Multimodal Robot Learning for Grasping and Manipulation

by

Simon Benjamin Stepputtis

A Dissertation Presented in Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy

Approved November 2021 by the
Graduate Supervisory Committee:

Heni Ben Amor, Chair
Chitta Baral
Yezhou Yang
Stefan Lee

ARIZONA STATE UNIVERSITY

December 2021

ABSTRACT

Enabling robots to physically engage with their environment in a safe and efficient manner is an essential step towards human-robot interaction. To date, robots usually operate as pre-programmed workers that blindly execute tasks in highly structured environments crafted by skilled engineers. Changing the robots behavior to cover new duties or handle variability is an expensive, complex, and time-consuming process. However, with the advent of more complex sensors and algorithms, overcoming these limitations becomes within reach. This work proposes innovations in artificial intelligence, language understanding, and multimodal integration to enable next-generation grasping and manipulation capabilities in autonomous robots. The underlying thesis is that multimodal observations and instructions can drastically expand the responsiveness and dexterity of robot manipulators. Natural language, in particular, can be used to enable intuitive, bidirectional communication between a human user and the machine. To this end, this work presents a system that learns context-aware robot control policies from multimodal human demonstrations. Among the main contributions presented are techniques for (a) collecting demonstrations in an efficient and intuitive fashion, (b) methods for leveraging physical contact with the environment and objects, (c) the incorporation of natural language to understand context, and (d) the generation of robust robot control policies. The presented approach and systems are evaluated in multiple grasping and manipulation settings ranging from dexterous manipulation to pick-and-place, as well as contact-rich bimanual insertion tasks. Moreover, the usability of these innovations, especially when utilizing human task demonstrations and communication interfaces, is evaluated in several human-subject studies.

ACKNOWLEDGMENTS

Thinking back to being an undergraduate student, I never envisioned pursuing a doctoral degree; however, my advisers during my bachelors and masters thesis, David Vogt and Prof. Bernhard Jung, showed me the exciting world of robotics research. They introduced me to the world of research and gave me the courage to pursue a Ph.D. program in the United States. For that, I would like to thank them wholeheartedly. Of course, at this point, I would also like to thank my mother, Gisela Stepputtis, as well as my sisters Anne Rohde and Eva Stepputtis, who always encouraged me to pursue my goals, even if they were far from home, as well as their continued support of my decision to study abroad. Pursuing a Ph.D. program is certainly not always easy and comes with many challenges along the way. In these challenging times, my adviser, Heni Ben Amor, was always a supportive constant, and I would like to thank him specifically for believing in me and my research, even when reviewer number two told me to better go and change careers. In addition to my adviser, I would also like to thank Chitta Baral, Yezhou Yang, and Stefan Lee, who fortunately agreed to join my committee and advised me along the way on many topics related to my work, but also related to career goals, life in academia, and many more things. While not part of my committee, I would also like to thank Mariano Phielipp for his continuous support and advice. Furthermore, this would also not have been possible without my close friends Joseph Campbell and Mehrdad Zaker Shahrak, who not only provided advice on my research but also made me feel welcome in the United States and became good friends, even roommates, during my time at Arizona State University. Finally, I would also like to thank the many members of the Interactive Robotics Lab that have been an invaluable part of this journey, particularly Kevin Luck, Geoffrey Clark, Mark Strickland, Shubham Sonawani, Trevor Richardson, and Trevor Barron.

TABLE OF CONTENTS

CHAPTER Page

CHAPTER Page

LIST OF TABLES

LIST OF FIGURES

Figure Page

Chapter 1

INTRODUCTION

An emerging field in which robots are applied are in-home environments, requiring the execution of various manipulation tasks, including picking, pouring, and placing objects. Enabling robots to complete a task in our environments is usually restricted to executing tasks in highly constrained environments that skilled engineers have carefully crafted. Changing the robot's initial behavior to cover new duties or handle variability is an expensive and complex endeavor. However, as intelligent algorithms slowly start to take advantage of more complex sensor modalities and utilize techniques that are inspired by the human learning process, more complex skills can be learned in an intuitive and efficient manner. When deploying robots in highly dynamic environments, they need to quickly learn or adapt previously learned skills to the particular environment without requiring users to tediously program the desired behavior. A popular approach to learning such behaviors is imitation learning (Schaal, 1999), which has successfully been used in various areas of robotics, including locomotion, grasping, and even more complex actions like handover tasks or playing table tennis (Chalodhorn *et al.*, 2007b; Vogt *et al.*, 2018; Mülling *et al.*, 2013). Furthermore, taking inspiration from the human learning process in which multiple sensor modalities are combined to learn a skill is an important step towards allowing robots to interact in our environments. For example, imagine learning how to swing a tennis racket with the goal of hitting the ball. From a human perspective, we would use vision and object tracking to follow the ball's motion, use tactile sensing to feel we hit the ball with the racket, and maybe also follow instructions given by a trainer that telling us to *"hold the racket firmly and tilt it into the direction you want the ball*

to go!". Correlating the relevant information from each input modality enables us to learn the desired skills. Similarly, intelligent algorithm design can extract correlations between multiple modalities from a set of demonstrations by using imitation learning.

With imitation learning, new motor skills can easily be learned by observing a human expert or by experiencing a kinesthetic demonstration of the task. However, while motion information can be derived from execution traces, semantic information about the intended task is difficult to learn since no adequate communication channel exists between the human expert and the robot. For example, teaching the robot how to pick up a mug can become challenging without being able to explicitly tell the robot what the objective is. Incorporating an additional communication channel into the training process enables the demonstrator to outline the key concept of the demonstrated task. In case of learning to pour a cup of coffee from a can held by the robot, the demonstrator would not only provide a demonstration of the necessary movements but also explain what the shown motion is accomplishing, namely to *"pour the coffee into the cup"*. With this additional input modality, a context between the motion and the resulting action in the environment is created, allowing the system to interrelate these modalities to encode the correlations between the various input modalities.

To this end, this work presents systems that can learn various modalities in multiple grasping and manipulation settings ranging from dexterous manipulation and pick-and-place tasks to contact-rich bimanual insertion. Figure 1.1 provides an overview of the areas covered in this thesis:

Tactile Dexterity utilizes tactile sensor information to learn a predictive model that allows the robot to complete in-hand manipulation tasks by leveraging slippage. This model uses a single modality to predict a suitable control signal in a

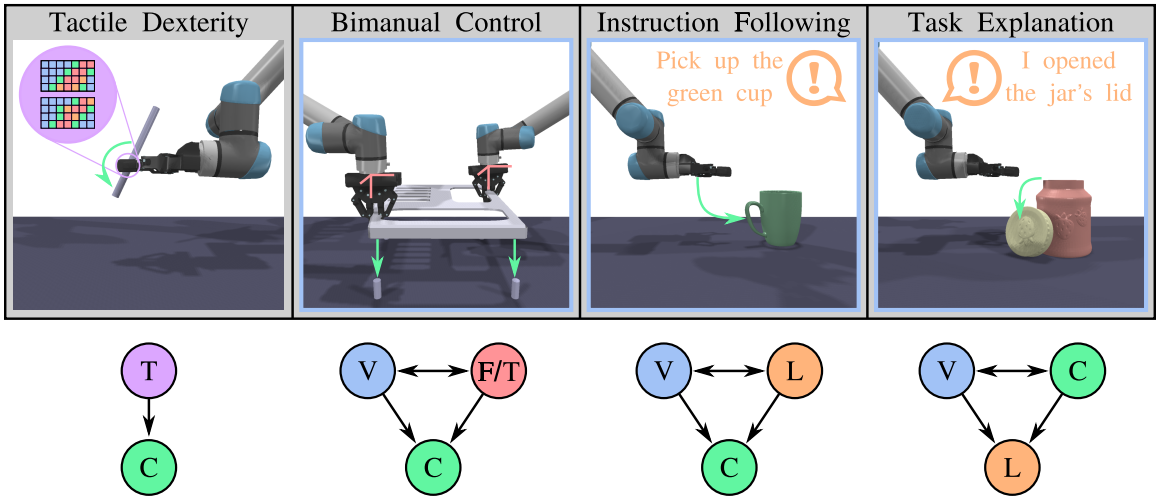

Figure 1.1: Scope of the Dissertation

The four figures show the main contributions of this work and the interaction between the used modalities: T: tactile sensing, C: robot control, V: vision, F/T: force/torque sensors, L: language

self-supervised approach. Further details can be found in chapter 2.

Bimanual Control orchestrates multiple robot arms to complete a contact-rich insertion task. In this task, vision and force/torque sensor data are combined to generate a control signal for the robots. This task uses a Bayesian approach to maneuver the robots in an over-constrained manipulation task in conjunction with an admittance controller to account for potential forces between the robots. Furthermore, chapter 3 also evaluates how humans can efficiently and intuitively provide demonstrations for multiple robots at the same time.

Instruction Following uses a supervised approach to translate verbal instructions into low-level robot control policies for picking and pouring tasks. During training, the robot learns to interrelate the demonstrated motion with the perception of the environment and the verbal instruction, allowing it to generate a suitable control policy

that is specific to the desired action. The resulting language-conditioned visuomotor policies can then be conditioned at run-time on new human commands and instructions, which allows for more fine-grained control over the trained policies. Further details can be found in chapter 4.

Task Explanations are generated in work presented in chapter 5 and utilizes vision and motion traces of the robot to generate a verbal description of the action that has been performed by the system. With this additional communication channel, full human-robot interaction can be achieved.

Future Work Finally, chapter 6 provides a future perspective of this work by utilizing language to learn a reward function instead of directly translating it into a control policy. A benefit of this approach is that (a) language might be better suited to describe the high-level goal of a task rather than being translated into low-level control signals and (b) it allows the robot to freely learn a policy in a reinforcement learning setting while only being guided by the applicability of the generated motion, rather than the individual steps of the motion.

1.1 Completed Work

Interaction with the real world is a basic requirement for robots, ranging from repetitive tasks like the assembly of cars or other common devices in well-defined environments to highly specific actions in dynamically changing environments found in our own homes. Not only does the state of the environment influence the robot's target behavior, but the target might even change between or during interactions, e.g., based on the changing objective of a human operator. Successfully acting in ever-changing environments and fulfilling dynamically changing objectives requires

robots to sense their environment with a variety of sensors and change their behavior towards ever-changing expectations.

In my early work, I focused on expanding the robot's sensing capabilities by incorporating tactile sensing into a manipulation task that utilized slippage to position a grasped object in the desired configuration (Stepputtis and Ben Amor, 2017a,b). With the addition of further modalities, in this case, tactile sensing, the robot's repertoire of skills was expanded to complete previously challenging actions. A detailed explanation of the conducted work can be found in chapter 2 and was published in Stepputtis *et al.* (2018b).

A limitation of the previous work is the inability of the robot to adapt its behavior to varying tasks or expectations. In order to adhere to changing requirements, the robot needs to be able to incorporate an intention into its motion generation, ideally by using a quick and intuitive approach that can easily be facilitated by a non-expert user. One such way of conveying user intentions to a robot is by using natural language. However, natural language is difficult to use in such a context due to its inherent ambiguity, context, and semantic meaning. Communication between humans usually assumes a certain degree of context, which can vary based on cultural backgrounds and between individuals, as well as semantic meaning, where some phrases should be interpreted differently based on cultural and social paradigms (Meyer, 2014). However, translating this ambiguous input into suitable low-level robot control policies is detailed in chapter 4, based on the preliminary work presented in Stepputtis *et al.* (2018a, 2019, 2021). Furthermore, the work in Stepputtis *et al.* (2019) received the best poster award by Nvidia, and the work in Stepputtis *et al.* (2020a) was selected as a spotlight presentation at NeurIPS 2020.

Chapter 2

TACTILE DEXTERITY THROUGH ACTIVE SLIP CONTROL

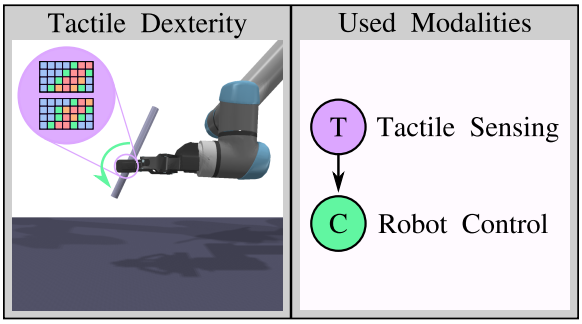

Figure 2.1: Tactile Modalities

Overview of the used modalities in the dexterous manipulation task

In this chapter, we present a machine learning methodology for actively controlling slip. Leveraging recent insights in deep learning, we propose a Deep Predictive Model that uses tactile sensor information to reason about slip and its future influence on the manipulated object. Instead of learning object manipulation from motion traces alone, tactile sensing is utilised as an additional input modality to enable additional manipulation skills. Figure 2.1 shows an overview of the used modalities from tactile sensing to control and the desired dexterous manipulation task. The obtained information is then used to precisely manipulate objects within a robot end-effector using external perturbations imposed by gravity or acceleration. We show in a set of experiments that this approach can be used to increase a robot's repertoire of motor skills.

The work presented in this chapter has been published and presented as a full conference paper at the International Conference on Robotics and Automation (ICRA) 2018.

2.1 Introduction

Throughout our lifetime, we learn to delicately grasp, manipulate, and use a wide range of objects. Repeated physical contact with our environment allows us to acquire complex motor skills, such as in-hand rotation of objects. In doing so, we also learn to actively use slip to our advantage. By contrast, in the robot grasping literature, approaches to slip detection often aim at reducing or eliminating the effects of slip Veiga *et al.* (2015); Ma *et al.* (2015); Stachowsky *et al.* (2016); Cirillo *et al.* (2017); Cavallo *et al.* (2014); Shaw and Dubey (2016). To this end, they require access to robust and accurate models of slip dynamics which, due to the non-deterministic nature of slippage, is a challenging problem in its own right.

In this paper, we discuss how slip can be actively controlled to increase robot dexterity. Previous approaches to slip control have focused on a theoretical analysis of the underlying forces, torques, and physical constraints Cirillo *et al.* (2017). However, in practice, such models are often infeasible since they fail to represent the uncertainty and variability inherent to manipulation tasks involving slip. We argue that a critical component necessary for active slip control is a stochastic model of dynamic object behavior under such conditions.

We therefore propose a Deep Predictive Model (DPM) which can be used to effectively learn the relationship between robot actions, incurred slip, and future object poses. A key feature of this approach is that we are able to manipulate a grasped object in-hand even when using a robot gripper with a single degree of freedom (DOF). This type of action is possible by utilizing *extrinsic dexterity* – external perturbations and forces such as gravity Dafle *et al.* (2014) that are actively utilized towards object manipulation. To account for the non-deterministic nature of slippage, we use stochastic forward passes (also referred to as *Monte Carlo Dropout*) Gal (2016) to

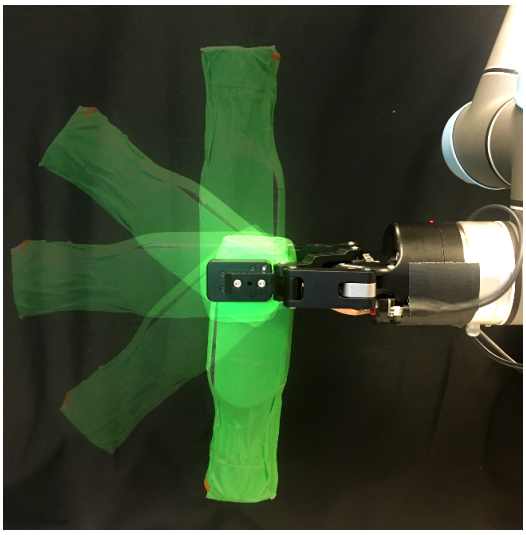

Figure 2.2: Dexterous Manipulation with Slip

Slip can be utilized to rotate an object to different target locations. This image shows five example positions of a green bottle. The robot is able to utilize slippage to rotate the object to the desired angle.

generate a probabilistic belief over predicted object states. Accordingly, instead of relying on point-estimates, we can compute and analyze a distribution over object poses, thereby gaining a deeper understanding of the prediction process Gal (2016). Finally, we perform experiments and provide examples for manipulating multiple objects having different surface structures, shapes and masses. These experiments show how this approach can be leveraged to achieve dexterous manipulation with robot hands that have a limited number of degrees of freedom. The experiments also provide evidence indicating that such learned models generalize to new, untrained objects.

