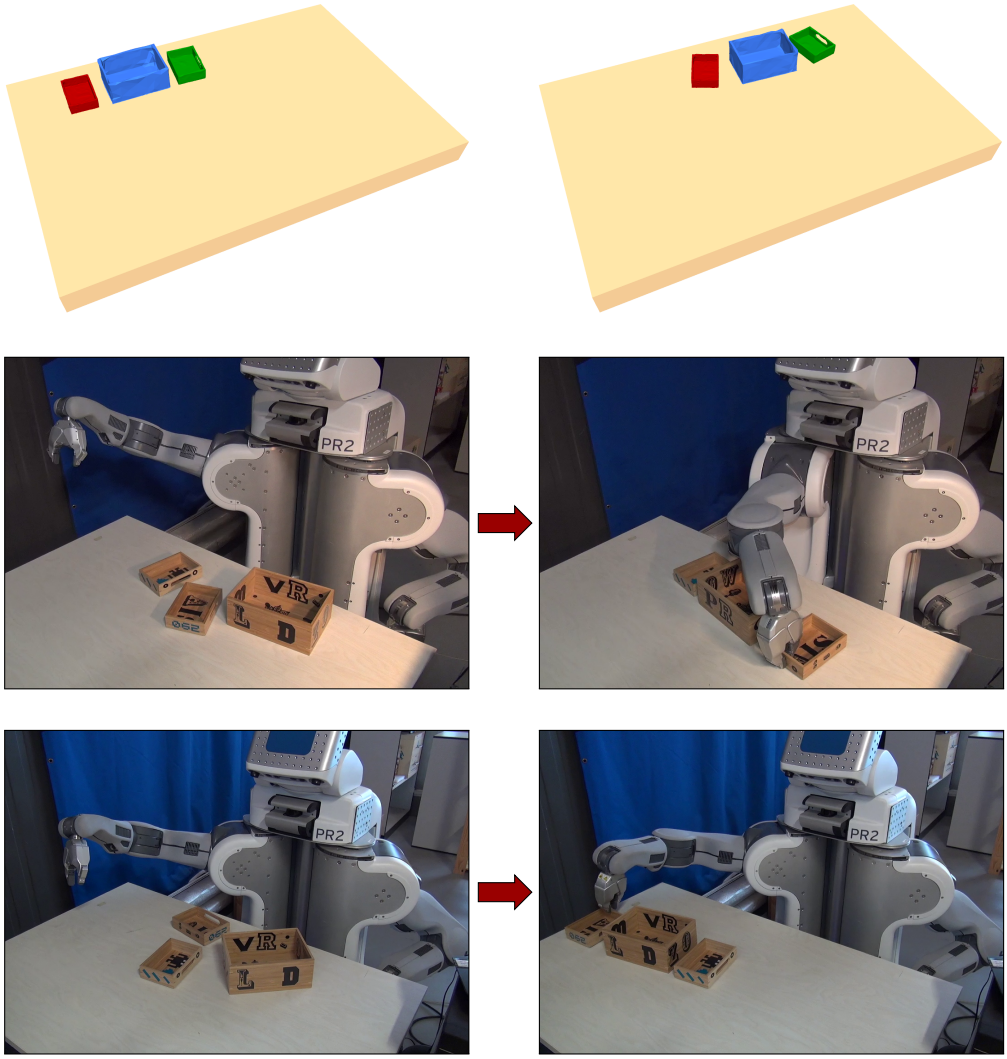


Figure A.12.: First row: examples of demonstrated goals we provided to the robot. Second and third rows: using our approach, the robot was able to compute and execute feasible plans to arrange the boxes on the table.

Bibliography

P. Abbeel and A. Y. Ng. Apprenticeship learning via inverse reinforcement learning. In *Int. Conf. on Machine Learning (ICML)*, 2004.

N. Abdo, H. Kretzschmar, L. Spinello, and C. Stachniss. Learning manipulation actions from a few demonstrations. In *IEEE Int. Conf. on Robotics & Automation (ICRA)*, 2013.

N. Abdo, L. Spinello, W. Burgard, and C. Stachniss. Inferring what to imitate in manipulation actions by using a recommender system. In *IEEE Int. Conf. on Robotics & Automation (ICRA)*, 2014a.

N. Abdo, L. Spinello, C. Stachniss, and W. Burgard. Collaborative filtering for learning user preferences for robotic tasks. In *Workshop on Learning Plans with Context from Human Signals at Robotics: Science and Systems (R:SS)*, 2014b.

N. Abdo, C. Stachniss, L. Spinello, and W. Burgard. Robot, organize my shelves! Tidying up objects by predicting user preferences. In *IEEE Int. Conf. on Robotics & Automation (ICRA)*, 2015.

N. Abdo, C. Stachniss, L. Spinello, and W. Burgard. Organizing objects by predicting user preferences through collaborative filtering. *Int. J. of Robotics Research (IJRR)*, 35(13):1587–1608, 2016.

G. Adomavicius and A. Tuzhilin. Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. *IEEE Transactions on Knowledge and Data Engineering*, 17(6):734–749, 2005.

I. Ahmad and P.-E. Lin. A nonparametric estimation of the entropy for absolutely continuous distributions. *IEEE Transactions on Information Theory*, 22(3):372–375, 1976.

S. R. Ahmadzadeh, A. Paikan, F. Mastrogiovanni, L. Natale, P. Kormushev, and D. G. Caldwell. Learning symbolic representations of actions from human demonstrations. In *IEEE Int. Conf. on Robotics & Automation (ICRA)*, 2015.

B. Akgun and A. Thomaz. Simultaneously learning actions and goals from demonstration. *Autonomous Robots*, 40(2):211–227, 2016.

B. Akgun, M. Cakmak, K. Jiang, and A. L. Thomaz. Keyframe-based learning from demonstration. *Int. J. of Social Robotics (IJSR)*, 4(4):343–355, 2012.

S. Alexandrova, M. Cakmak, K. Hsiao, and L. Takayama. Robot programming by demonstration with interactive action visualizations. In *Robotics: Science and Systems (R:SS)*, 2014.

S. Alexandrova, Z. Tatlock, and M. Cakmak. Roboflow: A flow-based visual programming language for mobile manipulation tasks. In *IEEE Int. Conf. on Robotics & Automation (ICRA)*, 2015.

B. D. Argall, S. Chernova, M. Veloso, and B. Browning. A survey of robot learning from demonstration. *Robotics and autonomous systems*, 57(5):469–483, 2009.

M. Arie-Nachimson and R. Basri. Constructing implicit 3d shape models for pose estimation. In *Int. Conf. on Computer Vision*, 2009.

T. Asfour, F. Gyarfas, P. Azad, and R. Dillmann. Imitation learning of dual-arm manipulation tasks in humanoid robots. In *IEEE-RAS Int. Conf. on Humanoid Robots (Humanoids)*, 2006.

P. Auer, N. Cesa-Bianchi, and P. Fischer. Finite-time analysis of the multiarmed bandit problem. *Machine learning*, 47(2-3):235–256, 2002.

A. Aydemir and P. Jensfelt. Exploiting and modeling local 3d structure for predicting object locations. In *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2012.

A. Bai, S. Srivastava, and S. Russell. Markovian state and action abstractions for mdps via hierarchical mcts. In *Int. Joint Conf. on Artificial Intelligence (IJCAI)*, 2016.

A. Baisero, Y. Mollard, M. Lopes, M. Toussaint, and I. Luetkebohle. Temporal segmentation of pair-wise interaction phases in sequential manipulation demonstrations. In *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2015.

C. Basu, H. Hirsh, and W. Cohen. Recommendation as classification: Using social and content-based information in recommendation. In *National Conf. on Artificial Intelligence (AAAI)*, 1998.

M. Beetz, U. Klank, I. Kresse, A. Maldonado, L. Mösenlechner, D. Pangercic, T. Rühr, and M. Tenorth. Robotic roommates making pancakes. In *IEEE-RAS Int. Conf. on Humanoid Robots (Humanoids)*, 2011.

A. Bellet, A. Habrard, and M. Sebban. A survey on metric learning for feature vectors and structured data. *arXiv preprint arXiv:1306.6709*, 2013.

R. Bellman. *Dynamic Programming*. Princeton University Press, 1957a.

R. Bellman. A markovian decision process. Technical report, DTIC Document, 1957b.

J. Bennett and S. Lanning. The Netflix prize. In *Proceedings of KDD cup and workshop*, 2007.

D. Bentivegna, C. G. Atkeson, and G. Cheng. Learning tasks from observation and practice. *Robotics & Autonomous Systems*, 47(2-3), 2004.

A. Billard, S. Calinon, R. Dillmann, and S. Schaal. Robot programming by demonstration. In B. Siciliano and O. Khatib, editors, *Handbook of Robotics*. Springer, 2008.

S. Brawner and M. L. Littman. Learning user's preferred household organization via collaborative filtering methods. In *Joint Workshop on Interfaces and Human Decision Making for Recommender Systems (IntRS) at the ACM Conf. on Recommender Systems (RecSys)*, 2016.

C. B. Browne, E. Powley, D. Whitehouse, S. M. Lucas, P. I. Cowling, P. Rohlfshagen, S. Tavener, D. Perez, S. Samothrakis, and S. Colton. A survey of monte carlo tree search methods. *IEEE Transactions on Computational Intelligence and AI in Games*, 4(1):1–43, 2012.

M. Cakmak and L. Takayama. Towards a comprehensive chore list for robots. In *Int. Conf. on Human-Robot Interaction (HRI)*, 2013.

S. Calinon, F. Guenter, and A. Billard. On learning, representing, and generalizing a task in a humanoid robot. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 37(2):286–298, 2007.

S. Calinon, F. D'halluin, D. G. Caldwell, and A. G. Billard. Handling of multiple constraints and motion alternatives in a robot programming by demonstration framework. In *IEEE-RAS Int. Conf. on Humanoid Robots (Humanoids)*, 2009.

S. Calinon, F. D'halluin, E. Sauser, D. Caldwell, and A. Billard. A probabilistic approach based on dynamical systems to learn and reproduce gestures by imitation. *IEEE Robotics and Automation Magazine*, 17(2):44–54, 2010.

J. Canny. Collaborative filtering with privacy via factor analysis. In *Annual Int. ACM SIGIR Conf. on Research and Development in Information Retrieval*, 2002.

E. Cha, J. Forlizzi, and S. Srinivasa. Robots in the home: Qualitative and quantitative insights into kitchen organization. In *Int. Conf. on Human-Robot Interaction (HRI)*, 2015.

T. L. Chen, M. Ciocarlie, S. Cousins, P. Grice, K. Hawkins, K. Hsiao, C. C. Kemp, C. H. King, D. A. Lazewatsky, A. Leeper, H. Nguyen, A. Paepcke, C. Pantofaru, W. D. Smart, and L. Takayama. Robots for humanity: User-centered design for assistive mobile manipulation. In *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2012.

S. Chopra, R. Hadsell, and Y. LeCun. Learning a similarity metric discriminatively, with application to face verification. In *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2005.

F. R. K. Chung. Spectral graph theory. In *CBMS Regional Conference Series in Mathematics*, 1996.

M. J.-Y. Chung, M. Forbes, M. Cakmak, and R. P. Rao. Accelerating imitation learning through crowdsourcing. In *IEEE Int. Conf. on Robotics & Automation (ICRA)*, 2014.

R. Coulom. Efficient selectivity and backup operators in monte-carlo tree search. In *Int. Conf. on Computers and Games*, 2006.

R. Cubek, W. Ertel, and G. Palm. High-level learning from demonstration with conceptual spaces and subspace clustering. In *IEEE Int. Conf. on Robotics & Automation (ICRA)*, 2015.

B. Da Silva, G. Konidaris, and A. Barto. Learning parameterized skills. In *Int. Conf. on Machine Learning (ICML)*, 2012.

K. Dautenhahn, S. Woods, C. Kaouri, M. Walters, K. L. Koay, and I. Werry. What is a robot companion - friend, assistant or butler? In *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2005.

R. Dearden and C. Burbridge. Manipulation planning using learned symbolic state abstractions. *Robotics and Autonomous Systems*, 62(3):355–365, 2014.

J. Deng, J. Krause, and L. Fei-Fei. Fine-grained crowdsourcing for fine-grained recognition. In *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2013.

M. Dogar and S. Srinivasa. A framework for push-grasping in clutter. In *Robotics: Science and Systems (R:SS)*, 2011.

M. R. Dogar, M. Cakmak, E. Uğur, et al. From primitive behaviors to goal-directed behavior using affordances. In *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2007.

Domino's Pizza Inc. DRU - Domino's robotic unit, 2017. URL `https://www.dominos.com.au/inside-dominos/technology/dru`. Last accessed: April 2017.

C. Dornhege and A. Hertle. Integrated symbolic planning in the tidyup-robot project. In *AAAI Spring Symposium*, 2013.

C. Dornhege, P. Eyerich, T. Keller, S. Trüg, M. Brenner, and B. Nebel. Semantic attachments for domain-independent planning systems. In *Int. Conf. on on Automated Planning and Scheduling (ICAPS)*, 2009.