2.2 Related Work

A large body of work on modeling slip has focused on prediction and prevention Shaw and Dubey (2016) using tactile sensing Veiga *et al.* (2015); Ma *et al.* (2015);

Stachowsky *et al.* (2016); Cirillo *et al.* (2017); Cavallo *et al.* (2014). Other research focused on utilizing tactile information to classify linear or rotational slip by training a neural network to discriminate between these two types of slip Su *et al.* (2015). Going beyond simple detection, slip has also been used for dexterous in-hand manipulation of grasped objects Jara *et al.* (2014). An approach that only uses tactile sensing to rotate an object is described in van Hoof *et al.* (2015). A drawback of that approach, however, is that it still requires an external object to support the movement instead of utilizing gravity. The work presented in Stachowsky *et al.* (2016); Shaw and Dubey (2016); Cirillo *et al.* (2017) focuses on grasping an object with the lowest possible force by calculating/estimating the friction coefficient between the gripper and the object. When utilizing slippage, knowing the friction coefficient between the gripper and the object is crucial. Especially the work in Cirillo *et al.* (2017) made an important effort in determining this coefficient for unknown objects while dynamically adjusting its estimation. However, the goal of this paper is to present a system that utilizes slippage to rotate objects in-hand, rather than estimating friction, resulting in a simple grid-search to determine friction.

In this work, slip is not considered an undesired source of perturbation, but rather as an opportunity for dexterous manipulation. Compared to Jara *et al.* (2014), the presented system is able to solely rely on tactile sensing for in-hand manipulation and does not need to use a supporting object as in van Hoof *et al.* (2015). In addition, no analytical model of the underlying mechanics and forces is needed as in Cirillo *et al.* (2017), since the DPM is able to learn the necessary relations to successfully manipulate the grasped objects.

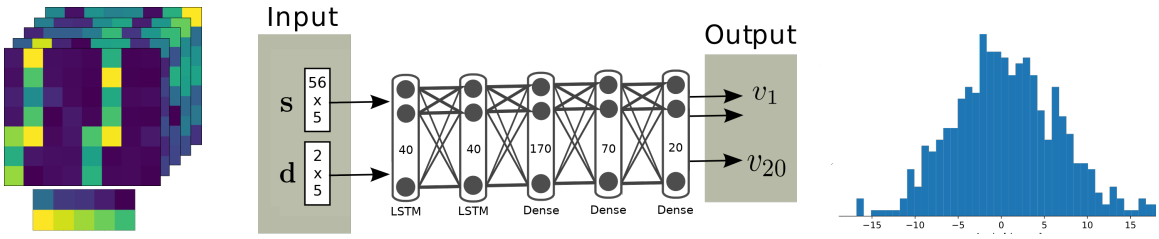

Figure 2.3: Deep Predictive Model for Active Slip Control

Schematic visualization of the deep predictive model used in this work. An example input for the DPM is shown on the left left, containing the static and dynamic tactile data. The right shows and Bayesian distribution over the output of the network for one exemplary value from $\mathbf{v}$.

2.3 Methodology

When utilizing slip to re-orient an object in the gripper, a robot needs to predict the implications of its actions on the pose of the manipulated object. For this purpose, a Deep Predictive Model (DPM) is trained to predict the absolute rotation of the object at run time. More specifically, we train a neural network with three hidden layers, as shown in Figure 2.3, to act as a DPM. The first two layers are recurrent layers to leverage temporal features, followed by a flattening layer and three fully connected dense layers which are separated by Dropout layers. These Dropout layers will later be used for *Monte Carlo Dropout* Gal (2016). The network structure was determined empirically by running a grid search. Once trained, the DPM generates an estimate of the object's future orientation for the next 20 time steps. Predictions are generated based on the static tactile data, allowing it to measure absolute pressure values and locations as well as dynamic tactile data. Dynamic sensors provide information about contact changes and events involving vibrations Cutkosky and Ulmen (2014). Another benefit of dynamic sensors is their ability to quickly respond to changes in sensation Stern (2017), which is especially useful for detecting slippage.

Figure 2.4: Manipulated Objects using Slippage

Overview of the objects used for training and evaluation. The two left most objects were used for training and are referenced as "Training A" and "Training B". Both are equipped with an acceleration sensor.

Due to their differential property, temporal features can be inferred from these sensors. Formally, the prediction process can be represented by the following equation by utilizing a sliding window of the past five time steps:

$$f_n(\mathbf{s_{t:t-5}}, \mathbf{d_{t:t-5}}) \rightarrow \mathbf{v_{t:t+20}} \tag{2.1}$$

where $\mathbf{s} \in \mathbb{R}^{56}$ is the vector of static tactile sensor activations and $\mathbf{d} \in \mathbb{R}^2$ is the vector of dynamic tactile sensor values. The output $\mathbf{v}$ is a vector containing the absolute rotations for the next twenty time steps $\{v_t, \cdots, v_{t+20}\}$, where one time step is set to a hundredth of a second. It is important to note that the predicted angle is given relative to the gripper. In addition to the recurrent layers, the neural network can also infer latent features about the changes from the dynamic sensors without solely resorting to recurrent layers. This is due to the sensors' ability to capture rapid changes that are characteristic for slippage Stern (2017).

Table 2.1: Properties of the Used Objects

	Object							
Property	1	3	3	4	5	6	7	8
Mass [g]	519	99	174	119	124	121	194	104
S-Rad. [cm]	4.6	1.6	3.2	2	2.2	3.8	2.1	2.5
Deformable	yes	no	no	yes	no	yes	no	no

Since slippage is a highly nondeterministic process, we use a Bayesian neural network to predict a distribution over its results Gal (2016). To this end, we leverage recent theoretical insights from Gal (2016) for estimating model uncertainty within the DPM. According to Gal (2016), multiple stochastic forward passes using a stochastic version of dropout are identical to variational inference in Gaussian processes. This is used to validate the accuracy of the neural network regarding different target angles of the test data set. For this purpose we activate dropout at test time to gain a better understanding of how the network performs in certain regions of the input space. The results from these analyses can be found in section 2.4.

2.3.1 Training

For training purposes, the ground truth data is collected by measuring the angle of the object with an acceleration sensor that is embedded or attached to the two training objects, as can be seen in Figure 2.4. The objects were chosen based on their form and rigidity. Training object A is slightly deformable, has a nearly flat surface and weighs about 500 grams. The second training object, object B, is not deformable, has a radius of 1.5 centimeters and weights about 100 grams. The acceleration sensor

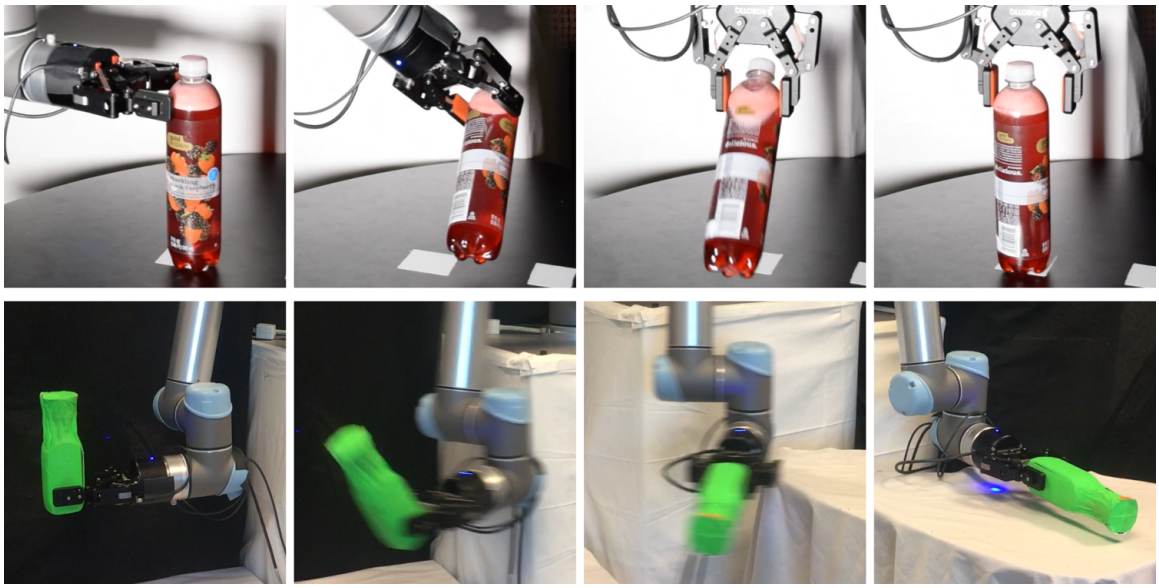

Figure 2.5: Manipulation with Active Slip Control

The above sequences visualize two scenarios for in-hand manipulation using active slip control. The top row shows an experiment in which slip is used to keep the object in the same global rotation even though the gripper is rotating. The bottom row shows how to rotate an object using centrifugal forces without any rotation of the robot gripper. The main difference between these two experiments is the way in which slip is induced. Slip occurs as either the result of rotation within the gripper or as an effect of the acceleration of the gripper.

is used to automatically label the collected training data. The sensor is calibrated to accurately represent the angle of the object between -20 and 220 degrees by using a Gaussian process Geng *et al.* (2015). The calibration was done by firmly holding the object in the gripper of the robot and using its controls to change the rotation of the gripper stepwise by one degree and collecting the 3 DOF data from the accelerometer. Zero degree was set to represent an upright object as can be seen in Figure 2.2 at the object's twelve o' clock position, continuing counter clockwise with increasing value. The evaluation of the calibrated acceleration sensor resulted in an average error of

0.4 degrees with a maximum error of 2.1 degrees.

Formally, the training data can be written as follows (Equation 2.2)

$$(\mathbf{s}_{t:t-5}, \mathbf{d}_{[t:t-5]}, v_{t:t+20}) \tag{2.2}$$

s denotes a concatenated list of five vectors with the most current static tactile data. d holds the corresponding dynamic tactile data over the past five time steps. **v** represents a vector of the next $t : t + 20$ object angles that where taken from the training data. One training example is constructed by randomly selecting one observation at time step t_0, representing the current observation $(\mathbf{s}, \mathbf{d})$. The sliding window over **s** goes from t_{-5} to t_0 and the values of **v** range from t_0 to t_{+20}. Additionally we made sure that no training example exceeds the boundaries of one experiment, since multiple demonstrations were used in the training process. If n is the number of observations from a particular training example, the maximal chosen time t_0 will be t_{n-20}.

2.3.2 Run Time

The network uses the static and dynamic tactile data to make a prediction regarding the object's angle for the next 20 time steps. Each prediction represent the position of the object and is used to decide whether or not the gripper needs to close to stop the object at the desired position. The benefit of predicting multiple steps into the future is the ability to select the correct amount of lookahead to resolve any network and control latencies in the pipeline. Based on the predictions, a simple controller as shown in Equation 2.3 is used to close the gripper as soon as the predicted angle a is equal to or greater than the target angle.

$$f_c(\mathbf{v}, l) \rightarrow a \tag{2.3}$$

An important parameter within Equation 2.3 is the value of l, which indicates the amount of lookahead used to best compensate for latencies. During training, the network does not experience any latencies, but at run time the prediction of $\mathbf{v}[0]$ gives a rotation that would already be obsolete when received by the controller due to latencies in the pipeline. Using the gripper controller solely based on these old predictions leads to constantly overshooting the target angle. On the other hand, making control decisions based on information that is too far into the future will result in undershooting the target. Besides the latencies, the angular velocity of the currently slipping object influences the latency error. If the object has a high angular velocity, it takes longer to slow down the object due to mass inertia until it is held firmly in the gripper. In this work, we empirically evaluated that a value of 19 for l yields to a good approximation of the final angle.

2.4 Evaluation

For the experiments, we are using a UR5 robotic arm equipped with the Robotiq adaptive two finger gripper. The two fingertips of the gripper are replaced with a tactile sensorLe *et al.* (2017). Each of the two fingers has a 4x7 sensor matrix for static tactile data as well as one dynamic tactile sensor to leverage temporal features.

To train the network, a total of 200 demonstrations of a 180 degree slip were recorded, 100 with every training object, resulting in 90,200 samples. Since the object has a different angular velocity during the slipping motion, these samples are not equally distributed over all angles. Analyzing the collected data showed that there are at least 200 samples for every angle between 0 and 180 degrees, which was used as the upper limit for all other angles. Without the previously mentioned preparation of the training data, the network would learn to have a strong preference towards low angles where the movement was still slow, resulting in much more training data

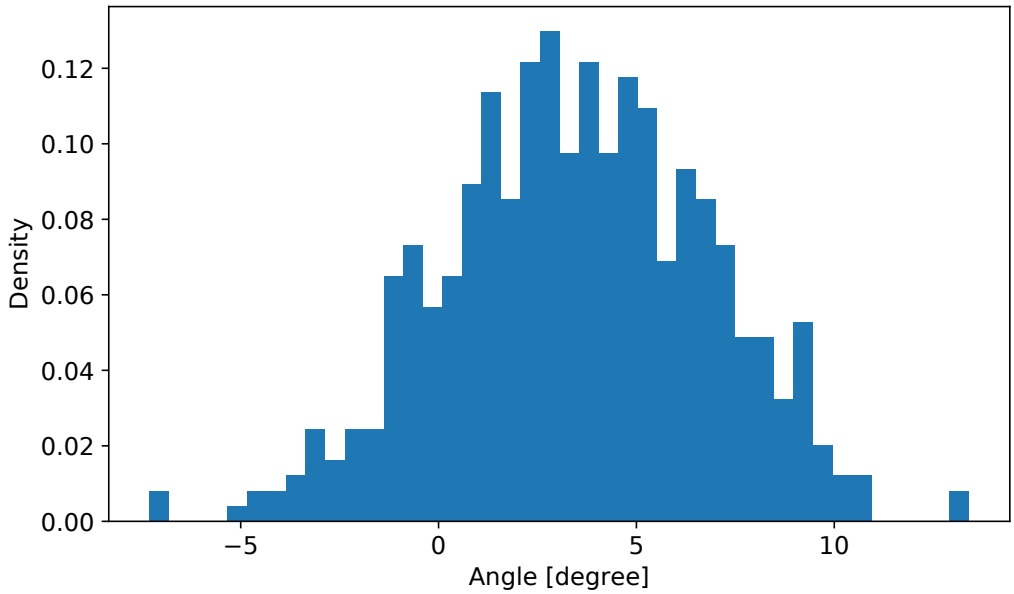

Figure 2.6: Slipping Object Pose Uncertainty: 90 Degree Target

Histogram of the predicted object values for 500 stochastic forward passes. Results are shown for time step 19 at a target angle of 90 degrees.

for these regions. The difference is roughly a factor of 25. Additionally, all angles below 13 degrees were cut out of the data set, since the robot was given an initial slope of 10 degrees to allow slippage of the grasped object. After these adjustments, the dataset contained 32,000 samples which were then split into 70% training, 20% validation and 10% testing data. Furthermore, all data points are normalized before they are used in the training process. To accurately represent the movement, all data points are collected at a rate of 100 Hz, resulting in an average resolution of one sample per degree. The demonstration ends as soon as the object reaches a total rotation of more than 180 degrees, measured with the embedded acceleration sensors within each training object. One training example is roughly illustrated in Figure 2.2 where the values mentioned above are collected while the object is rotating counter clockwise. Adaptive moment estimation (Adam) Kingma and Ba (2015) is used for

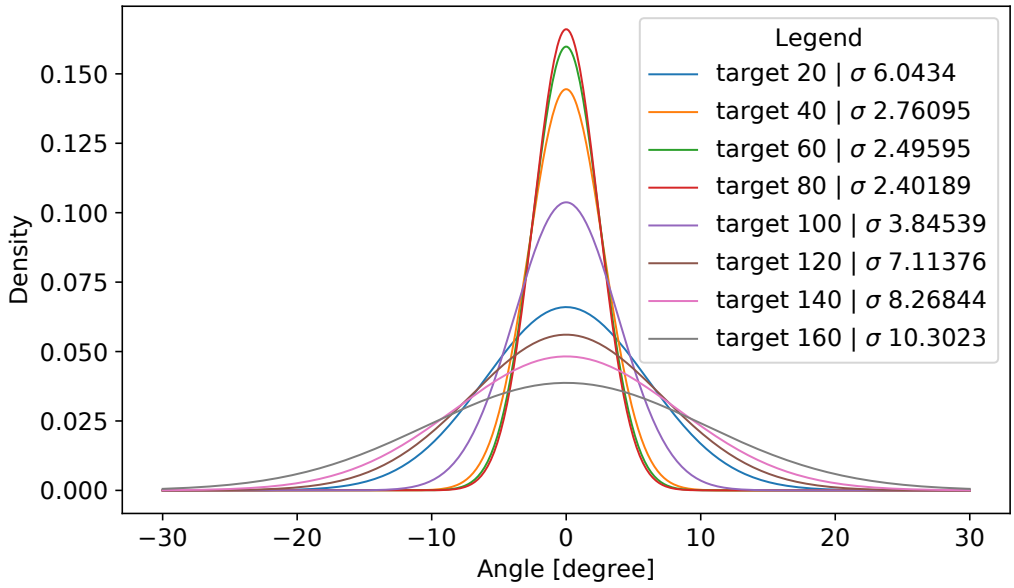

Figure 2.7: Slipping Object Pose Uncertainty: Varying Targets

Stochastic forward passes for different targets. Each distribution is based on 500 predictions of the same input. The plot shows the distribution of the predicted angle for different targets. A smaller variance indicates a DPM with higher certainty in its predictions.

training using a learning rate of 0.001 in combination with mean squared error as loss function.