C. Dornhege, P. Eyerich, T. Keller, S. Trüg, M. Brenner, and B. Nebel. Semantic attachments for domain-independent planning systems. In *Towards service robots for everyday environments*, pages 99–115. Springer, 2012.

A. Doumanoglou, A. Kargakos, T.-K. Kim, and S. Malassiotis. Autonomous active recognition and unfolding of clothes using random decision forests and probabilistic planning. In *IEEE Int. Conf. on Robotics & Automation (ICRA)*, 2014.

I. Dryanovsk, B. Morris, and G. Dumonteil. ar_pose. http://wiki.ros.org/ar_pose.

Y. Duan, M. Andrychowicz, B. Stadie, J. Ho, J. Schneider, I. Sutskever, P. Abbeel, and W. Zaremba. One-shot imitation learning. *arXiv preprint arXiv:1703.07326*, 2017.

E. B. Dynkin. The optimum choice of the instant for stopping a markov process. *Soviet Mathematics Doklady*, 4(627-629), 1963.

S. Džeroski, L. De Raedt, and K. Driessens. Relational reinforcement learning. *Machine learning*, 43(1-2):7–52, 2001.

A. Eitel, J. T. Springenberg, L. Spinello, M. Riedmiller, and W. Burgard. Multimodal deep learning for robust rgb-d object recognition. In *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2015.

S. Ekvall and D. Kragic. Learning task models from multiple human demonstrations. In *IEEE Int. Symposium on Robot and Human Interactive Communication (RO-MAN)*, 2006.

F. Endres, J. Hess, J. Sturm, D. Cremers, and W. Burgard. 3d mapping with an rgb-d camera. *IEEE Transactions on Robotics*, 30(1):177–187, 2014.

P. Englert and M. Toussaint. Combined optimization and reinforcement learning for manipulations skills. In *Robotics: Science and Systems (R:SS)*, 2016.

P. Englert, A. Paraschos, J. Peters, and M. P. Deisenroth. Model-based imitation learning by probabilistic trajectory matching. In *IEEE Int. Conf. on Robotics & Automation (ICRA)*, 2013.

C. Eppner, J. Sturm, M. Bennewitz, C. Stachniss, and W. Burgard. Imitation learning with generalized task descriptions. In *IEEE Int. Conf. on Robotics & Automation (ICRA)*, 2009.

M. Ersen, M. D. Ozturk, M. Biberci, S. Sariel, and H. Yalcin. Scene interpretation for lifelong robot learning. In *International Cognitive Robotics Workshop (CogRob) at the European Conf. on Artificial Intelligence (ECAI)*, 2014.

M. Ewerton, G. Neumann, R. Lioutikov, H. B. Amor, J. Peters, and G. Maeda. Learning multiple collaborative tasks with a mixture of interaction primitives. In *IEEE Int. Conf. on Robotics & Automation (ICRA)*, 2015.

S. Feng, X. Xinjilefu, C. G. Atkeson, and J. Kim. Optimization based controller design and implementation for the atlas robot in the darpa robotics challenge finals. In *IEEE-RAS Int. Conf. on Humanoid Robots (Humanoids)*, 2015.

Fetch Robotics. Fetch robotics, 2017. URL `http://fetchrobotics.com/`. Last accessed: April 2017.

S. Fichtl, A. McManus, W. Mustafa, D. Kraft, N. Krüger, and F. Guerin. Learning spatial relationships from 3d vision using histograms. In *IEEE Int. Conf. on Robotics & Automation (ICRA)*, 2014.

N. Figueroa, A. L. P. Ureche, and A. Billard. Learning complex sequential tasks from demonstration: A pizza dough rolling case study. In *Int. Conf. on Human-Robot Interaction (HRI)*, 2016.

C. Finn, X. Y. Tan, Y. Duan, T. Darrell, S. Levine, and P. Abbeel. Deep spatial autoencoders for visuomotor learning. In *IEEE Int. Conf. on Robotics & Automation (ICRA)*, 2016.

M. Fisher, D. Ritchie, M. Savva, T. Funkhouser, and P. Hanrahan. Example-based synthesis of 3d object arrangements. *ACM Transactions on Graphics (TOG) - Proceedings of ACM SIGGRAPH Asia 2012*, 31(6):135:1–135:11, 2012.

J. Forlizzi and C. DiSalvo. Service robots in the domestic environment: A study of the roomba vacuum in the home. In *Int. Conf. on Human-Robot Interaction (HRI)*, 2006.

S. Gelly and D. Silver. Combining online and offline knowledge in uct. In *Int. Conf. on Machine Learning (ICML)*, 2007.

S. Gelly and D. Silver. Monte-carlo tree search and rapid action value estimation in computer go. *Artificial Intelligence*, 175(11):1856–1875, 2011.

M. Gharbi, R. Lallement, and R. Alami. Combining symbolic and geometric planning to synthesize human-aware plans: toward more efficient combined search. In *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2015.

J. J. Gibson. *The Ecological Approach to Visual Perception*. Houghton Mifflin, 1979.

M. Gienger, M. Mühlig, and J. J. Steil. Imitating object movement skills with robots – a task-level approach exploiting generalization and invariance. In *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2010.

G. Gierse, T. Niemueller, J. Claßen, and G. Lakemeyer. Interruptible task execution with resumption in golog. In *European Conf. on Artificial Intelligence (ECAI)*, 2016.

D. Goldberg, D. Nichols, B. M. Oki, and D. Terry. Using collaborative filtering to weave an information tapestry. *Comm. of the ACM*, 1992.

O. Goldschmidt and D. S. Hochbaum. A polynomial algorithm for the k-cut problem for fixed k. *Mathematics of Operations Research*, 1994.

K. Grave and S. Behnke. Learning sequential tasks interactively from demonstrations and own experience. In *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2013.

K. Grave and S. Behnke. Bayesian exploration and interactive demonstration in continuous state MAXQ-learning. In *IEEE Int. Conf. on Robotics & Automation (ICRA)*, 2014.

S. Guadarrama, L. Riano, D. Golland, D. Go, Y. Jia, D. Klein, P. Abbeel, T. Darrell, et al. Grounding spatial relations for human-robot interaction. In *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2013.

M. Guillaumin, J. Verbeek, and C. Schmid. Is that you? metric learning approaches for face identification. In *Int. Conf. on Computer Vision*, 2009.

R. Hadsell, S. Chopra, and Y. LeCun. Dimensionality reduction by learning an invariant mapping. In *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2006.

K. He, X. Zhang, S. Ren, and J. Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Int. Conf. on Computer Vision*, 2015.

D. Hennes and D. Izzo. Interplanetary trajectory planning with monte carlo tree search. In *Int. Joint Conf. on Artificial Intelligence (IJCAI)*, 2015.