We evaluated our approach with two experiments. The first experiment uses the test data from the dataset, whereas the second experiments evaluates the network with eight different objects (2.4) on an actual robot (2.2).

2.4.1 Nondeterministic evaluation

For the nondeterministic evaluation we use stochastic forward passes to analyze the certainty of the neural network. This is helpful since slippage is a highly nondeterministic process. With this in mind, we randomly chose samples from the training

set that were labeled with 90 degrees at $\mathbf{v}[19]$. This sample was provided 500 times to the DPM and the error between the predictions and the 90 degree target was collected. Figure 2.6 shows a histogram of the error between the prediction and the target.

This result indicates that the predicted values can be approximated using an exponential distribution. We assume that a single Gaussian is feasible to represent the data. Based on this assumption, Figure 2.7 shows the normal distribution for different target angles between 20 and 160 degrees, using 20 degree steps. μ is set to zero for all targets for an easier comparison of the variance. Smaller variances indicate that the network has a higher certainty for the predicted angle. As for the previous experiment, the network was provided with the same sample 500 times for each target angle. The samples were randomly chosen from the test set.

The distributions over the data shows that the precision of the network is highest for angles between 60 and 80 degrees. Between these two angles, the variance of the predictions with respect to the ground truth is ~ 2.5 degrees, whereas the variance for higher angles is over 10 degrees.

Even though the ground truth data was collected with 100 Hz, fast object movements have a reduced accuracy compared to slower object movements during the early slippage. This explains the lower precision at higher angles. Another parameter that contributes to the uncertainty at higher angles is the latency in the system, since its influence is higher when the object is moving faster.

Figure 2.8 gives a more general overview over the error. Each box contains the accumulated error for one target angle over all time steps of $\mathbf{v}$ between t_0 and t_{19}. As already seen in Figure 2.7, the error for predictions above 100 degrees increases quickly. The region between 60 and 100 degrees provides a nearly constant error with ± 10 degrees, or ± 4 degrees between the upper and lower quantile. When only

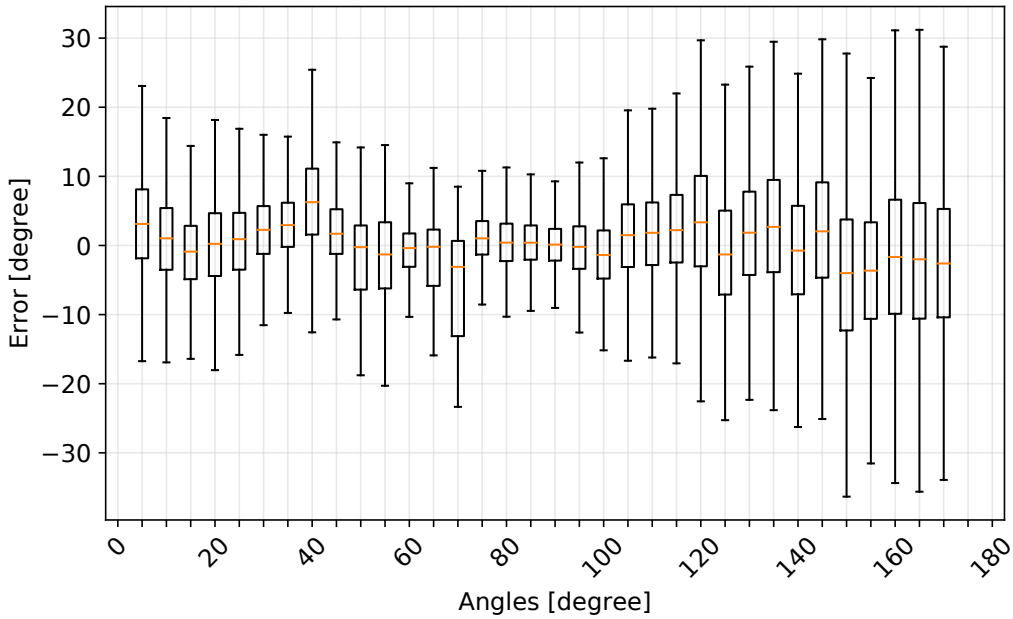

Figure 2.8: Predictive Model Accuracy for Varying Targets

Overview of the neural network regarding different target angles on the test dataset. The error above is accumulated over all predicted time steps.

considering a single future time step like in Figure 2.7, the error decreases to ±2.5 degrees within the first quantile. The slightly increased error below 60 degrees can be explained with the occasional inability of the tactile sensor to distinguish an object at 0 from the same object at 180 degrees. The average rotation time over all objects was approximately 4 seconds.

2.4.2 Generalization with different objects

While the nondeterministic evaluation of the neural network helped to understand its properties and to empirically evaluate the structure of the network, we use a deterministic network for the robot experiments by disabling dropout at run time. This is done to get the average of all the different subnetworks that are trained with dropout Srivastava *et al.* (2014).

At run time, the force used to grasp the object is crucial since it needs to be high enough to still hold the object while being low enough to allow rotational slippage. The amount of pressure between the object and the gripper depends on the surface properties of the gripper and the object, as well as the object's mass. However, these parameters remain unknown. In this work, we used a grid-search approach to find the correct value for each object, such that the grasp allows the object to slip.

Prior to the actual experiment, every object was evaluated regarding the amount of force needed to allow slip. The gripper can be controlled with 255 steps between fully open and closed, where 255 denotes a fully closed gripper. To control the force between the gripper and object, we opened the gripper one unit at a time and observed if the grasped object slips. When the slip for one value exceeded 90 degrees within 7 seconds, the smallest of these values was taken in the experiment later on. For this test we opened the gripper between 0 and 15 units after a full grasp.

In this scenario, the robot picks an object and lifts it up to allow the object to rotate. The grasping process is done in multiple steps. First, the object is grasped with full force, followed by opening the gripper for an object-specific amount of degrees to prepare for slip. The grasp is validated by ensuring that the network is predicting an angle smaller than 90 degrees for five consecutive time steps. This is because the DPM was never trained to handle the grasping process and to prevent any errors in the prediction due to untrained data. To initiate slippage, the robot then gives an initial slope of 10 degrees to the object. This initial slope is enough to initiate rotational slippage, since the grasp of the robot is already loosened. However, it is still tight enough to prevent most of the linear slippage. The objective is to stop the object precisely at 90 degrees.

Figure 2.9 shows the results of our experiments with eight different objects. The

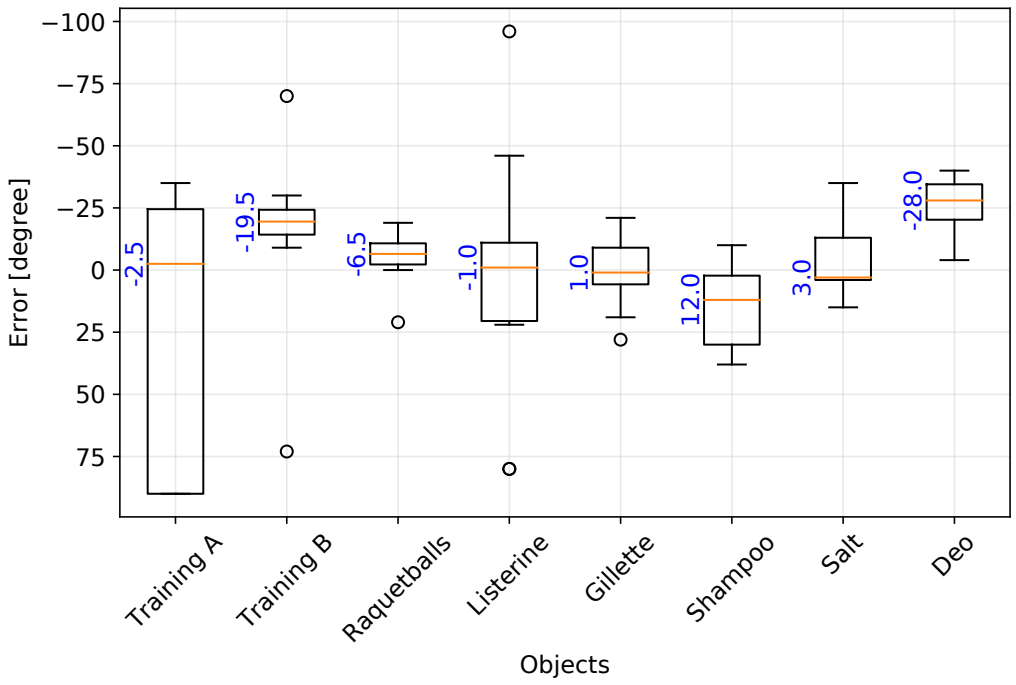

Figure 2.9: Generalization to New Objects

Evaluation of eight objects (including the two training objects). The goal was to stop the object at exactly 90 degrees. Negative values correspond to not reaching the target angle whereas positive values indicate overshooting the target. The blue numbers outline the average offset.

shown data are based on ten target approaches with each object. The order of the objects (from left to right) is equivalent to the order of the objects as shown in Figure 2.4. On the y-axis, positive values indicate that the target of 90 degrees was overshot, while negative values indicate that the object did not rotate enough before the network predicted that the target was reached.

All experiments with the robot use the same DPM that was also evaluated in section 2.4.1. It can be seen from Figure 2.9 that the DPM is able to generalize towards different objects. The objects were chosen due to their different properties

like mass, shape and ductility as outlined in Table 2.1. Even though different objects have a general offset that causes them to slightly over- or undershoot the 90 degree target, the variance of the actual object angles (excluding "Training A") is still within an average ± 8 degree limit for most of the approaches. For these experiments, the object angles were manually measured since six of the eight objects did not have an acceleration sensor. An interesting observation is that the result for the "Training A" object is noticeably worse in comparison to all other objects while overshooting the target by up to 90 degrees. A possible explanation for the error is the weight of the "Training A" object. Since the object already had a high mass inertia when reaching the target, the gripper had a notable problem to stop the object.

2.4.3 Dynamic movements

Especially during fast movements, where an object is more likely to slip due to acceleration or centrifugal forces, our approach is able to utilize normally undesirable slippage to enhance the robot's ability to manipulate objects in-hand. Two examples of these movements can be seen in Figure 2.5.

Preliminary results with these experiments suggest that even though the physics of the experiment were changed to a fully dynamic task, the DPM is able to rotate the object to a 90-degree target. For these experiments, the same DPM was used. Initial experiments showed that the the error between the target and the actual object position slightly increased.

2.5 Conclusion

In this paper, a novel approach for dexterous manipulation is presented that leverages slippage to rotate an object in-hand. We showed that a single data-driven DPM is able to utilize external dexterity to precisely manipulate an object. A benefit of our approach is that we do not need to use an external object and we showed that gravity and acceleration are enough to conduct complex object manipulation even with a 1-DOF gripper. Compared to Jara *et al.* (2014) and Cirillo *et al.* (2017), our approach solely relies on tactile data and does not require a detailed model of the environment. Experiments support the feasibility of this approach in real-world applications and indicate how the approach can increase a robot's repertoire of skill. They further show that the DPM can generalize its predictive ability to different, unknown objects.

A potential expansion of this approach is the transition to a more complex task in contact-rich environment. One such task is peg-insertion, which could greatly benefit from tactile and force/torque sensors. Furthermore, additional modalities could further expand the capabilities of the system towards adaptive and responsive systems.

Chapter 3

BIMANUAL MANIPULATION FOR CONTACT-RICH TASKS

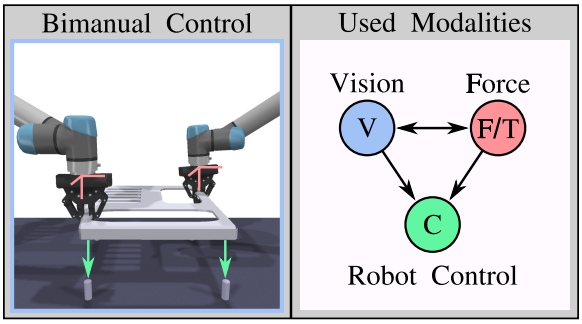

Figure 3.1: Bimanual Modalities

Overview of the used modalities in the bimanual insertion task.

The previous chapter used tactile data to perform dexterous manipulation; however, input data from the force domain can also be used for more complex tasks like contact-rich insertion. Figure 3.1 describes how vision and force/torque sensors can be used as part of a framework to learn a complex bimanual control task by imitation. To this end, we present a system and algorithms for learning compliant and contact-rich robot behavior from human demonstrations. The presented system combines insights from admittance control and machine learning to learn control policies that can (a) deal and adapt to a variety of disturbances in time and space, while also (b) effectively leveraging physical contact with the environment. To this end, we discuss techniques for identifying and recovering from perturbations. We demonstrate the effectiveness of our approach using a real-world insertion task involving the multiple simultaneous contacts between a manipulated object and insertion pegs. We also investigate efficient means of collecting training data for such bimanual settings.

To this end, we conduct a human-subject study and analyze the effort and mental demand as reported by the users.

3.1 Introduction

Manipulation still remains a critical challenge of robotics. Over the past decades, there has been tremendous progress in endowing robots with motor skills for grasping and dexterity. However, the vast majority of work in this field focuses on scenarios involving a single robot arm and tightly controlled physical interactions with the environment. Making early or premature contact with a target object may create forces that may seriously jeopardize the manipulation process. With decreasing prices for robot arms, as well as the proliferation of collaborative and humanoid robotics, there is increased need for techniques that enable reliable, efficient and safe bimanual manipulation. Bimanual robots need to perform manipulation tasks that involve multiple points of contact and dynamic force exchange with objects (and humans) in their surroundings, e.g., lifting a box, inserting a tight-fitting part, unscrewing a bottle cap, or reacting to a human push. In addition to physical interaction with their surroundings, the individual robot arms in a bimanual setup may themselves be exchanging forces and torques through the manipulated objects. Hence, compliant control policies are required that allow bimanual robots to (a) deal and adapt to a wide variety of disturbances in space and time, and (b) effectively leverage physical contact to their advantage. Yet, designing control algorithms that can bridge these (potentially conflicting) requirements can be challenging and time-consuming.

In this paper, we propose an imitation learning framework for extracting compliant bimanual policies. We describe a robotic setup in which human demonstrations of the task are recorded across a variety of sensing modalities. The setup leverages

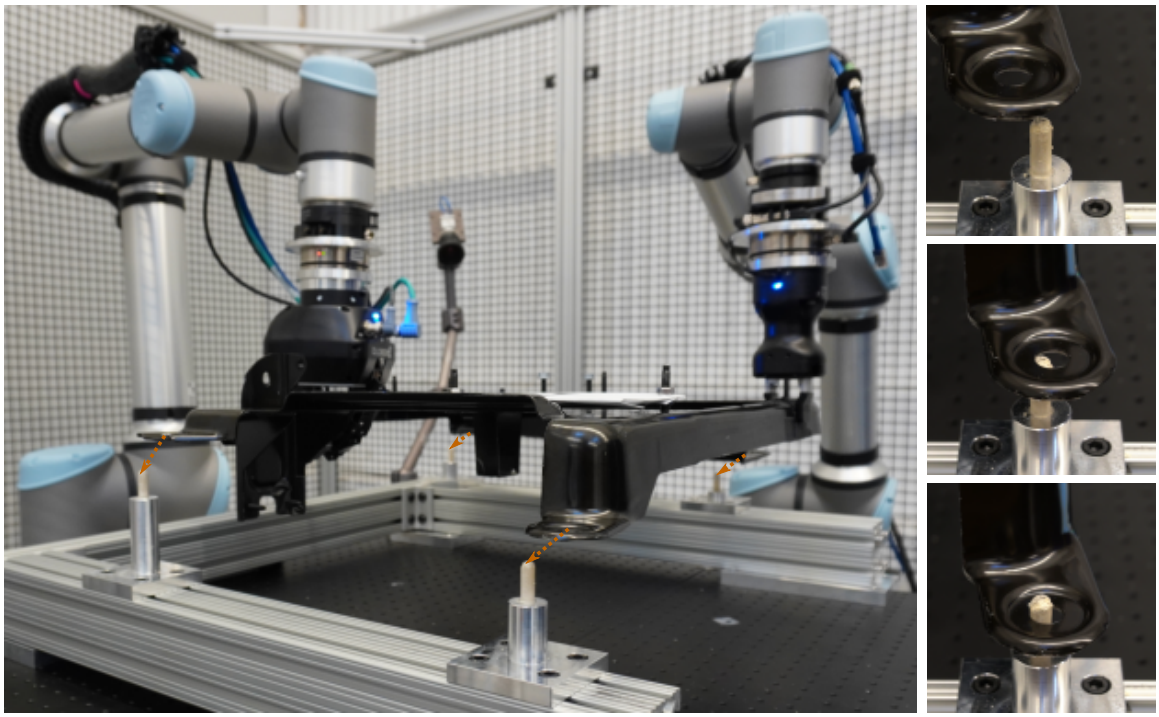

Figure 3.2: Bimanual Insertion Task Overview

Overview of the bimanual insertion task in which the two robots need to jointly insert the bracket onto four pins. The detail pictures on the right indicate the corrective abilities of our proposed method.

principles of admittance control to enable contact-rich and dynamic demonstrations without the risk of collisions, damage or wear-and-tear. In turn, the recorded data is used to learning a Bayesian Interaction Primitive (BIP) which encodes the demonstrated behavior in time and space. At runtime, the BIP is used to identify (a) the temporal task progress as a function of external perturbations and (b) the optimal robot response to these perturbations and environmental conditions. In cases where a physical perturbations affects the task execution, the robot is then able to correct its actions by performing corrective actions in either time or space. Since the presented approach is Bayesian in nature, it allows for powerful spatiotemporal inference from

multimodal datastreams. We show in experiments in a real-world insertion task with tight tolerances that our approach results in compliant, responsive and contact-rich manipulation. To this end, we also investigate the influence of user interface for collecting training data in such a bimanual setting. We conduct a human-subject study using both kinesthetic teaching and space mouse teleoperation and evaluate the intuitiveness, ease-of-use and mental load as reported by the users.