P. Henry, M. Krainin, E. Herbst, X. Ren, and D. Fox. RGB-D mapping: Using depth cameras for dense 3d modeling of indoor environments. In *Int. Symp. on Experimental Robotics (ISER)*, 2010.

M. Herman, T. Gindele, J. Wagner, F. Schmitt, and W. Burgard. Inverse reinforcement learning with simultaneous estimation of rewards and dynamics. In *Int. Conf. on Artificial Intelligence and Statistics*, 2016.

A. Herzog, P. Pastor, M. Kalakrishnan, L. Righetti, J. Bohg, T. Asfour, and S. Schaal. Learning of grasp selection based on shape-templates. *Autonomous Robots*, 36(1-2): 51–65, 2014.

J. Hess, G. D. Tipaldi, and W. Burgard. Null space optimization for effective coverage of 3D surfaces using redundant manipulators. In *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2012.

S. Höfer and O. Brock. Coupled learning of action parameters and forward models for manipulation. In *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2016.

S. Höfer, T. Lang, and O. Brock. Extracting kinematic background knowledge from interactions using task-sensitive relational learning. In *IEEE Int. Conf. on Robotics & Automation (ICRA)*, 2014.

E. Hoffer and N. Ailon. Deep metric learning using triplet network. In *Int. Workshop on Similarity-Based Pattern Recognition*, 2015.

T. Hofmann, T. Niemueller, J. Claßen, and G. Lakemeyer. Continual planning in golog. In *National Conf. on Artificial Intelligence (AAAI)*, 2016.

S. C. Hoi, W. Liu, and S.-F. Chang. Semi-supervised distance metric learning for collaborative image retrieval. In *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2008.

A. Hornung, K. M. Wurm, M. Bennewitz, C. Stachniss, and W. Burgard. OctoMap: An efficient probabilistic 3D mapping framework based on octrees. *Autonomous Robots*, 34(3):189–206, 2013.

A. J. Ijspeert, J. Nakanishi, and S. Schaal. Movement imitation with nonlinear dynamical systems in humanoid robots. In *IEEE Int. Conf. on Robotics & Automation (ICRA)*, 2002.

iRobot Corp. iRobot: Our history, 2017. URL `http://www.irobot.com/us/Company/About/Our_History.aspx`. Last accessed: April 2017.

A. Jain, B. Wojcik, T. Joachims, and A. Saxena. Learning trajectory preferences for manipulators via iterative improvement. In *Advances in Neural Information Processing Systems (NIPS)*, 2013.

A. Jain, D. Das, J. Gupta, and A. Saxena. Planit: A crowdsourcing approach for learning to plan paths from large scale preference feedback. In *IEEE Int. Conf. on Robotics & Automation (ICRA)*, 2015.

R. Jäkel, P. Meissner, S. R. Schmidt-Rohr, and R. Dillmann. Distributed generalization of learned planning models in robot programming by demonstration. In *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2011.

N. Jetchev and M. Toussaint. Task space retrieval using inverse feedback control. In *Int. Conf. on Machine Learning (ICML)*, 2011.

N. Jetchev, T. Lang, and M. Toussaint. Learning grounded relational symbols from continuous data for abstract reasoning. In *Workshop on Autonomous Learning at the Int. Conf. on Robotics & Automation (ICRA)*, 2013.

Y. Jiang, M. Lim, and A. Saxena. Learning object arrangements in 3d scenes using human context. In *Int. Conf. on Machine Learning (ICML)*, 2012a.

Y. Jiang, M. Lim, C. Zheng, and A. Saxena. Learning to place new objects in a scene. *Int. J. of Robotics Research (IJRR)*, 2012b.

Jibo. Jibo, 2017. URL `https://www.jibo.com/`. Last accessed: April 2017.

D. Joho, M. Senk, and W. Burgard. Learning search heuristics for finding objects in structured environments. In *Robotics & Autonomous Systems*, volume 59, pages 319–328, 2011.

D. Joho, G. D. Tipaldi, N. Engelhard, C. Stachniss, and W. Burgard. Nonparametric Bayesian models for unsupervised scene analysis and reconstruction. In *Robotics: Science and Systems (R:SS)*, 2012.

L. P. Kaelbling and T. Lozano-Pérez. Hierarchical task and motion planning in the now. In *IEEE Int. Conf. on Robotics & Automation (ICRA)*, 2011.

L. P. Kaelbling and T. Lozano-Pérez. Integrated task and motion planning in belief space. *Int. J. of Robotics Research (IJRR)*, 32(9-10):1194–1227, 2013.

P. Kaiser, M. Lewis, R. Petrick, T. Asfour, and M. Steedman. Extracting common sense knowledge from text for robot planning. In *IEEE Int. Conf. on Robotics & Automation (ICRA)*, 2014.

M. Kalakrishnan, L. Righetti, P. Pastor, and S. Schaal. Learning force control policies for compliant manipulation. In *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2011.

M. Kalakrishnan, L. Righetti, P. Pastor, and S. Schaal. Learning force control policies for compliant robotic manipulation. In *Int. Conf. on Machine Learning (ICML)*, 2012.

B. Kartal, E. Nunes, J. Godoy, and M. L. Gini. Monte carlo tree search for multi-robot task allocation. In *National Conf. on Artificial Intelligence (AAAI)*, 2016.

L. E. Kavraki and S. M. LaValle. Motion planning. In *Springer handbook of robotics*, pages 139–162. Springer International Publishing, 2016.

D. Kedem, S. Tyree, F. Sha, G. R. Lanckriet, and K. Q. Weinberger. Non-linear metric learning. In *Advances in Neural Information Processing Systems (NIPS)*, 2012.

B. Kehoe, A. Matsukawa, S. Candido, J. Kuffner, and K. Goldberg. Cloud-based robot grasping with the google object recognition engine. In *IEEE Int. Conf. on Robotics & Automation (ICRA)*, 2013.

T. Keller and P. Eyerich. Prost: Probabilistic planning based on uct. In *Int. Conf. on on Automated Planning and Scheduling (ICAPS)*, 2012.

T. Keller and F. Geißer. Better be lucky than good: Exceeding expectations in MDP evaluation. In *National Conf. on Artificial Intelligence (AAAI)*, 2015.

T. Keller and M. Helmert. Trial-based heuristic tree search for finite horizon mdps. In *Int. Conf. on on Automated Planning and Scheduling (ICAPS)*, 2013.

D. Kent, M. Behrooz, and S. Chernova. Crowdsourcing the construction of a 3d object recognition database for robotic grasping. In *IEEE Int. Conf. on Robotics & Automation (ICRA)*, 2014.

K. Kersting, M. V. Otterlo, and L. De Raedt. Bellman goes relational. In *Int. Conf. on Machine Learning (ICML)*, 2004.

P. Khandelwal, E. Liebman, S. Niekum, and P. Stone. On the analysis of complex backup strategies in monte carlo tree search. In *Int. Conf. on Machine Learning (ICML)*, 2016.

S. M. Khansari-Zadeh and A. Billard. Learning stable nonlinear dynamical systems with gaussian mixture models. In *IEEE Transactions on Robotics*, volume 27, pages 943–957. IEEE, 2011.

J. Knopp, M. Prasad, G. Willems, R. Timofte, and L. V. Gool. Hough transform and 3d surf for robust three dimensional classification. In *European Conf. on Computer Vision (ECCV)*, 2010.

J. Kober and J. Peters. Policy search for motor primitives in robotics. In *Advances in Neural Information Processing Systems (NIPS)*, 2009.

J. Kober and J. Peters. Imitation and reinforcement learning - practical algorithms for motor primitive learning in robotics. (2):55–62, 2010.

L. Kocsis and C. Szepesvári. Bandit based monte-carlo planning. In *European Conf. on Machine Learning (ECML)*, 2006.

G. Konidaris, S. Kuindersma, R. Grupen, and A. Barto. Robot learning from demonstration by constructing skill trees. *Int. J. of Robotics Research (IJRR)*, 31(3):360–375, 2012.

G. Konidaris, L. P. Kaelbling, and T. Lozano-Perez. Constructing symbolic representations for high-level planning. In *National Conf. on Artificial Intelligence (AAAI)*, 2014.

G. Konidaris, L. P. Kaelbling, and T. Lozano-Perez. Symbol acquisition for probabilistic high-level planning. In *Int. Joint Conf. on Artificial Intelligence (IJCAI)*, 2015.

G. D. Konidaris and A. G. Barto. Efficient skill learning using abstraction selection. In *Int. Joint Conf. on Artificial Intelligence (IJCAI)*, 2009.

Y. Koren. Factorization meets the neighborhood: a multifaceted collaborative filtering model. In *Int. Conf. on Knowledge Disc. and Data Mining (SIGKDD)*, 2008.

Y. Koren. Factor in the neighbors: Scalable and accurate collaborative filtering. In *ACM Trans. on Knowledge Disc. from Data (TKDD)*, 2010.

O. Kroemer and J. Peters. Predicting object interactions from contact distributions. In *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2014.

O. Kroemer, C. Daniel, G. Neumann, H. Van Hoof, and J. Peters. Towards learning hierarchical skills for multi-phase manipulation tasks. In *IEEE Int. Conf. on Robotics & Automation (ICRA)*, 2015.

N. Krüger, C. Geib, J. Piater, R. Petrick, M. Steedman, F. Wörgötter, A. Ude, T. Asfour, D. Kraft, D. Omrčen, et al. Object–action complexes: Grounded abstractions of sensory–motor processes. *Robotics and Autonomous Systems*, 59(10):740–757, 2011.

M. Kuderer, H. Kretzschmar, C. Sprunk, and W. Burgard. Feature-based prediction of trajectories for socially compliant navigation. In *Robotics: Science and Systems (R:SS)*, 2012.

D. Kulic, W. Takano, and Y. Nakamura. Online segmentation and clustering from continuous observation of whole body motions. *IEEE Transactions on Robotics*, 25 (5):1158–1166, 2009.

D. Kulic, C. Ott, D. Lee, J. Ishikawa, and Y. Nakamura. Incremental learning of full body motion primitives and their sequencing through human motion observation. *Int. J. of Robotics Research (IJRR)*, 31(3), 2012.

J. Kulick, M. Toussaint, T. Lang, and M. Lopes. Active learning for teaching a robot grounded relational symbols. In *Int. Joint Conf. on Artificial Intelligence (IJCAI)*, 2013.

B. Kulis et al. Metric learning: A survey. *Foundations and Trends in Machine Learning*, 5(4):287–364, 2013.

L. Kunze and M. Beetz. Envisioning the qualitative effects of robot manipulation actions using simulation-based projections. In *Artificial Intelligence*. Elsevier, 2015.

L. Kunze, C. Burbridge, and N. Hawes. Bootstrapping probabilistic models of qualitative spatial relations for active visual object search. In *AAAI Spring Symposium Series*, 2014.

F. Lagriffoul and B. Andres. Combining task and motion planning: A culprit detection problem. *Int. J. of Robotics Research (IJRR)*, 35(8):890–927, 2016.

K. Lai, L. Bo, X. Ren, and D. Fox. A large-scale hierarchical multi-view rgb-d object dataset. In *IEEE Int. Conf. on Robotics & Automation (ICRA)*, 2011a.

K. Lai, L. Bo, X. Ren, and D. Fox. Sparse distance learning for object recognition combining rgb and depth information. In *IEEE Int. Conf. on Robotics & Automation (ICRA)*, 2011b.

M. Lanctot, M. H. Winands, T. Pepels, and N. R. Sturtevant. Monte carlo tree search with heuristic evaluations using implicit minimax backups. In *IEEE Conf. on Computational Intelligence and Games*, 2014.

T. Lang and M. Toussaint. Planning with noisy probabilistic relational rules. *Journal of Artificial Intelligence Research*, 39(1):1–49, 2010.

S. Lazebnik, C. Schmid, and J. Ponce. A sparse texture representation using local affine regions. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(8): 1265–1278, 2005.

K. H. Lee, J. Lee, A. L. Thomaz, and A. F. Bobick. Effective robot task learning by focusing on task-relevant objects. In *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2009.

B. Leibe and B. Schiele. Interleaved object categorization and segmentation. In *British Machine Vision Conf. (BMVC)*, 2003.

B. Leibe, A. Leonardis, and B. Schiele. Combined object categorization and segmentation with an implicit shape model. In *ECCV Workshop on statistical learning in computer vision*, 2004.

S. Lemaignan and R. Alami. Explicit knowledge and the deliberative layer: Lessons learned. In *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2013.

I. Lenz, H. Lee, and A. Saxena. Deep learning for detecting robotic grasps. *Int. J. of Robotics Research (IJRR)*, 34(4-5):705–724, 2015.

S. Levine, P. Pastor, A. Krizhevsky, and D. Quillen. Learning hand-eye coordination for robotic grasping with deep learning and large-scale data collection. *arXiv preprint arXiv:1603.02199*, 2016.

M. Lorbach, S. Höfer, and O. Brock. Prior-assisted propagation of spatial information for object search. In *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2014.

L. v. d. Maaten and G. Hinton. Visualizing data using t-sne. *Journal of Machine Learning Research*, 9(Nov):2579–2605, 2008.