3.2 Related Work

Common manipulation tasks in the real world are usually setup in a bimanual affortance, designed to be completed by a human. Yet, robotics mostly focuses on single arm control with specialized grippers to complete manipulation tasks in highly constrained environments. However, in recent years, there has been increased interest in bimanual control (Chu *et al.*, 2015) and the introduction of various benchmarks (Chatzilygeroudis *et al.*, 2020) and challenges (Hackett *et al.*, 2014) has only increased the interest in automation of bimanual tasks. So far, however, bimanual control has remained to be challenging as controling a single object with multiple robots poses an over-constrained control problem. The work presented in Huang *et al.* (2020a) introduces an impedance model-based optimal control approach to regulate the forces between two robots that are holding on to an object, however, other approaches like admittance control can also be used to address the force-exchange between multiple robots. Other approaches attempt to learn the necessary control from data by using a latent space to decouple high-level effects and low-level motions (Fang *et al.*, 2020). Another recent approach is to use deep learning in which a graph neural network (GNN) (Scarselli *et al.*, 2009) can be used to represent the configuration of the robot (Wang *et al.*, 2018) while also modeling the interaction between it and the

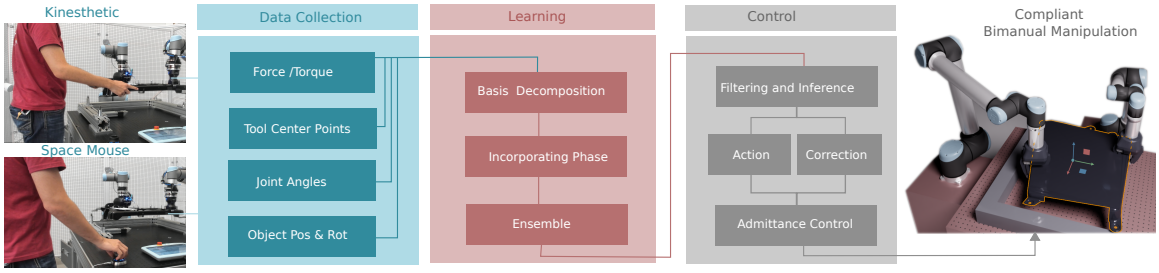

Figure 3.3: Bimanual Learning from Demonstration Pipeline

Overview of our approach from data collection to control. We evaluate two different teaching modalities to learn the task.

environment (Zhu *et al.*, 2018).

However, bimanual manipulation can roughly be categorized into two areas in which the robots either complete a joint task by holding on to the same object to manipulate it (Xie *et al.*, 2020), or robots that only share a common work space and one of the robots is commonly used as a fixture, meaning that it is holding another object in place (Kim *et al.*, 2021; Motoda *et al.*, 2021). Manipulating objects jointly between multiple robots as discussed in Xie *et al.* (2020) and Chitnis *et al.* (2020) addresses many interesting problems that result from jointly manipulating an object, however, the presented tasks are either limited to simulation, or simple lifting tasks. In this work, a bimanual control approach in a contact-rich multi-peg insertion task is introduced that was entirely learned and evaluated on a real-world robot setup.

3.3 Methodology

The main objective of this paper is to learn compliant bimanual manipulation policies from human example demonstrations. Fig. 3.3 shows an overview of our approach. As a first step, human demonstrations of the behavior are provided. To this end, we employ either direct kinesthetic teaching or tele-operation via a space

mouse. We will later discuss the advantages and shortcomings of each one of these input modalities. Once the demonstrations are recorded, they are used in the learning step to extract a Bayesian Interaction Primitive. The BIP is, in turn, used in realtime to infer the spatial and temporal state of the execution of the manipulation task. Inference takes into account all the available sensor sources, e.g., force-torque readings, in order to identify the current progress in task execution and a distribution over potential future robot actions. Based on the generated temporal information, we then determine whether the robot should continue with the task execution or whether a corrective action needs to be performed. The generated action is then sent to an admittance controller to for execution on the robot. A critical element in this control scheme is the realtime inference using Bayesian Interaction Primitives, which will be described subsequently.

3.3.1 Bayesian Interaction Primitives

Each Recorded demonstration $\mathbf{Y}$ is represented as a time series of D-dimensional vectors $\mathbf{Y}_{1:T} = [\mathbf{y}_1, \ldots, \mathbf{y}_T] \in \mathbb{R}^{D \times T}$ where T is the total number of time steps. Of the D dimensions, D_o represent the *observable* state dimensions of the robot, and D_c represent *controlled* dimensions, i.e., robot actions, with $D = D_o + D_c$. Our overall goal in this paper is to learn a bimanual manipulation policy that can generate the accurate robot controls from observed states. Given the current state of the robot, the policy should generate optimal control signals that mimic the behavior of the human demonstrator. However, in the context of human-robot interaction and physical contact with the environment, it is also critical that the manipulation policy adapts to external perturbations. This may be, for example, a push from a human partner or forces generated from premature contact with a screw.

To learn such a policy, we use Ensemble Interaction Primitives and learn a generative probabilistic model over the training demonstrations. However, instead of immediately using time-discretized data, EnBIP transform the recorded demonstrations into a time-invariant representation by performing a basis function decomposition. Such decompositions have been popularized in , and have since been used in a number of motor primitive formulations . Applying the basis function decomposition, we now approximate each state dimension $\mathbf{Y}_t^d = \mathbf{\Phi}_{\phi(t)}^\intercal \mathbf{w}^d + \epsilon_y$ through a linear combination of B basis functions $\mathbf{\Phi}_{\phi(t)} \in \mathbb{R}^{B^d}$ with corresponding basis weights $\mathbf{w}^d \in \mathbb{R}^{B^d}$. Note the shift into a relative time measure known as *phase* $\phi(t) \in \mathbb{R}$, where $0 \leq \phi(t) \leq 1$, as well as the approximation error ϵ_y. The above decomposition generates compact weight vectors $\mathbf{w}$ of the same length d and is therefore a practical first step towards efficient encoding and modelling of the recorded data. Concatenating all basis weight vectors for the individual dimensions then forms the compressed representation $\mathbf{w} = [\mathbf{w}^{0\intercal}, \ldots, \mathbf{w}^{D\intercal}] \in \mathbb{R}^B$ of a given trajectory where $B = \sum_d^D B^d$. In turn, we can now extract a probability distribution $p(\mathbf{w})$ over all given demonstrations. Sampling from this distribution generates an sample trajectory containing all observed sensor states and robot controls in the bimanual task. Similarly, we can also condition on previous states of the robot to yield a posterior distribution $p(\mathbf{w}_t | \mathbf{Y}_{1:t}, \mathbf{w}_0)$ over *future* states and controls.

However, a critical insight in Bayesian Interaction Primitives is the interplay between temporal and spatial reasoning. An estimation error can be the result of either errors in time or space. The inference mechanism in Eq. 3.1 assumes that the current time step or phase are known. This is typically only the case, when the robots behavior is unencumbered by physical interactions with humans or the environment. For example, a human operator may hold back the robot by pushing the end-effector.

In such a case, the phase variable needs to be adjusted or set back to prior values. To enable such adaptation, Bayesian Interaction Primitives reformulate the problem as a joint spatio-temporal inference – time and space are coupled and are jointly estimated. This insight is realized by forming a new state vector $\mathbf{s} = [\phi, \dot{\phi}, \mathbf{w}]$. The state vector holds both a spatial component, contained in the basis function weights $\mathbf{w}$, as well as temporal components in the form of the phase ϕ and the phase velocity $\dot{\phi}$. The temporal variables ϕ and $\dot{\phi}$ describe where we are in time and how fast we are progressing with the task. Hence, generating the posterior:

$$p(\mathbf{s}_t|\mathbf{Y}_{1:t}, \mathbf{s}_0) \propto p(\mathbf{y}_t|\mathbf{s}_t)p(\mathbf{s}_t|\mathbf{Y}_{1:t-1}, \mathbf{s}_0). \tag{3.1}$$

now yields the both information about the spatial and temporal aspects of robot control, since these are encoded in $\mathbf{s}_t$. Performing the above inference step can efficiently be done via recursive filtering. To this end, we will use an Ensemble Kalman Filter (EnKF) as proposed in . A major advantage of EnKF is its ability to model complex nonlinear distributions without having to specify any parametric family. More specifically, the initial distribution is immediately formed by the provided demonstrations – no fitting to a parametric family of densities is needed. In addition, EnKF can be used with nonlinear transition functions and observation functions. As a result, linearization errors as found in other filters can be avoided.

We start by defining an ensemble $\mathbf{X}$ of E members shown by $\mathbf{X} = [\mathbf{x}^1, \ldots, \mathbf{x}^E]$. Optimally we want to sample the initial ensemble $\mathbf{X}_0$ directly from the prior $\mathbf{x}_0 \sim p(\mathbf{w}_0)$ for all $\mathbf{x}_0 \in \mathbf{X}_0$; however, since we do not have direct access to $p(\mathbf{w}_0)$, as a data-driven method, it is standard to instead sample from observed training demonstrations. Random selection on ensemble members is reasonable as the ensemble-based filtering approach provides robustness against possible non-Gausian uncertainties, provided the number of ensemble members is not less than the number of example demonstra-

tions $E \leq N$. As a two-step Bayesian estimation method, our first step approximates $p(\mathbf{w}_t|\mathbf{y}_{1:t-1}, \mathbf{w}_0)$ by propagating each ensemble member forward one time step with:

$$\mathbf{x}^j_{t|t-1} = g(\mathbf{x}^j_{t-1|t-1}) + \epsilon_x, \quad 1 \leq j \leq E, \tag{3.2}$$

with constant-velocity state transition operator $g(\cdot)$, and noise error ϵ_x. Next, the ensemble members are updated from the observation and the nonlinear observation operator $h(\cdot)$ via

$$\mathbf{H}_t\mathbf{X}_{t|t-1} = \left[h(\mathbf{x}^1_{t|t-1}), \ldots, h(\mathbf{x}^E_{t|t-1})\right]^\intercal, \tag{3.3}$$

$$\begin{aligned}\mathbf{H}_t\mathbf{A}_t = \mathbf{H}_t\mathbf{X}_{t|t-1} \\ - \left[\frac{1}{E}\sum_{j=1}^{E} h(\mathbf{x}^j_{t|t-1}), \ldots, \frac{1}{E}\sum_{j=1}^{E} h(\mathbf{x}^j_{t|t-1})\right],\end{aligned} \tag{3.4}$$

The deviation of each ensemble member from the sample mean $\mathbf{H}_t\mathbf{A}_t$ and the observation noise matrix $\mathbf{R}$ can then be used to compute the innovation covariance with:

$$\mathbf{w}_t = \frac{1}{E-1}(\mathbf{H}_t\mathbf{A}_t)(\mathbf{H}_t\mathbf{A}_t)^\intercal + \mathbf{R}. \tag{3.5}$$

The Kalman gain is likewise calculated directly from the ensemble, with no need to specify an explicit covariance matrix, with

$$\mathbf{A}_t = \mathbf{X}_{t|t-1} - \frac{1}{E}\sum_{j=1}^{E} \mathbf{x}^j_{t|t-1}, \tag{3.6}$$

$$\mathbf{K}_t = \frac{1}{E-1}\mathbf{A}_t(\mathbf{H}_t\mathbf{A}_t)^\intercal\mathbf{w}_t^{-1}. \tag{3.7}$$

As is typical in recursive filtering, partial observation are sufficient to optimally estimate the full state, which we leverage to generate a posterior over unobservable latent variables, i.e. robot controls. Since the posterior is over weights $\mathbf{w}$ it defines the controls for all future time steps.

By performing this inference scheme in each time step, we can generate posterior distributions that are conditioned on a multitude of sensors. In Fig. 3.4, for example, we can see the effect of conditioning an execution on high sensor readings from the force-torque sensor. In this specific case, the robot learned to change direction away from this force exchange. Note, however, that strategy was acquired in a purely data-driven fashion – how the robot responds to such stimuli is learned from the human expert's demonstrations.

3.4 Evaluation

We evaluate our approach with a bimanual insertion task that requires two robots to jointly position a bracket onto four pegs. One peg is located at each corner of the object, as shown on the right in Figure 3.3. This task has two main challenges, one being that small control discrepancies can have a large impact on the success of the insertion task due to the distance between the pegs, and secondly, in addition to the object interacting with the world, the robots also exchange forces between each other while jointly moving the object. This section introduces the experiment setup in greater detail as well as discusses the different data collection modalities. Finally, we evaluate the contribution of force/torque sensors on the task success rate as well as the generalizability of the proposed method.

3.4.1 Experiment Setup

Our experiment setup consists of two robot arms, one UR10 and one UR5 robot from Universal Robots, each equipped with a force-torque sensor and gripper. For the grippers, we are using the Robotiq two and three-finger grippers on the UR5 and UR10, respectively. Both robots are located in close proximity to each other and

share a common workspace in front of them in which we are conducting our bimanual insertion task. Initially, the robots pick up the bracket shown in figure 3.3 and jointly lift it up with a predefined motion. After lifting the object, the force-torque sensors are initialized to account for the added weight of the bracket, and control is handed over to either the data collection algorithm or the inference controller that attempts to insert the bracket onto the pegs. The following sections describe the data collection, learning, and evaluation process.

Data Collection

Imitation learning is an intuitive approach for humans to demonstrate the desired action to a robot. In this work, we evaluate two different approaches to collecting the necessary training data from the real-world robot setup. Our two approaches either use kinesthetic teaching in which the user is physically moving the object after it has been lifted by the robots or utilize a space mouse to provide six-dimensional control signals for the object that are relayed by the two robots. When collecting demonstrations for the two robots, we are collecting the following modalities: joint angles $\in \mathbb{R}^{12}$, tool center point positions $\in \mathbb{R}^{7}$ for each robot, the position $\in \mathbb{R}^{3}$ and rotation $\in \mathbb{R}^{4}$ of the object, as well as the force/torque sensor data in the form of a wrench at the robot's tool $\in \mathbb{R}^{6}$. Depending on the used training modality, the the force/torque sensor data can be used during task learning. I.e., when using the space mouse demonstrations, the force/torque sensor data can be used later on to give the robots an additional modality to perceive their world, e.g., when being stuck on top of a peg. However, when using the kinesthetic approach to create demonstrations, the force sensors are used for compliant robot movement and record the external forces induced by the demonstrator. As these external forces are not

present during inference, the force/torque sensor data can not be used for learning when using kinesthetically collected data.

Data Representation

Interaction primitives internally use a basis representation to encode each collected demonstration. Using these representations has two benefits: (a) the trajectories are smoothed by being represented by a fixed number of basis functions, and (b), the basis-representation is independent of the actual trajectory length as all trajectories are re-defined by an artificial phase value that ranges from zero to one. Given a basis representation of a demonstration, a trajectory of arbitrary length can be generated by stepping through the phase and sampling from the basis functions. Our method allows us to use different basis functions for each feature; however, during our experiments, we use the same basis function type for each feature dimension in a given modality. The type and number of basis functions have been determined empirically as a trade-off between model complexity and the ability to capture the nuances of the recorded motions. For the following experiments, we chose to represent the robot's joints with a Gaussian model utilizing five basis functions and a variance of 0.1, whereas the tool center points are represented with a Gaussian model with seven basis functions and a variance of 0.2. The object is represented with a Sigmoidal model with 11 basis functions and a variance of 0.15, while the force/torque sensor is represented with a Polynomial model with 5 degrees.