J. MacQueen. Some methods for classification and analysis of multivariate observations. In *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, volume 1, pages 281–297, 1967.

S. Manschitz, J. Kober, M. Gienger, and J. Peters. Learning to sequence movement primitives from demonstrations. In *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2014.

S. Manschitz, J. Kober, M. Gienger, and J. Peters. Probabilistic progress prediction and sequencing of concurrent movement primitives. In *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2015.

T. Mar, V. Tikhanoff, G. Metta, and L. Natale. Self-supervised learning of grasp dependent tool affordances on the icub humanoid robot. In *IEEE Int. Conf. on Robotics & Automation (ICRA)*, 2015.

Marble. Marble - Urban last-mile robots, 2017. URL `https://angel.co/marble-1`. Last accessed: April 2017.

P. Matikainen, R. Sukthankar, and M. Hebert. Model recommendation for action recognition. In *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2012.

P. Matikainen, P. M. Furlong, R. Sukthankar, and M. Hebert. Multi-armed recommendation bandits for selecting state machine policies for robotic systems. In *IEEE Int. Conf. on Robotics & Automation (ICRA)*, 2013.

O. Mees, N. Abdo, M. Mazuran, and W. Burgard. Metric learning for generalizing spatial relations to new objects. In *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2017. Accepted for publication.

F. Meier, E. Theodorou, and S. Schaal. Movement segmentation and recognition for imitation learning. In *Int. Conf. on Artificial Intelligence and Statistics (AISTATS)*, pages 761–769, 2012.

P. Meißner, R. Reckling, V. Wittenbeck, S. R. Schmidt-Rohr, and R. Dillmann. Active scene recognition for programming by demonstration using next-best-view estimates from hierarchical implicit shape models. In *IEEE Int. Conf. on Robotics & Automation (ICRA)*, 2014.

P. Meißner, R. Schleicher, R. Hutmacher, S. R. Schmidt-Rohr, and R. Dillmann. Scene recognition for mobile robots by relational object search using next-best-view estimates from hierarchical implicit shape models. In *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2016.

S. Miller, J. Van Den Berg, M. Fritz, T. Darrell, K. Goldberg, and P. Abbeel. A geometric approach to robotic laundry folding. *Int. J. of Robotics Research (IJRR)*, 2012.

B. Moldovan, P. Moreno, M. van Otterlo, J. Santos-Victor, and L. De Raedt. Learning relational affordance models for robots in multi-object manipulation tasks. In *IEEE Int. Conf. on Robotics & Automation (ICRA)*, 2012.

Y. Mollard, T. Munzer, A. Baisero, M. Toussaint, and M. Lopes. Robot programming from demonstration, feedback and transfer. In *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2015.

L. Montesano, M. Lopes, A. Bernardino, and J. Santos-Victor. Learning object affordances: From sensory–motor coordination to imitation. *Transactions on Robotics*, 24 (1):15–26, 2008.

M. Mühlig, M. Gienger, S. Hellbach, J. Steil, and C. Goerik. Task-level imitation learning using variance-based movement optimization. In *IEEE Int. Conf. on Robotics & Automation (ICRA)*, 2009a.

M. Mühlig, M. Gienger, J. Steil, and C. Goerik. Automatic selection of task spaces for imitation learning. In *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2009b.

G. Neumann, W. Maass, and J. Peters. Learning complex motions by sequencing simpler motion templates. In *Int. Conf. on Machine Learning (ICML)*, 2009.

A. Y. Ng, S. J. Russell, et al. Algorithms for inverse reinforcement learning. In *Int. Conf. on Machine Learning (ICML)*, 2000.

S. Niekum, S. Osentoski, G. Konidaris, and A. Barto. Learning and generalization of complex tasks from unstructured demonstrations. In *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2012.

S. Niekum, S. Osentoski, G. Konidaris, S. Chitta, B. Marthi, and A. G. Barto. Learning grounded finite-state representations from unstructured demonstrations. In *Int. J. of Robotics Research (IJRR)*, volume 34, pages 131–157, 2015.

J. Nocedal. Updating quasi-newton matrices with limited storage. *Mathematics of computation*, 1980.

A. Nüchter and J. Hertzberg. Towards semantic maps for mobile robots. *Robotics and Autonomous Systems*, 56(11):915–926, 2008.

D. Nyga, F. Balint-Benczedi, and M. Beetz. PR2 looking at things: Ensemble learning for unstructured information processing with markov logic networks. In *IEEE Int. Conf. on Robotics & Automation (ICRA)*, 2014.

K. Ogawara, J. Takamatsu, H. Kimura, and K. Ikeuchi. Extraction of essential interactions through multiple observations of human demonstrations. *IEEE Transactions on Industrial Electronics*, 50(4):667–675, 2003.

S. Oßwald, A. Görög, A. Hornung, and M. Bennewitz. Autonomous climbing of spiral staircases with humanoids. In *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2011.

D. Pangercic, V. Haltakov, and M. Beetz. Fast and robust object detection in household environments using vocabulary trees with sift descriptors. In *Workshop on Active Semantic Perception and Object Search in the Real World at the Int. Conf. on Intelligent Robots and Systems (IROS)*, 2011.

D. Pangercic, M. Tenorth, B. Pitzer, and M. Beetz. Semantic object maps for robotic housework - representation, acquisition and use. In *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2012.

C. Pantofaru, L. Takayama, T. Foote, and B. Soto. Exploring the role of robots in home organization. In *Int. Conf. on Human-Robot Interaction (HRI)*, 2012.

M. Pardowitz, S. Knoop, R. Dillmann, and R. D. Zollner. Incremental learning of tasks from user demonstrations, past experiences, and vocal comments. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 37(2):322–332, 2007.

P. Pastor, H. Hoffmann, T. Asfour, and S. Schaal. Learning and generalization of motor skills by learning from demonstration. In *IEEE Int. Conf. on Robotics & Automation (ICRA)*, 2009.

H. M. Pasula, L. S. Zettlemoyer, and L. P. Kaelbling. Learning symbolic models of stochastic domains. *Journal of Artificial Intelligence Research*, 29:309–352, 2007.

K. Pauwels and D. Kragic. Simtrack: A simulation-based framework for scalable real-time object pose detection and tracking. In *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2015.

C. Paxton, F. Jonathan, M. Kobilarov, and G. D. Hager. Do what i want, not what i did: Imitation of skills by planning sequences of actions. In *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2016.

M. J. Pazzani and D. Billsus. Learning and revising user profiles: The identification of interesting web sites. *Machine Learning*, 1997.

D. Pellier, B. Bouzy, and M. Métivier. An uct approach for anytime agent-based planning. In *Advances in Practical Applications of Agents and Multiagent Systems*, pages 211–220. Springer, 2010.