Evaluation Metrics

We consider an insertion task to be successful when all four grommets are placed on their respective pegs and both robots release the object. If a subset of grommets

	Training Data Modality		Features and Datasets during Training		Model Evaluation	
Model	Space-Mouse	Kinesthetic	F/T Sensors	Extended Dataset	Variation	Task Success
1 BBIP	✓	✗	✓	✓(3 demonstrations)	✓	53.3%
2 BBIP	✓	✗	✓	✓	✓	90.0%
3 BBIP	✓	✗	✓	✓	✗	93.3%
4 BBIP	✓	✗	✓	✗	✓	20.0%
5 BBIP	✓	✗	✓	✗	✗	90.0%
6 BBIP	✓	✗	✗	✓	✓	70.0%
7 BBIP	✓	✗	✗	✓	✗	100.0%
8 BBIP	✓	✗	✗	✗	✓	13.3%
9 BBIP	✓	✗	✗	✗	✗	100.0%
10 BBIP	✗	✓	✗	✓	✓	73.3%
11 BBIP	✗	✓	✗	✓	✗	100.0%
12 BBIP	✗	✓	✗	✗	✓	0.0%
13 BBIP	✗	✓	✗	✗	✗	100.0%
14 ProMP	✓	✗	✓	✗	✗	33.3%
15 ProMP	✓	✗	✓	✓	✓	20.0%
16 BC	✓	✗	✓	✓	✓	0.0%

Table 3.1: Bimanual Bayesian Interaction Primitives Performance on Peg-Insertion

are inserted, and one or more are stuck or not fully inserted, we consider the task as failed. Similarly, when the robots do not release the object at the end of the task, we consider the task failed even if all four grommets are inserted. Such a failure is usually caused when the phase estimation that predicts the task progress is incorrectly estimating that the task is not complete. Generally, we release the object with a simple threshold at 96% task completion, which empirically generated the best motions. It is important to note here that we did not choose 100% task completion before opening the grippers as the phase estimation might be slightly erroneous.

3.4.2 Model Evaluation

We compare our model, BBIP, against Probabilistic Movement Primitives (ProMP) (Paraschos *et al.*, 2013) as well as a simple Behavioral Cloning approach (BC) using a Bayesian neural network. Table 3.1 shows the results of the evaluated approaches based on which training modality has been used, either kinesthetic teaching or the space mouse. Note that in the case of using kinesthetic demonstrations, the force/torque sensor data will never be used. However, for the space mouse data, we evaluated the performance of the models with and without force/torque data. Furthermore, our standard evaluation starts with the bracket at a fixed pickup location, but we also test the performance when varying the initial pose of the bracket within a 3cm range along the x and y-direction. Column six indicates whether or not the starting position was variable, while column 5 indicates whether or not additional seven demonstrations (or three demonstrations when specified) data have been used to train the model to adapt to varying starting positions. Generally, all our models and baselines are trained on the same ten demonstrations, either collected by using the space mouse or kinesthetic teaching are being evaluated in 30 insertion attempts on the real robot.

While Table 3.1 also compares our approach against three baselines, it mainly evaluates the impact of having force/torque data available during training and the resulting adaptability to varying starting positions. Additionally, we also evaluate the model quality when using the two different training modalities. Line 2 shows the results of our main model, completing the insertion task with a 90% success rate when using a total of 17 demonstrations and varying starting positions. Using these seven additional training data is essential as a smaller number of data, as shown in

line 1, causes a drastic decrease in model performance. The efficacy against a model that is not trained on these additional data is shown in line 4, where the success rate is reduced to 20%. However, adding these additional training data does not have a catastrophic forgetting effect on the task in which we would always start at the same location as can be seen when comparing lines 3 and 2, resulting in a 3.33% increased performance when adding these additional variations.

The main difference between the different training modalities is the availability of the force/torque sensor data. Lines 6 to 9 show the same experiments as lines 2 to 5 discussed in the previous section, but with the important difference that the experiments in lines 6 to 9 do not use the force/torque data. Lines 7 and 9 show a 100% success rate when the starting position is not varied. This is a 10% over the same situations in lines 3 and 5, indicating that the force/torque data might not be beneficial to the model's success. However, when always using the same starting position, the model only needs to learn a single motion that is roughly the same for each evaluation run. Yet, when varying the starting positions, the model's success rate drastically decreases (line 6 and 8), even when training it on the varying starting positions (line 6). Besides the used object tracking, the force/torque sensor data are the only way the model perceives its environment. When not perfectly inserting the bracket on the first attempt, the additional force/torque sensors allows the model to dynamically recover from such situations, as seen in the comparison of lines 2 and 6, resulting in a 20% performance increase.

In the previous paragraph, the impact of force/torque sensor data has been evaluated on the exact same demonstrations, just by omitting the additional data. However, we would now expect to see similar performance in the data collected from the kinesthetic approach, which by nature doesn't have usable force/torque sensor data.

Lines 10 to 13 show the same experiments as lines 6 to 9, but without the different teaching modality. We can see comparable performance between the models trained on the different modalities. However, as discussed earlier, force/torque sensor data are important for error recovery.

Lines 14 to 16 compare our model with three different baselines. In order to give the baselines the best chance, we provide them with the data collected from the space mouse with force/torque data and evaluate it on the standard and varied starting position. However, all baselines show a drastically decreased performance as compared to our model in line 2, thus underlining the capabilities of our approach to adapt to temporal and spatial requirements during task execution dynamically. The next chapter will have a more detailed look at our model's ability to adapt to temporal and spatial disturbances dynamically.

3.4.3 Spatial Conditioning

In the previous section, we discussed the overall performance of our model; however, a key contribution is our model's ability to condition on the current situation and generate an appropriate trajectory that completes the task while also adjusting temporally to potential disturbances in the motion. First, Figure 3.4 and Figure 3.5 show our model's ability to condition spatially when certain sensor modalities change. Figure 3.4 shows the influence of the force/torque sensors on the bracket movement when additional forces are sensed. These additional forces could, for example, be added when being stuck on the pins. The figure shows 200 trajectories of the bracket that have been executed without disturbances and shows the general motion when the bracket is about to be inserted on the pins. Generally, the trajectories vary by about 5mm; however, when adding disturbances in the force sensor readings, the tra-

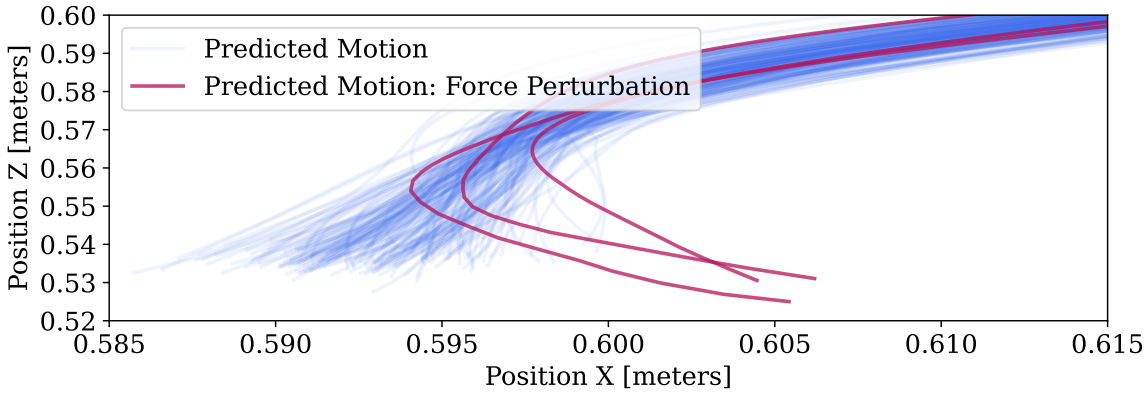

Figure 3.4: Motion Adaption from Force/Torque Sensors

Generalization capabilities of our IntPrim model. Conditioned on the initial object pose, the model predicts similar trajectories over consecutive runs (blue), however, introducing constant disturbances to the F/T sensors results in adjusted trajectories (red) that still follow the general trend, however, will not be able to complete the task.

jectories take drastic turns. These turns are shown in red and are solely caused by the increased force readings. Furthermore, Figure 3.5 shows a similar experiment; however, here, control discrepancies have been introduced, and the model is conditioned on its motion on these changed behaviors.

In both cases, disturbances have been introduced as a constant factor on the force-torque sensors or absolute joint values, respectively. In order to reflect the influence of the disturbed sensors on the remaining system, predictions of the interaction primitives have been carried over to the next time step. While the shown insertion motion is not successful in either case, it is clearly visible that our model is sensitive to changes in the received sensor data and can continue to follow the general trend of the demonstrated behaviors without causing extreme deviations from the expected behaviors, allowing small disturbances to yield successful task completions still.

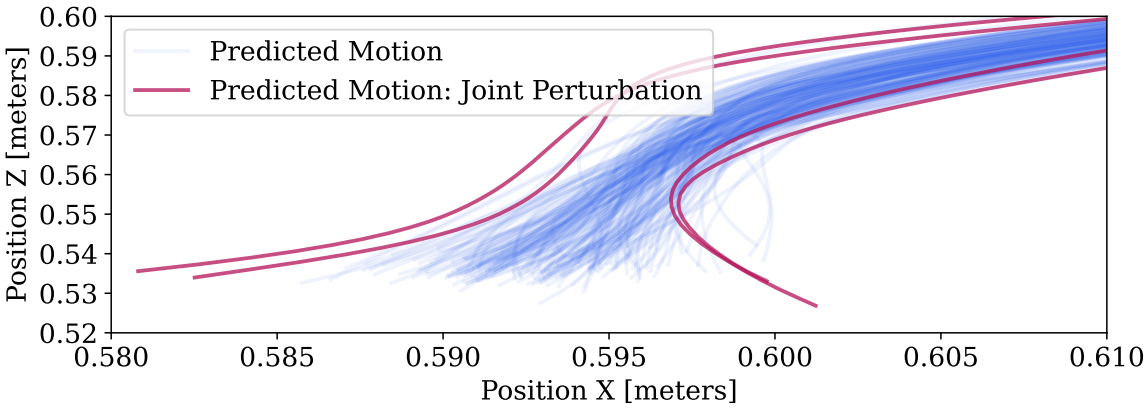

Figure 3.5: Motion Adaption from Joint Position Sensors

Generalization capabilities of our IntPrim model. Conditioned on the initial object pose, the model predicts similar trajectories over consecutive runs (blue); however, introducing constant disturbances to the joints results in adjusted trajectories (red) that still follow the general trend, however, will not be able to complete the task.

3.4.4 Temporal Conditioning

Besides disturbances in space, as discussed in the previous section, disturbances in time are also a common occurrence during task execution. Time disturbances are, for example, common when the bracket is about to be inserted into the four pegs but is slightly off, causing it to be stuck on the pegs. In such a case, the task is not progressing and needs a spacial adjustment before being able to be completed. In such a case, the estimated phase progression should stop and only continue when the motion is resumed. Figure 3.6 shows this behavior in the blue line where the motion has been stopped after 60% of the task completion. In that case, the model is able to adapt to this situation by stopping the estimated task progression until the spatial correction was able to move the bracket into a position that allows further motion. A more extreme case of this issue is if a human is in the loop and pushes

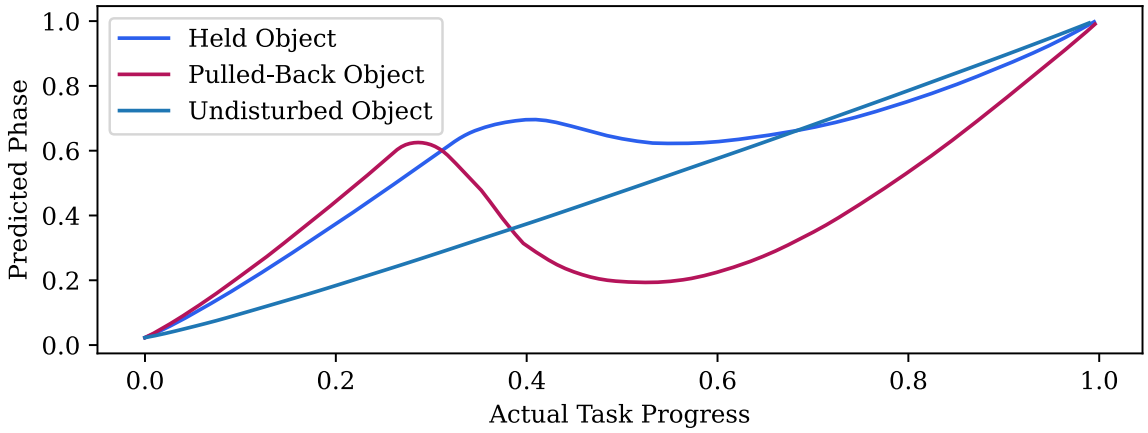

Figure 3.6: Temporal (Phase) Adaption to Various Motion Disturbances
Phase progression in three scenarios: Held object (blue), showing how the phase stops and continues, pulled-back object (red) where the object is pulled back to the start after completing 60% of the task, and an undisturbed execution.

the robot in orthogonal or opposite directions of the intended task. In such cases, the system should recognize the disturbance and not only stop the estimated task progression but also apply a negative phase velocity to revert the task to a previous state. Such a negative velocity is shown in the red line of Figure 3.6 in which the object was pulled back towards the start of the interaction. In each case, it is visible that our model is able to address these situations by either stopping or reversing the previously predicted task progress.

3.4.5 NASA TLX Workload Evaluation

Lastly, we evaluate how easy it is to provide the demonstrations used in our experiment using the different teaching modalities. The results in Table 3.1 indicates that using the space mouse data is better for the learning algorithm as using the force/torque data has a non-negligible benefit when handling variations in the ob-

ject starting pose. However, collecting the data requires a different amount of effort. Since evaluating the required workload is a subjective measure, we utilize the NASA Task Load Index Hart (2006); Hart and Staveland (1988). NASA TLX is a subjective assessment of the workload a user is experiencing when completing a task, where workload represents the cost of the labor. To assess the workload of each training modality, additional demonstrations have been collected from eight users that are familiar with robots; however, they did not interact with a biannual setup before. Each demonstrator is providing three demonstrations with each of the two teaching modalities, which are either using a space mouse to control the six degrees of freedom of the carried bracket or manually moving the object in a kinesthetic demonstration. In both cases, the participants are not instructed on how the bracket should be inserted, besides being shown the final configuration prior to providing the first demonstration. Additionally, we also allowed the participants to familiarize themselves with the input modalities by allowing an unrecorded first demonstration attempt.

After providing a set of three demonstrations with one of the modalities, users are asked to complete a brief survey that collects their subjective assessment in six different categories related to workload:

- **Mental Demand**: The Mental demand of the task.
- **Physical Demand**: The physical demand of the task.
- **Temporal Demand**: How rushed a participant felt.
- **performance**: How successful the participant was.
- **Effort**: Evaluating how hard participants had to work to achieve the performance they achieved.

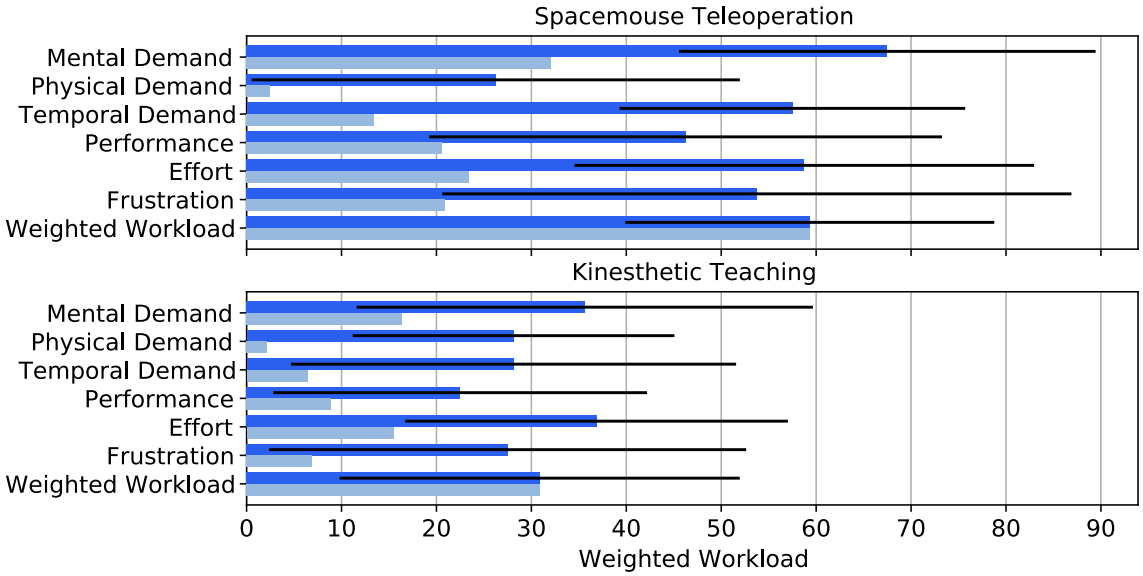

Figure 3.7: NASA TLX Workload Assessment

Evaluation of the NASA TLX workload index. Blue bars show the raw data and orange bars the weighted averages.

- **frustration**: How insecure, discouraged, irritated, stressed or annoyed a participant felt during the task.

However, since every user has a different subjective definition of what workload means to them, weights for each category are derived from simple decisions about which member of each paired combination of the six metrics are more important to them. Utilizing this one-time weighting approach further increases the sensitivity of the metrics across multiple subjects. Figure 3.7 provides the results of the human subject study in each category shown in blue, as well as the overall weighted workload shown in light-blue.

Given the assessment of the workload, the eight participants indicate that the overall workload of providing a demonstration with the space mouse is approximately twice as high as when using kinesthetic teaching. Particularly, the temporal demand,

perceived performance, and frustration are significantly higher compared to the kinesthetic teaching mode. This is also reflected in the length of the collected trajectories. While the average kinesthetic demonstration length from a non-expert is 173 and 114 time-steps from an expert, the average trajectory lengths for a space mouse demonstration is 587 and 292 from a non-expert and expert, respectively. Given the results of this human subject study, collecting data by using the kinesthetic teaching mode is significantly easier from a human perspective; however, the model's ability to adapt to variations by using the force/torque sensors from the space mouse teaching mode has significant benefits. The decision on which mode should be preferred is an open discussion topic. When it can be expected that the object variations are minor, using kinesthetic demonstrations should be preferred; however, when variability is an unavoidable aspect of the task, data from the space mouse should be preferred.

3.5 Conclusion

In this paper, we introduced a framework for extracting compliant bimanual policies from human demonstrations. Our approach combines the strength of probabilistic inference with underlying admittance control strategies in order to enable contact-rich interactions with the environment. We show that spatio-temporal inference can play a major role in determining safe, reliable, and efficient control signals for physical interaction with objects and humans. We further applied our approach to a bimanual insertion task that requires the joint insertion of multiple grummets into their corresponding pegs. Using the introduced methodology, we achieve a success rate of 90%. Further, we have conducted a user study to identify optimal data collections interfaces. Our study shows that participants largely preferred kinesthetic teaching to teleoperation with a space mouse. However, we show in this chapter that the

kinesthetic teaching approach renders the recorded force-torque readings unusable. This results in a tradeoff between higher reproduction accuracy (space mouse) and higher usability (kinesthetic teaching).

Chapter 4

LANGUAGE-CONDITIONED IMITATION LEARNING

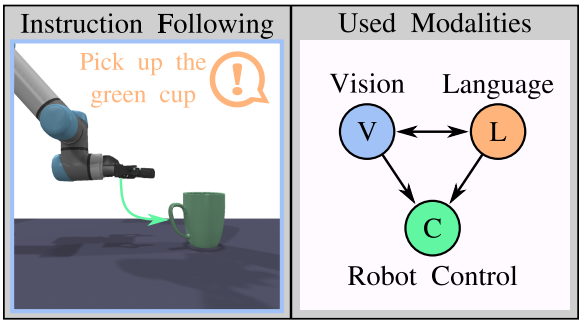

Figure 4.1: Instruction Following Modalities

Overview of the used modalities in the instruction following task.

Imitation learning is a popular approach for teaching motor skills to robots. However, most approaches focus on extracting policy parameters from execution traces alone (i.e., motion trajectories and perceptual data). In the previous chapters, no adequate communication channel existed between the human expert and the robot to describe critical aspects of the task, such as the properties of the target object or the intended shape of the motion. Motivated by insights into the human teaching process, we introduce a method for incorporating unstructured natural language into imitation learning. At training time, the expert can provide demonstrations along with verbal descriptions in order to describe the underlying intent (e.g., "go to the large green bowl"). The training process then interrelates these two modalities to encode the correlations between language, perception, and motion. Figure 4.1 shows an overview of the instruction following task and the relation between the used modalities. After training, the resulting language-conditioned visuomotor policies can be

conditioned at runtime on new human commands and instructions, which allows for more fine-grained control over the trained policies while also reducing situational ambiguity. We demonstrate in a set of simulation experiments how our approach can learn language-conditioned manipulation policies for a seven-degree-of-freedom robot arm and compare the results to a variety of alternative methods.

The work presented in this chapter has been published and presented at the Conference on Neural Information Processing Systems (NeurIPS) 2020 and was selected for a spotlight presentation.

4.1 Introduction

Learning robot control policies by imitation (Schaal, 1999) is an appealing approach to skill acquisition and has been successfully applied to several tasks, including locomotion, grasping, and even table tennis (Chalodhorn *et al.*, 2007a; Amor *et al.*, 2012; Mülling *et al.*, 2013). In this paradigm, expert demonstrations of robot motion are first recorded via kinesthetic teaching, teleoperation, or other input modalities. These demonstrations are then used to derive a control policy that generalizes the observed behavior to a larger set of scenarios that allow for responses to perceptual stimuli (e.g., joint angles and an RGBD camera image of the work environment) with appropriate actions (e.g., moving a table-tennis paddle to hit an incoming ball).

In goal-conditioned tasks, perceptual inputs alone may be insufficient to dictate optimal actions (Codevilla *et al.*, 2018) (e.g., without a target object, what should a picking robot retrieve from a bin when activated?). Consequently, expert demonstration and control policies must also be conditioned on a representation of the goal. While we use the term *goals*, it may refer to end goals (e.g., target objects) or constraints on motion (e.g., minimizing end-point effector acceleration) (Ding

et al., 2019). Prior work has typically employed manually designed goal specifications (e.g., vectors indicating a target position, a one-hot vector indicating target objects, or a single value indicating the execution speed). However, this is an inflexible approach that must be pre-defined before training and cannot be modified after deployment.

In the present work, we consider language as a flexible goal specification for imitation learning in manipulation tasks. As shown in Fig. 4.2(center), we consider a seven-degree-of-freedom robot manipulator anchored to a flat workspace populated with a set of objects that vary in shape, size, and color. The agent is instructed by a user to manipulate these objects in language for picking (e.g., "grab the blue cup") and pouring tasks (e.g., "pour some of its contents into the small red bowl"). In order to succeed, the agent must relate these instructions to the objects in the environment, as well as constraints on how they are manipulated (e.g., pouring some or all of something require different motions). We examine the role of imitation learning from demonstrations in this setting that consist of developing a training set of instructions and associated robot motion trajectories.

We developed an end-to-end model for the language-conditioned control of an articulated robotic arm – mapping directly from observation pixels and language-specified goals to motor control. We conceptually divided our architecture into two modules: a high-level semantic network that encodes goals and the world state and a lower-level controller network that uses the higher encoding to generate suitable control policies. Our high-level semantic network must relate language-specified goals, visual observation of the work environment, and the robot's current joint positions into a single encoding. To do this, we leveraged advances in attention mechanisms from vision-and-language research Anderson *et al.* (2018a) to associate instructions

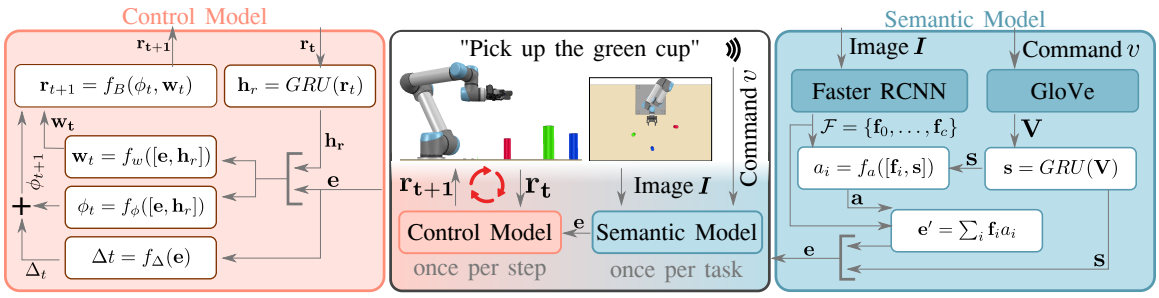

Figure 4.2: Language-Conditioned Imitation Learning System Overview

Overview of the general system architecture. (Left) Details of the controller model, which synthesizes robot control signals . (Right) details of the semantic model, which extracts critical information about the task from both perceptual input and language commands. Dark-blue boxes indicate pre-trained components of our model.

and target objects. Our low-level controller synthesizes parameters of a motor primitive that specifies the entire future motion trajectory, providing insight into the predicted future behavior of the robot from the current observation. The model was trained end-to-end to reproduce demonstrated behavior while minimizing a set of auxiliary losses to guide the intermediate outputs.

We evaluated our model in a dynamic-enabled simulator with random assortments of objects and procedurally generated instructions, with success in 84% of sequential tasks that required picking up a cup and pouring its contents into another vessel. This result significantly outperformed state-of-the-art baselines. We provided detailed ablations of modeling decisions and auxiliary losses, as well as detailed analysis of our model's generalization to combinations of modifiers (color, shape, size, and pour-quantity specifiers). We also assessed robustness to visual and physical perturbations in the environments. While our model was trained on synthetic language, we also ran human-user experiments with free-form natural-language instructions for

picking/pouring tasks, with a success rate of 64% for these instructions.

All data used in this chapter, along with a trained model and the full source code can be found at: `https://github.com/ir-lab/LanguagePolicies`. The release features a number of videos and examples on how to train and validate language-conditioned control policies in a physics-based simulation environment. Additionally, detailed information about the experimental setup and the human data collection process can be found under the link above.

Contributions. To summarize our contributions, we

1. introduced a language-conditioned manipulation task setting in a dynamically accurate simulator,

2. provided a natural-language interface which allows laymen users to provide robot task specifications in an intuitive fashion,

3. developed an end-to-end, language-conditioned control policy for manipulation tasks composed of a high-level semantic module and low-level controller, integrating language, vision, and control within a single framework,

4. demonstrated that our model, trained with imitation learning, achieved a high success rate on both synthetic instructions and unstructured human instructions.

4.2 Background

Imitation learning (IL) provides an easy and engaging way to teach new skills to an agent. Instead of programming, the human can provide a set of demonstrations (Argall *et al.*, 2009) that are turned into functional (Ijspeert *et al.*, 2013) or probabilistic (Maeda *et al.*, 2014; Campbell *et al.*, 2019) representations. However, a

limitation of this approach is that the state representation must be carefully designed to ensure that all necessary information for adaptation is available. Furthermore, it is assumed that either a sufficiently large task taxonomy or set of motion primitives is already available (i.e., semantics and motions are not trained in conjunction). Neural approaches scale imitation learning (Pomerleau, 1989; Anderson *et al.*, 2019; Kuo *et al.*, 2020; Abolghasemi *et al.*, 2019; Chang *et al.*, 2021) to high-dimensional spaces by enabling agents to learn task-specific feature representations. However, both foundational references (Pomerleau, 1989), as well as more recent literature (Codevilla *et al.*, 2018), have noted that these methods lack "a communication channel," which would allow the user to provide further information about the intended task, at nearly no additional cost Cui *et al.* (2020). Hence, both the designer (programmer) and the user have to resort to numerical approaches for defining goals. For example, a one-hot vector may indicate which of the objects on the table is to be grasped. This limitation results in an unintuitive and potentially hard-to-interpret communication channel that may not be expressive enough to capture user intent regarding *which* object to act upon or *how* to perform the task. Another popular methodology for providing such semantic information is to use formal specification languages such as temporal logic Kress-Gazit *et al.* (2009); Raman *et al.* (2012). Such formal frameworks are compelling, since they support the formal verification of provided commands. Even for experts, specifying instructions in these languages can be a challenging, complicated, and time-consuming endeavor. An interesting compromise was proposed in Gopalan *et al.* (2018), where natural-language specifications were first translated to temporal logic via a deep neural network. However, such a methodology limits the range of descriptions that can be provided due to the larger expressivity of the English language relative to the formal specification language.

DeepRRT, presented in (Kuo *et al.*, 2020), describes a path-planning algorithm that uses natural-language commands to steer search processes, and (Sugita and Tani, 2005) introduced the use of language commands for low-level robot control. A survey of natural language for robotic task specification can be found in Matuszek (2018). Beyond robotics, the combination of vision and language has received ample attention in visual question-and-answering systems (VQA) (Lu *et al.*, 2019; Antol *et al.*, 2015) and vision-and-language navigation (VNL) (Zhu *et al.*, 2020; Krantz *et al.*, 2020; Codevilla *et al.*, 2018). Our approach is most similar to (Abolghasemi *et al.*, 2019). However, unlike our model, the work in (Abolghasemi *et al.*, 2019) used a fixed alphabet and required information about the task to be extracted from the sentence before being used for control. In contrast, our model can extract a variety of information directly from natural language.

4.3 Problem formulation and approach

We considered the problem of learning a policy π from a given set of demonstrations $\mathcal{D} = \{\mathbf{d}^0, .., \mathbf{d}^m\}$, where each demonstration contained the desired trajectory given by robot states $\boldsymbol{R} \in \mathbb{R}^{T \times N}$ over T time steps and with N control variables (Subsequently, we would assume, without loss of generality, a seven-degree-of-freedom (DOF) robot – i.e., N = 7.). We also assumed that each demonstration contained perceptual data $\boldsymbol{I}$ of the agent's surroundings and a task description v in natural language. Given these data sources, our overall objective was to learn a policy $\pi(v, \boldsymbol{I})$, which imitated the demonstrated behavior in $\mathcal{D}$ while considering the semantics of the natural-language instructions and critical visual features of each demonstration. After training, we provided the policy with a different, new state for the agent's environment, given as image $\boldsymbol{I}$, and a new task description (instruction) v. In turn, the

policy generated control signals that were needed to achieve the objective described in the task description. We did not assume any manual separation or segmentation into different tasks or behaviors. Instead, the model was assumed to independently learn such a distinction form the provided natural-language description. Fig. 4.2 shows an overview of our proposed method. At a high level, our model takes an image $\boldsymbol{I}$ and task description v as input to create a task embedding $\boldsymbol{e}$ in the semantic model. Subsequently, this embedding is used in the control model to generate robot actions at each time in a closed-loop fashion.

4.3.1 Preprocessing vision and language

We first preprocessed both the input image and verbal description by building upon existing frameworks for image processing and language embedding. More specifically, we used a pre-trained object detection network on image $\boldsymbol{I} \in \mathbb{R}^{569\times320\times3}$ of the robot's environment that identified salient image regions of any object found in the robot's immediate workspace. In our approach, we used Faster R-CNN (Ren *et al.*, 2015) to identify a set of candidate objects $\mathcal{F} = \{\boldsymbol{f}_0, .., \boldsymbol{f}_c\}$, each represented by a feature vector $\boldsymbol{f} = [f^o, \boldsymbol{f}^b]$ composed of the detected class f^o, as well as their bounding boxes $\boldsymbol{f}^b \in \mathbb{R}^4$, within the workspace of the robot, ordered by the detection confidence f^c of each class. Based on a pre-trained FRCNN model trained from ResNet-101 on the COCO dataset, we continued to fine-tune the model for our specific use-case on 40 thousand arbitrarily generated environments from our simulator. After fine-tuning, the certainty of FRCNN on our objects was above 98%.

Regarding the preparation of the language input, each verbal description v was split into individual words and converted into a row index of a matrix $\boldsymbol{G} \in \mathbb{R}^{30000\times50}$, representing the 30 thousand most-used English words, initialized and frozen to utilize

pre-trained GloVe word embeddings (Pennington *et al.*, 2014). Our model took the vector of row-indices as input, and the conversion to their respective 50-dimensional word embedding was done within our model to allow further flexibility for potentially untrained words.

4.3.2 Semantic model

The goal of the semantic model is to identify relevant objects described in the natural-language command, given a set of candidate objects. In order to capture the information represented in the natural-language command, we first converted the task description v into a fixed-size matrix $\boldsymbol{V} \in \mathbb{R}^{15\times 50}$, encoding up to 15 words with their respective 50-dimensional word embedding. Based on $\boldsymbol{V}$, a sentence embedding $\boldsymbol{s} \in \mathbb{R}^{32}$ was generated with a single GRU cell $\boldsymbol{s} = \mathrm{GRU}(\boldsymbol{V})$.