R. P. Petrick and F. Bacchus. A knowledge-based approach to planning with incomplete information and sensing. In *Int. Conf. on Artificial Intelligence Planning Systems (AIPS)*, 2002.

L. Pinto and A. Gupta. Supersizing self-supervision: Learning to grasp from 50k tries and 700 robot hours. In *IEEE Int. Conf. on Robotics & Automation (ICRA)*, 2016.

E. Plaku and G. D. Hager. Sampling-based motion and symbolic action planning with geometric and differential constraints. In *IEEE Int. Conf. on Robotics & Automation (ICRA)*, 2010.

D. Prattichizzo and J. C. Trinkle. Grasping. In *Springer handbook of robotics*, pages 955–988. Springer International Publishing, 2016.

R. Rahmatizadeh, P. Abolghasemi, and L. Bölöni. Learning manipulation trajectories using recurrent neural networks. *arXiv preprint arXiv:1603.03833*, 2016.

N. Ratliff, M. Zucker, J. A. Bagnell, and S. Srinivasa. CHOMP: gradient optimization techniques for efficient motion planning. In *IEEE Int. Conf. on Robotics & Automation (ICRA)*, 2009.

E. Ratner, B. Cohen, M. Phillips, and M. Likhachev. A web-based infrastructure for recording user demonstrations of mobile manipulation tasks. In *IEEE Int. Conf. on Robotics & Automation (ICRA)*, 2015.

C. Ray, F. Mondada, and R. Siegwart. What do people expect from robots? In *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2008.

F. Riccio, R. Capobianco, and D. Nardi. Using monte carlo search with data aggregation to improve robot soccer policies. In *Proceedings of the 20th International RoboCup Symposium*, 2016.

S. Rockel, B. Neumann, J. Zhang, K. S. R. Dubba, A. G. Cohn, Š. Konečnỳ, M. Mansouri, F. Pecora, A. Saffiotti, M. Günther, et al. An ontology-based multi-level robot architecture for learning from experiences. In *Designing Intelligent robots: Reintegrating AI II. AAAI Spring Symposium-Technical Report*, pages 52–57, 2013.

B. Rosman and S. Ramamoorthy. Learning spatial relationships between objects. *Int. J. of Robotics Research (IJRR)*, 30(11):1328–1342, 2011.

S. T. Roweis and L. K. Saul. Nonlinear dimensionality reduction by locally linear embedding. *Science*, 290(5500):2323–2326, 2000.

M. Rudolph, M. Mühlig, M. Gienger, and H.-J. Böhme. Learning the consequences of actions: Representing effects as feature changes. In *Int. Conf. on Emerging Security Technologies (EST)*, 2010.

R. B. Rusu. Semantic 3d object maps for everyday manipulation in human living environments. *KI-Künstliche Intelligenz*, 24(4):345–348, 2010.

B. Sarwar, G. Karypis, J. Konstan, and J. Riedl. Item-based collaborative filtering recommendation algorithms. In *Int. Conf. on World Wide Web*, 2001.

T. R. Savarimuthu, A. G. Buch, C. Schlette, N. Wantia, J. Rossmann, D. Martinez, G. Alenya, C. Torras, A. Ude, B. Nemec, A. Kramberger, F. Wörgötter, E. E. Aksoy, J. Papon, S. Haller, J. Piater, and N. Krüger. Teaching a robot the semantics of assembly tasks. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 2017.

M. Saveriano, S.-i. An, and D. Lee. Incremental kinesthetic teaching of end-effector and null-space motion primitives. In *IEEE Int. Conf. on Robotics & Automation (ICRA)*, 2015.

Savioke. Savioke, 2017. URL `http://www.savioke.com/`. Last accessed: April 2017.

M. Savva, A. X. Chang, P. Hanrahan, M. Fisher, and M. Nießner. Scenegrok: Inferring action maps in 3d environments. *ACM Transactions on Graphics (TOG)*, 2014.

A. Saxena, J. Driemeyer, and A. Y. Ng. Robotic grasping of novel objects using vision. *Int. J. of Robotics Research (IJRR)*, 2008.

J. Schaeffer. The history heuristic and alpha-beta search enhancements in practice. *IEEE transactions on pattern analysis and machine intelligence*, 11(11):1203–1212, 1989.

S. Schröer, I. Killmann, B. Frank, M. Völker, L. D. J. Fiederer, T. Ball, and W. Burgard. An autonomous robotic assistant for drinking. In *IEEE Int. Conf. on Robotics & Automation (ICRA)*, 2015.

T. Schulte and T. Keller. Balancing exploration and exploitation in classical planning. In *Seventh Annual Symposium on Combinatorial Search*, 2014.

M. Schuster, J. Okerman, H. Nguyen, J. Rehg, and C. Kemp. Perceiving clutter and surfaces for object placement in indoor environments. In *IEEE-RAS Int. Conf. on Humanoid Robots (Humanoids)*, 2010.

M. Schuster, D. Jain, M. Tenorth, and M. Beetz. Learning organizational principles in human environments. In *IEEE Int. Conf. on Robotics & Automation (ICRA)*, 2012.

C. Schwering, T. Niemueller, G. Lakemeyer, N. Abdo, and W. Burgard. Sensor fusion in the epistemic situation calculus. *Journal of Experimental & Theoretical Artificial Intelligence*, 28(5):871–887, 2016.

M. Shahid, T. Naseer, and W. Burgard. Dtlc: Deeply trained loop closure detections for lifelong visual slam. In *Visual Place Recognition: What is it Good For? Workshop at Robotics: Science and Systems (R:SS)*, 2016.

D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. Van Den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *Nature*, 529(7587):484–489, 2016.

S. Singh, T. Jaakkola, M. L. Littman, and C. Szepesvári. Convergence results for single-step on-policy reinforcement-learning algorithms. *Machine learning*, 38(3): 287–308, 2000.

C.-A. Smarr, T. Mitzner, J. Beer, A. Prakash, T. Chen, C. Kemp, and W. Rogers. Domestic robots for older adults: Attitudes, preferences, and potential. In *Int. J. of Social Robotics (IJSR)*. Springer Netherlands, 2014.

D. Song, K. Huebner, V. Kyrki, and D. Kragic. Learning task constraints for robot grasping using graphical models. In *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2010.

A. Sorokin, D. Berenson, S. Srinivasa, and M. Hebert. People helping robots helping people: Crowdsourcing for grasping novel objects. In *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2010.

S. Srivastava, E. Fang, L. Riano, R. Chitnis, S. Russell, and P. Abbeel. Combined task and motion planning through an extensible planner-independent interface layer. In *IEEE Int. Conf. on Robotics & Automation (ICRA)*, 2014.