To identify the object referred to by the natural-language command v from the set of candidate regions $\mathcal{F}$, we calculated a likelihood for each region given the sentence embedding $\boldsymbol{s}$ (Anderson *et al.*, 2018a). The likelihood $a_i = \boldsymbol{w}_a^T f_a([\boldsymbol{f}_i, \boldsymbol{s}])$ was calculated by concatenating the sentence embedding $\boldsymbol{s}$ with each candidate object $\boldsymbol{f}_i$, applying the attention network $f_a : \mathbb{R}^{37} \to \mathbb{R}^{64}$ and converting the result into a scalar by multiplying it with a trainable weight $\boldsymbol{w}_a \in \mathbb{R}^{64}$. The function $f_a(\boldsymbol{x}) = \tanh(\boldsymbol{W}\boldsymbol{x} + \boldsymbol{b}) \odot \sigma(\boldsymbol{W}'\boldsymbol{x} + \boldsymbol{b}')$ is a nonlinear transformation that used a gated hyperbolic tangent activation (Dauphin *et al.*, 2016), where $\odot$ represents the Hadamard product of the elements, and $\boldsymbol{W}, \boldsymbol{W}', \boldsymbol{b}, \boldsymbol{b}'$ are trainable weights and biases, respectively. This operation was repeated for all c candidate regions, and the individual likelihoods a_i were used to form a probability distribution over candidate objects $\boldsymbol{a} = \mathrm{softmax}([a_0, ..., a_c])$. Then, the language-conditioned task representation was the mean $\boldsymbol{e}' = \sum_{i=0}^{c} \boldsymbol{f}_i \boldsymbol{a}_i$ where $\boldsymbol{e}' \in \mathbb{R}^5$. The final task representation $\boldsymbol{e} \in \mathbb{R}^{32}$

was computed by reintroducing the sentence embedding $\boldsymbol{s}$, which was needed in the low-level controller to determine task modifiers like *everything* or *some*, and concatenating it with $\boldsymbol{e}'$. The task embedding was then created with a single fully connected layer $\boldsymbol{e} = \text{ReLU}(\boldsymbol{W}[\boldsymbol{e}', \boldsymbol{s}] + \boldsymbol{b})$, where $\boldsymbol{W}$ and $\boldsymbol{b}$ were trainable variables.

4.3.3 Control model

The generation of controls is a function that maps a task embedding $\boldsymbol{e}$ and the current agent state $\boldsymbol{r}_t$ to control values for future time steps. Control signal generation is performed in two steps. In the first step, the control model produces the parameters that fully specify a motor primitive. A motor primitive in this context is a trajectory of the control signals for all the robot's degrees of freedom and can be executed in an open-loop fashion until task completion. However, to account for the nondeterministic nature of control tasks (e.g., physical perturbations, sensor noise, execution noise, force exchange, etc.) we employed a closed-loop approach by recalculating the motor primitive parameters at each time step.

Motor primitive generation We used motor primitives inspired by the approach in Ijspeert *et al.* (2013). A motor primitive was parameterized by $\boldsymbol{w} \in \mathbb{R}^{1\times(B*7)}$, where B is the number of kernels for each DOF of the robot. These parameters specified the weights for a set of radial basis function (RBF) kernels, which would be used to synthesize control signal trajectories in space. In addition, the motor primitive generation step also estimated the current (temporal) progress towards the goal as phase variable $0 \leq \phi \leq 1$ and the desired phase progression Δ_ϕ. A phase variable of 0 means that the behavior has not yet started, while a value of 1 indicates a completed execution. Predicting the phase and phase progression allowed our model

to dynamically update the speed at which the policy was evaluated. In order to keep track of the robot's current and previous movements, we used a GRU cell that was initialized with the start configuration $\boldsymbol{r}_0$ of the robot and encoded all subsequent robot states $\boldsymbol{r}_t$ at each step t of the control loop into a latent robot state $\boldsymbol{h}_t \in \mathbb{R}^7$. Based on the task encoded in the latent task embedding $\boldsymbol{e}$ and the latent state of the robot $\boldsymbol{h}_t$ at time step t, the model generated the full set of motor primitive parameters for the current time step $(\boldsymbol{w}_t, \phi_t, \Delta_\phi) = (f_{\boldsymbol{w}}([\boldsymbol{h}_t, \boldsymbol{e}]), f_\phi([\boldsymbol{h}_t, \boldsymbol{e}]), f_\Delta(\boldsymbol{e}))$, where $f_\phi : \mathbb{R}^{39} \to \mathbb{R}^1$, $f_{\boldsymbol{w}} : \mathbb{R}^{39} \to \mathbb{R}^{B*7}$ and $f_\Delta : \mathbb{R}^{32} \to \mathbb{R}^1$ are multilayer perceptrons. Finally, the generated parameters were used to synthesize and execute robot control signals, as described in the next section.

Motor primitive execution A motor primitive parameterization $(\boldsymbol{w}_t, \phi_t, \Delta_\phi)$ encoded the full trajectory for all robot DOF. To generate the next control signals $\boldsymbol{r}_{t+1}$ to be executed, we evaluated the motor primitive at phase $\phi_t + \Delta_\phi$. Each motor primitive was a weighted combination of radial basis function (RBF) kernels positioned equidistantly between phases 0 and 1. Each kernel was characterized by its location μ with a fixed scale σ: $\Phi_{\mu,\sigma}(x) = \exp\left(-\left((x-\mu)^2 / (2\sigma)\right)\right)$. All B kernels (We use $B = 11$ RBF kernels for each of the 7 DOF with a scale of $\sigma = 0.012$) for a single DOF were represented as a basis function vector $[\Phi\mu_0, \sigma(\phi), \ldots, \Phi\mu_B, \sigma(\phi)] \in \mathbb{R}^{B\times 1}$, and each kernel was a function of ϕ, representing the relative temporal progress towards the goal. Given that a linear combination of RBF kernels approximated the desired control trajectory, we could define a sparse linear map $\boldsymbol{H}_{\phi_t} \in \mathbb{R}^{7\times(B*7)}$, which contained the basis function vectors for each DOF along the diagonals. The control signal at time $t+1$ was given as $\boldsymbol{r}_{t+1} = f_B(\phi_t + \Delta_\phi, \boldsymbol{w}_t) = \boldsymbol{H}_{\phi_t+\Delta_\phi}\boldsymbol{w}_t^T$, which allowed us to quickly calculate the target joint configuration at a desired phase $\phi_t + \Delta_\phi$ in

a single multiplication operation. The respective parameters were generated in the previously described motor primitive generation step. The control model worked in a closed loop, taking potential discrepancies (and perturbations) between the desired robot motion and the actual motion of the robot into consideration. Based on the past motion history of the robot, our model was able to identify its progress within the desired task by utilizing phase estimation. This phase estimation was a unique feature in our controllers and differed from previous approaches with a fixed phase progression Ijspeert *et al.* (2013).

4.3.4 Model integration

The components described in the previous sections were combined sequentially to create our final model. After preprocessing the input command into a sequence of word IDs for GloVe and detecting object locations in the robot's immediate surrounding using FRCNN, the semantic model (section 4.3.2) created a task-specific embedding $\boldsymbol{e}$ that encoded all the necessary information about the desired action. Subsequently, the control model translated the latent task embedding $\boldsymbol{e}$ and current robot state $\boldsymbol{r}_t$ at each time step t into hyper-parameters for a motor primitive (section 4.3.3). These parameters were defined as the weights $\boldsymbol{w}_t$, phase ϕ_t, and phase progression Δ_ϕ at time t. By using these parameters, the motor primitive was used to infer the next robot configuration $\boldsymbol{r}_{t+1}$, as well as the entire trajectory $\boldsymbol{R} = \{\boldsymbol{r}_0, ..., \boldsymbol{r}_T\}$, allowing for subsequent motion analysis. At each time step, a new motor primitive was defined by generating a new set of hyper-parameters from the task representation $\boldsymbol{e}$. While $\boldsymbol{e}$ was constant over the duration of an interaction, the current robot state $\boldsymbol{r}_t$ was used at each time step to update the motor primitive's parameters. An overview of the architecture can be seen in figure 4.2.

The integration of our model resulted in an end-to-end approach that takes high-level features and directly converts them to low-level robot control parameters. As opposed to a multi-staged approach, which requires a significant amount of additional feature engineering, our framework learned how language affects the behavior (type, goal position, velocity, etc.) automatically, while also learning the control itself. Another advantage of this end-to-end approach was that the overall system could be trained such that the individual components harmonized. This was particularly important for the interplay of language embedding and control when using language as a modifier for trajectory behaviors.

4.3.5 End-to-end training

Our model was trained in an end-to-end fashion, utilizing five auxiliary losses to aid the training process. The overall loss was a weighted sum of five auxiliary losses: $\mathcal{L} = \alpha_a \mathcal{L}_a + \alpha_t \mathcal{L}_t + \alpha_\phi \mathcal{L}_\phi + \alpha_w \mathcal{L}_w + \alpha_\Delta \mathcal{L}_\Delta$. The guided attention loss $\mathcal{L}_a = -\sum_i^C x_i \log(y_i)$ trained the attention model and was defined as the cross-entropy loss for a multi-class classification problem over c classes, where $\boldsymbol{a}^\star \in \mathbb{R}^c$ are the ground truth labels and $\boldsymbol{a} \in \mathbb{R}^c$ the predicted classes. The training label $\boldsymbol{a}^\star$ was a one-hot vector created alongside the image preprocessing. It indicated which object is referred to by the task description, depending on the order of candidate objects in $\mathcal{F}$. The controller was guided by four mean-squared-error losses, starting with the phase estimation $\mathcal{L}_\phi = \text{MSE}(\phi_t, \phi_t^\star)$ and with the phase progression, defined as $\mathcal{L}_\Delta = \text{MSE}(\Delta_\phi, \Delta_\phi^\star)$, indicating where the robot was in its current behavior and how much the current configuration would be updated for the next time steps. Both of the labels $\Delta_\phi^\star$ and $\phi_t^\star$ could easily be inferred from the number of steps in the given demonstration. Furthermore, we minimized the difference between two consecutive

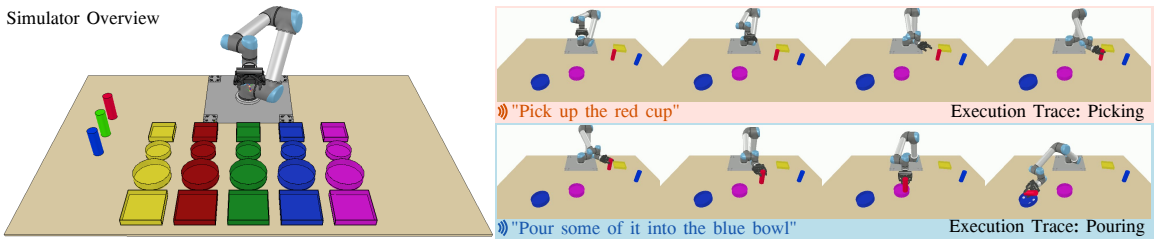

Figure 4.3: Instruction Following: Scene and Examples

Overview of the available objects in simulation (left) and sample task execution sequences with their respective commands of the two tasks: picking (top right) and pouring (bottom right).

basis weights with $\mathcal{L}_w = \text{MSE}(\boldsymbol{W}_t, \boldsymbol{W}_{t+1})$. By minimizing this loss, the model was ultimately able to predict full motion trajectories at each time step, since significant updates between consecutive steps were mitigated. Finally, the overall error of the generated trajectory $\boldsymbol{R} = [\boldsymbol{r}_{\phi=0}, ..., \boldsymbol{r}_{\phi=1}]$ was calculated via $\mathcal{L}_t = \text{MSE}(\boldsymbol{R}, \boldsymbol{R}^\star)$ against the demonstrated trajectory $\boldsymbol{R}^\star$. Values $\alpha_a = 1$, $\alpha_t = 5$, $\alpha_\phi = 1$, $\alpha_w = 50$, $\alpha_\Delta = 14$ were empirically chosen as hyper-parameters for $\mathcal{L}$ that had been found by a grid-search approach. We trained our model in a supervised fashion by minimizing $\mathcal{L}$ with an Adam optimizer using a learning rate of 0.0001.

4.4 Evaluation and results

We evaluated our approach in a simulated robot task with a table-top setup. In this task, a seven-DOF robot manipulator had to be taught by an expert how to perform a combination of picking and pouring behaviors. At training time, the expert provided both a kinesthetic demonstration of the task and a verbal description (e.g.,"pour a little into the red bowl"). The table might feature several differently shaped, sized, and colored objects, which often led to ambiguities in natural-language

descriptions thereof. The robot had to learn how to efficiently extract critical information from the available raw-data sources in order to determine *what* to do, *how* to do it, or *where* to go. We show that our approach leveraged perception, language, and motion modalities to generalize the demonstrated behavior to new user commands or experimental setups.

The evaluation was performed using CoppeliaSim (Rohmer *et al.*, 2013; James *et al.*, 2019), which allowed for accurate dynamics simulations at an update rate of 20Hz. Fig. 4.3 depicts the table-top setup and the different variations of the objects used. We utilized three differently colored cups containing a granular material that could be poured into the bowls. Additionally, we used 20 variations of bowls in two sizes (large and small), two shape types (round and squared), as well as five colors (red, green, blue, yellow, and pink). When generating an environment, we randomly placed a subset of the objects on the table, with a constraint to prevent collisions or other artifacts. A successful picking action was achieved when a grasped object could be stably lifted from the table. Successful pouring was detected whenever the cup's dispersed content ended up in the correct bowl. Tasks of various difficulties could be created by placing multiple objects with overlapping properties on the table.

To generate training and test data, we asked five human experts to provide templates for verbal task descriptions. Annotators watched prerecorded videos of a robot executing either of the two tasks (picking or pouring) and were asked to issue a command that they thought the robot was executing. During annotation, participants were encouraged to use free-form natural language and not adhere to a predefined language pattern. The participants in our data collection were graduate students familiar with robotics but not familiar with the goal of the present research. Overall, we collected 200 task explanations from five annotators where each participant

labeled 20 picking and 20 pouring actions. These 200 task descriptions were then manually templated to create replaceable noun phrases and adverbs, as well as basic sentence structures. To train our model, task descriptions for the training examples were then automatically generated from the set of sentence templates and synonyms from which multiple sentences could be extracted via synonym replacement. In order to generate natural task descriptions, we first identified the minimal set of visual features required to uniquely identify the target object, breaking ties randomly. The set of required features was dependent on which objects were in the scene – e.g., if only one red object existed, a viable description that uniquely describes the object in the given scene could refer only to the target's color; however, if multiple red objects were present, other or further descriptors might be necessary. Synonyms for objects, visual feature descriptors, and verbs were chosen at random and applied to a randomly chosen template sentence in order to generate a possible task description. Given the sent of synonyms and templates, our language generator could create 99,864 unique task descriptions of which we randomly used 45,000 to generate our data set. The final data set contained 22,500 complete task demonstrations composed of the two subtasks (grasping and pouring), resulting in 45,000 training samples. Of these samples, we used 4,000 for validation and 1,000 for testing, leaving 40,000 for training.