F. Stulp, E. Theodorou, M. Kalakrishnan, P. Pastor, L. Righetti, and S. Schaal. Learning motion primitive goals for robust manipulation. In *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2011.

I. A. Sucan and S. Chitta. Moveit!, 2013. URL `http://moveit.ros.org`.

I. A. Sucan, M. Moll, and L. E. Kavraki. The open motion planning library. *IEEE Robotics & Automation Magazine*, 19(4):72–82, 2012.

J. Sung, S. H. Jin, and A. Saxena. Robobarista: Object part-based transfer of manipulation trajectories from crowd-sourcing in 3d pointclouds. In *Int. Symp. of Robotics Research (ISRR)*, 2015.

R. S. Sutton and A. G. Barto. *Reinforcement learning: An introduction*, volume 1. MIT press Cambridge, 1998.

M. Temerinac-Ott, A. Naik, and R. Murphy. Deciding when to stop: Efficient experimentation to learn to predict drug-target interactions (extended abstract). In *Research in Computational Molecular Biology*, Lecture Notes in Computer Science. Springer International Publishing, 2015.

M. Tenorth and M. Beetz. Knowrob: A knowledge processing infrastructure for cognition-enabled robots. *Int. J. of Robotics Research (IJRR)*, 32(5):566–590, 2013.

M. Tenorth, U. Klank, D. Pangercic, and M. Beetz. Web-enabled Robots – Robots that Use the Web as an Information Resource. 18(2):58–68, 2011.

S. Thrun, W. Burgard, and D. Fox. *Probabilistic robotics*. MIT press, 2005.

R. Toris, D. Kent, and S. Chernova. The robot management system: A framework for conducting human-robot interaction studies through crowdsourcing. *Journal of Human-Robot Interaction*, 2014.

R. Toris, D. Kent, and S. Chernova. Unsupervised learning of multi-hypothesized pick-and-place task templates via crowdsourcing. In *IEEE Int. Conf. on Robotics & Automation (ICRA)*, 2015.

M. Toussaint. Logic-geometric programming: An optimization-based approach to combined task and motion planning. In *Int. Joint Conf. on Artificial Intelligence (IJCAI)*, 2015.

M. Toussaint, T. Munzer, Y. Mollard, L. Y. Wu, N. A. Vien, and M. Lopes. Relational activity processes for modeling concurrent cooperation. In *IEEE Int. Conf. on Robotics & Automation (ICRA)*, 2016.

E. Ugur, S. Szedmak, and J. Piater. Bootstrapping paired-object affordance learning with learned single-affordance features. In *Int. Conf. on Development and Learning and Epigenetic Robotics (ICDL-Epirob)*, 2014.

A. L. P. Ureche, K. Umezawa, Y. Nakamura, and A. Billard. Task parameterization using continuous constraints extracted from human demonstrations. *IEEE Transactions on Robotics*, 31(6):1458–1471, 2015.

S. Vasudevan, S. Gächter, V. Nguyen, and R. Siegwart. Cognitive maps for mobile robots-an object based approach. *Robotics & Autonomous Systems*, 2007.

H. Veeraraghavan and M. Veloso. Teaching sequential tasks with repetition through demonstration. In *Int. Conf. on Autonomous Agents and Multiagent Systems*, 2008.

N. A. Vien and M. Toussaint. Hierarchical monte-carlo planning. In *National Conf. on Artificial Intelligence (AAAI)*, 2015.

U. von Luxburg. A tutorial on spectral clustering. *Statistics and Computing*, 2007.

M. Wachter and T. Asfour. Hierarchical segmentation of manipulation actions based on object relations and motion characteristics. In *Int. Conf. on Advanced Robotics*, 2015.

M. Wachter, S. Schulz, T. Asfour, E. Aksoy, F. Worgotter, and R. Dillmann. Action sequence reproduction based on automatic segmentation and object-action complexes. In *IEEE-RAS Int. Conf. on Humanoid Robots (Humanoids)*, 2013.

K. Q. Weinberger and L. K. Saul. Distance metric learning for large margin nearest neighbor classification. *Journal of Machine Learning Research*, 10(Feb):207–244, 2009.

E. Weitnauer, R. Haschke, and H. Ritter. Evaluating a physics engine as an ingredient for physical reasoning. In *Int. Conf. on Simulation, Modeling, and Programming for Autonomous Robots*, 2010.

K. Welke, P. Kaiser, A. Kozlov, N. Adermann, T. Asfour, M. Lewis, and M. Steedman. Grounded spatial symbols for task planning based on experience. In *IEEE-RAS Int. Conf. on Humanoid Robots (Humanoids)*, 2013.

T. Welschehold, C. Dornhege, and W. Burgard. Learning manipulation actions from human demonstrations. In *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2016.

M. H. Winands and Y. Björnsson. Evaluation function based monte-carlo loa. In *Advances in Computer Games*, pages 33–44. Springer, 2009.

Z. Wu and M. Palmer. Verbs semantics and lexical selection. In *Proceedings of the 32nd Annual Meeting on Association for Computational Linguistics*, 1994.

S. Xiang, F. Nie, and C. Zhang. Learning a mahalanobis distance metric for data clustering and classification. *Pattern Recognition*, 41(12):3600–3612, 2008.

K. Xu, J. Stewart, and E. Fiume. Constraint-based automatic placement for scene composition. In *Graphics Interface*, 2002.

D. Yi, Z. Lei, S. Liao, and S. Z. Li. Deep metric learning for person re-identification. In *Int. Conf. on Pattern Recognition (ICPR)*, 2014.

K. Zampogiannis, Y. Yang, C. Fermuller, and Y. Aloimonos. Learning the spatial semantics of manipulation actions through preposition grounding. In *IEEE Int. Conf. on Robotics & Automation (ICRA)*, 2015.

L. Zelnik-Manor and P. Perona. Self-tuning spectral clustering. In *Advances in Neural Information Processing Systems (NIPS)*, 2004.

H. Zender, O. Martínez Mozos, P. Jensfelt, G.-J. Kruijff, and W. Burgard. Conceptual spatial representations for indoor mobile robots. 56(6):493–502, 2008.

M. M. Zhang, N. Atanasov, and K. Daniilidis. Active tactile object recognition by monte carlo tree search. *arXiv preprint arXiv:1703.00095*, 2017.

B. Zhou, A. Lapedriza, J. Xiao, A. Torralba, and A. Oliva. Learning deep features for scene recognition using places database. In *Advances in Neural Information Processing Systems (NIPS)*, 2014.

B. D. Ziebart, A. L. Maas, J. A. Bagnell, and A. K. Dey. Maximum entropy inverse reinforcement learning. In *National Conf. on Artificial Intelligence (AAAI)*, 2008.