Basic metrics: Table 4.1 summarizes the results of testing our model on a set of 100 novel environments. Our model's overall task success describes the percentage of cases in which the cup was first lifted and then successfully poured into the correct bowl. This sequence of steps was successfully executed in 84% of the new environments. Picking alone achieved a 98% success rate, while pouring resulted in 85%. We argue that the drop in performance was due to increased linguistic variability when describing the pouring behavior. These results indicate that the model appropriately

Table 4.1: Model Ablations Regarding Losses, Structure, and Training Size

	Our model					Tasks Success			Execution statistics					Error statistics (pouring)							
	Att	Δ_ϕ	ϕ	$\boldsymbol{W}$	Trj	Pick	Pour	Seq	Dtc	PIn	QDif	MAE	Dst	None	C	S	F	C+S	C+F	S+F	C+S+F
17					✓	0.57	0.53	0.28	0.83	0.61	0.79	0.15	9.33	0.83	0.36	0.69	**1.00**	0.31	0.00	0.90	0.56
18		✓	✓	✓	✓	0.00	0.44	0.00	0.67	0.57	0.74	0.17	20.78	**1.00**	0.33	0.62	0.50	0.27	0.00	0.80	0.3
19	✓				✓	0.62	0.84	0.51	0.97	0.89	0.94	0.12	4.16	0.83	**0.89**	0.85	**1.00**	0.82	0.67	0.80	0.67
20	FF attention					0.00	0.01	0.00	0.41	0.14	0.60	0.22	25.63	0.00	0.00	0.00	0.00	0.07	0.00	0.00	0.00
21	RNN controller					0.02	0.00	0.00	0.44	0.17	0.71	0.38	19.72	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
22	FF step orediction					0.91	**0.87**	0.79	**0.96**	0.93	**0.96**	0.06	4.41	**1.00**	0.86	0.87	0.88	0.82	0.67	**1.00**	0.89
23	Dataset size 2,500					0.69	0.15	0.10	0.67	0.36	0.55	0.18	13.92	0.33	0.06	0.23	0.50	0.00	0.00	0.50	0.00
24	Dataset size 5,000					0.58	0.17	0.10	0.69	0.39	0.65	0.20	11.57	0.67	0.09	0.15	0.67	0.08	0.00	0.30	0.0
25	Dataset size 10,000					0.54	0.55	0.29	0.86	0.65	0.67	0.11	7.17	0.83	0.42	0.69	**1.00**	0.35	0.33	0.90	0.44
26	Dataset size 20,000					0.80	0.72	0.59	0.90	0.84	0.91	0.13	8.81	0.83	0.71	0.85	**1.00**	0.71	0.33	0.70	0.56
27	Dataset size 30,000					0.94	0.86	0.80	0.94	**0.95**	0.94	**0.05**	**4.12**	0.67	0.86	**0.92**	**1.00**	**0.88**	0.33	0.90	**1.00**
28	**Our model**					**0.98**	0.85	**0.84**	0.94	0.94	0.94	**0.05**	4.85	0.83	0.83	0.85	**1.00**	**0.88**	**1.00**	0.70	0.89

Table 4.2: Generalization to New Sentences and Changes in Illumination

	Tasks			Execution statistics		
	Pick	Pour	Seq	Dtc	PIn	MAE
1 Illumination	0.93	0.67	0.62	0.84	0.81	0.07
2 Language	0.93	0.69	0.64	0.86	0.83	0.09
3 **Our model**	**0.98**	**0.85**	**0.84**	**0.94**	**0.94**	**0.05**

generalized the trained behavior to changes in object position, verbal command, or perceptual input. While the task success rate is the most critical metric in such a dynamic control scenario, Table 4.1 also shows the object detection rate (Dtc), the percentage of dispersed cup content inside the designated bowl (PIn), the percentage of correctly dispersed quantities (QDif), underlining our model's ability to adjust motions based on semantic cues, the mean-average-error of the robot's configuration in radians (MAE), as well as the distance between the robot tool center point and

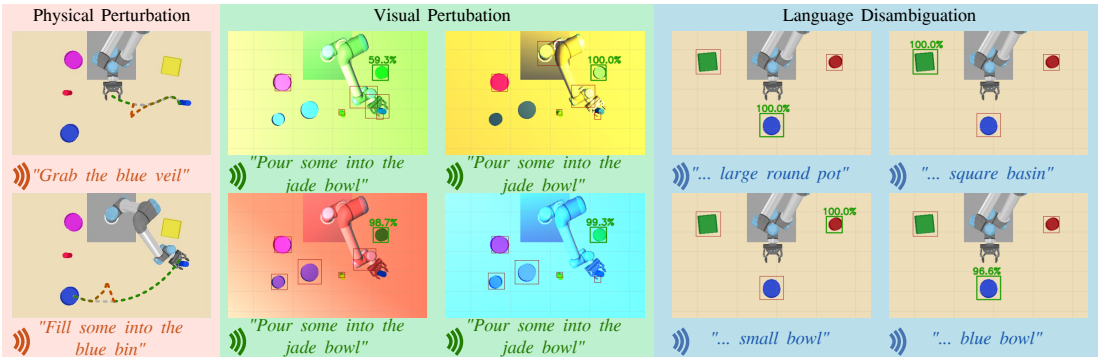

Figure 4.4: Handling of Physical, Visual, and Verbal Disturbances

Generalization of our model towards physical perturbations (left), visual perturbations (middle), and verbal disambiguation (right). All experiments used the same model.

the center of the described target (Dst). The error statistics describe the success rate of the pouring tasks depending on which combination of visual features was used to uniquely describe the target. For example, when no features were used (column "None"), only one bowl was available in the scene, and no visual features were necessary. Further combinations of color (C), size (S), and shape (F) are outlined in the remaining columns. We noticed that the performance decreased to about 70% when the target bowl needed to be described in terms of both shape and size, even though the individual features had substantially higher success rates of 100% and 85%, respectively. It is also notable that our model successfully completed the pouring action in all scenarios in which either the shape or a combination of shape and color were used. The remaining feature combinations reflected the general success rate of 85% for the pouring action.

Generalization to new users and perturbations: Subsequently, we evaluated our model's performance when interacting with a new set of four human users,

from which we collected 160 new sentences. The corresponding results can be seen in Table 4.2, row 2. When tested with new language commands, our model successfully performed the entire sequence in 64% of cases. The model nearly doubled the trajectory error rate but maintained a reasonable success rate. It is also observable that most of the failed task sequences primarily resulted from a deterioration in pouring task performance (a reduction from 85% to 69%). The picking success rate remained at 93%.

Fig. 4.4 depicts different experiments for testing the ability of our model to deal with physical perturbations, visual perturbations, and linguistic ambiguity. In the physical perturbation experiment, we pushed the robot out of its path by applying a force at around 30% of the distance to the goal. We can see that the robot recovered (red line) from such perturbations in our model. In the visual perturbation experiment (middle), we perturbed the visual appearance of the environment and evaluated if the correct object was detected. We can see that, in all of the above cases, the object was correctly detected at a reasonably high rate (between $59\% - 100\%$). Fig. 4.4 (right) shows the model's ability to identify the target objects as described in the verbal commands. Depending on the descriptive features used in the task description, the robot assigned probabilities to different objects in the visual field. These values described the likelihood of the corresponding object being the subject of the sentence – a feature that enabled increased transparency of the decision-making process. Fig. 4.4 (middle) shows examples of the same task executed in differently illuminated scenarios. This experiment highlighted the ability of this approach to cope with perceptual disturbances. Evaluating the model under these conditions yielded a task completion rate of 62% (Table 4.2, row 1). The main source of accuracy loss was the detection network misclassifying or failing to detect the target object.

Baselines: We also compared our approach to two relevant baseline methods. As a first baseline, we evaluated a three-layered LSTM network augmented with extracted features $\mathcal{F}$ of all objects from the object tracker and sentence embedding $\boldsymbol{s}$. The LSTM network concatenated the features in an intermediate embedding and, in turn, predicted the next robot configuration. The second baseline was a current state-of-the-art method called *PayAttention!*, as described in (Abolghasemi *et al.*, 2019). The objective of PayAttention! was similar to our approach and aimed at converting language and image inputs into low-level robot control policies via imitation learning.

Table 4.3 compares the results of the two baselines to our model for the pouring, picking, and sequential tasks. Furthermore, the table also shows the percentage of detected objects (Dtc), percentage of dispersed cup content that ended up in the correct bowl (PIn), and the mean-absolute-error (MAE) of the joint trajectory. For fairness, the models were evaluated using two modes: closed loop (CL) and ground truth (GT). In the first mode, using a closed-loop controller, a model was only provided with the start configuration of the robot. In each consecutive time step, the new robot configuration was generated by the simulator after applying the predicted action and calculating dynamics. In the second mode, using ground truth states, a model was constantly provided with the ground truth configurations of the robot as provided by the demonstration. This mode reduced the complexity of the task by eliminating the effect of dynamics and sensor or execution noise but allowed for easier comparison across methods. Results in Table 4.3 show that the baselines largely failed at executing the sequential task. However, partial success was achieved in the picking task when using the full RNN baseline. Both methods particularly struggled with the more dynamic closed-loop setup, in which they achieved a 0% success rate. Overall, our model (row 5) significantly outperformed both comparison methods. Unlike

our model, the PayAttention! method used a fixed alphabet and required information about the task to be extracted from the sentence before use in the model. In contrast, our model could extract a variety of information directly from natural language. We argue that in our case, adverbs and adjectives played a critical role in disambiguating objects and modulating behavior. PayAttention!, however, primarily focused on objects that could be clearly differentiated by their noun, making it difficult to correctly identify the target objects.

Ablations of our model: We studied the influence of auxiliary losses on model performance. Table 4.1 (rows 1-3) shows the task and execution statistics for different combinations of the auxiliary losses. When training with the trajectory loss (Trj) only, our model successfully completed about 28% of the test cases (row 1). This limited amount of generalization hints at the presence of overfitting. Rather than focusing on task understanding and execution, the network learned to reproduce trajectories. Adding the three remaining controller losses ($\boldsymbol{W}$, ϕ, and Δ_ϕ) aggravated the situation and led to a 0% task completion (row 2). We noticed that attention (Att) was a critical component for training a model with high generalization abilities. Attention ensured that the detected object was in line with the object clause of the verbal task description. A combination of Att and Trj already resulted in a 51% task success rate (row 3). When using the full loss function, including all components, our model achieved an 84% success rate (row 12). This result highlights the critical nature of the loss function, in particular in such a multimodal task. The different objectives related to vision, motion, temporal progression, etc. had to be balanced to achieve the intended generalization.

We also consider an ablation that replaced the attention mechanism with a simple feed forward network. This network took all image features $\mathcal{F}$ as input and gener-

Table 4.3: Comparison to a Baseline and a Current State-of-the-Art Method

			Tasks			Execution statistics		
	GT	CL	Pick	Pour	Seq	Dtc	PIn	MAE
1 Full RNN	✓		0.58	0.00	0.00	0.52	0.07	0.30
2 Full RNN		✓	0.00	0.00	0.00	0.39	0.07	0.39
3 PayAttention!	✓		0.23	0.08	0.00	0.66	0.41	0.13
4 PayAttention!		✓	0.00	0.00	0.00	0.52	0.06	0.53
5 **Our model**		✓	**0.98**	**0.85**	**0.84**	**0.94**	**0.94**	**0.05**

ated an intermediate representation via a combination with the sentence embedding $\boldsymbol{s}$ (without any attention mechanism). All other elements of the approach remained untouched. Table 4.1 (row 4) shows a severe decline in performance when using this modification. This insight underlines the central importance of the attention model in our approach. Pushing the ablation analysis further, we also investigated the impact of the choice of low-level controller. More specifically, we evaluated a variant of our model that used attention but replaced the controller module with a three-layer recurrent neural network that directly predicted the next joint configuration (row 5). Again, performance dropped significantly. Finally, we performed an experiment in which we, again, maintained the attention model but replaced only the motor primitives with a feed forward neural network. This variant produced a similar performance to our controller (row 6); the task performance was only marginally lower, by about 5%. While this was a reasonable variant of our framework, it lost the ability to generate entire trajectories indicating the robot's future motion. Such lookahead trajectories could be of significant importance for evaluating secondary aspects and

safety of a control task (e.g., checking for collision with obstacles, calculating distances to human interaction partners, etc). Therefore, we argue that the specific control model proposed in this paper was more amenable to integration into hierarchical robot control frameworks. Finally, we investigated the impact of the sample size on model performance. Table 4.1 presents results from different dataset sizes in rows 7 to 11. Significant performance increases could be seen when gradually increasing the sample size from 2,500 to 30,000 training samples. However, the step from 30,000 to 40,000 samples (our main model) only yielded a 4% performance increase, which was negligible compared to the previous increases of $\geq 20\%$ between each step.

4.5 Conclusion

We present an approach for end-to-end imitation learning of robot manipulation policies that combines language, vision, and control. The extracted language-conditioned policies provided a simple and intuitive interface to a human user for providing unstructured commands. This represents a significant departure from existing work on imitation learning and enables a tight coupling of semantic knowledge extraction and control signal generation. Empirically, we showed that our approach significantly outperformed alternative methods, while also generalizing across a variety of experimental setups and achieving credible results on free-form, unconstrained natural-language instructions from previously unseen users. While we use FRCNN for perception and GloVe for language embeddings, our approach is independent of these choices and more recent models for vision and language, such as BERT (Devlin *et al.*, 2019a), can easily be used as a replacement. This extensibility is an appealing feature of this methodology and allows for steady improvements in the future. An interesting direction for this work would be to better analyze failure cases rooting

from either providing impossible instructions, or failing to complete the task. By analyzing the model's uncertainty given a new instruction, the user can be informed about potential issues. An intuitive way to convey such information would be to extend the currently one-directional communication channel with the ability of the system to formulate such issues or even use language as a means to explain to a user what task the system has performed.

4.6 Broader impact

Our work describes a machine-learning approach that fundamentally combined language, vision, and motion to produce changes in a physical environment. While each of these three topics has a large, dedicated community working on domain-relevant benchmarks and methodologies, there are only a few works that have addressed the challenge of integration. The presented robot simulation scenario, the experiments, and the presented algorithm provide a reproducible benchmark for investigating the challenges at the intersection of language, vision, and control. Natural language as an input modality is likely to have a substantial impact on how users interact with embedded, automated, and/or autonomous systems. For instance, recent research on the Amazon Alexa Lopatovska *et al.* (2018) suggests that the fluency of the interaction experience is more important to users than the actual interaction output. Surprisingly, "users reported being satisfied with Alexa even when it did not produce sought information"Lopatovska *et al.* (2018).

Beyond the scope of this paper, having the ability to use a natural-language processing system to direct, for example, an autonomous wheelchair Williams and Scheutz (2017) may substantially improve the quality of life of many people with disabilities. Natural-language instructions, as discussed in this paper, could open

up new application domains for machine learning and robotics, while at the same time improving transparency and reducing technological anxiety. Especially in elder care, there is evidence that interactive robots for physical and social support may substantially improve quality of care, as the average amount of in-person care in only around 24 hours a week. However, for the machine-learning community to enable such applications, it is important that natural-language instructions can be understood across a large number of users, without the need for specific sentence structures or perfect grammar. While far from conclusive, the generalization experiments with free-form instructions from novel human users (see Sec.4.4) are an essential step in this direction and represent a significant departure from typical evaluation metrics in robotics papers. In particular, we holistically tested whether the translation from verbal description to physical motion in the environment brought about the intended change and task success.

Even before adoption in homes and healthcare facilities, robots with verbal instructions may become an important asset in small and medium-sized enterprises (SMEs). To date, robots have been rarely used outside of heavy manufacturing due to the added burden of complex reprogramming and motion adaptation. In the case of small product batch sizes, as typically used by SMEs, repeated programming becomes economically unsustainable. However, using systems that learn from human demonstration and explanation also comes with the risk of exploitation for nefarious objectives. We mitigated this problem in our work by carefully reviewing all demonstrations, as well as the provided verbal task descriptions, in order to ensure appropriate usage. In addition to the training process, another source of system failure could come from adversarial attacks on our model. This is of particular interest since our model does not only work as software but ultimately controls a physical

robotic manipulator that may potentially harm a user in the real world. We addressed this issue in our work by utilizing an attention network that allowed users to verify the selected target object, thereby providing transparency regarding the robot's intended behavior. Despite these features, we argue that more research needs to focus on the challenges posed by adversarial attacks. This statement is particularly true for domains like ours in which machine learning is connected to a physical system that can exert forces in the real world.

Chapter 5

VERBAL DESCRIPTIONS FROM MULTIMODAL ROBOT DATA

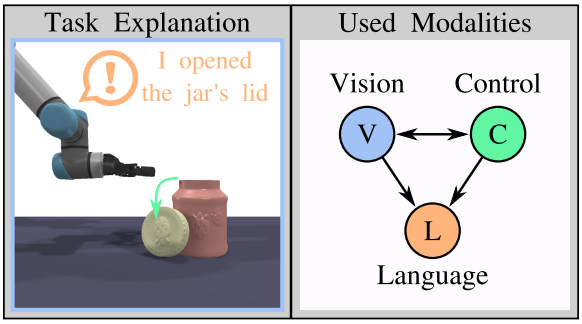

Figure 5.1: Language Generation Modalities

Overview of the used modalities in the language generation task.

Giving robots the ability to communicate with their environment and other agents in it efficiently is an integral step towards human-robot collaboration. The previous chapter introduced an approach that combines vision and language to generate the robot's motion, allowing users to provide instructions that the robot then executes. However, most approaches lack the ability to efficiently communicate with the user if the task fails, doesn't run as expected, or when additional information is needed. In such cases, natural language is an efficient and intuitive way to express the robot's action. While the previous chapter introduced a multimodal approach that uses vision and language to generate motion, this chapter presents an approach to allow a robotic system to explain its actions to a user from two sample images of a task and a motion trace of the robots. Figure 5.1 shows the relation between the three used modalities. Using multiple modalities for language generation allows our approach to explain the

robot's action in greater detail than single-modality approaches and current state-of-the-art image captioning methods. Furthermore, we utilize a Bayesian approach to reason over our model's uncertainty and its generated task explanation to gain further insights into the robot's actions.

5.1 Introduction

Language plays a critical role in our everyday lives. Throughout history, language has been the most efficient way for humans to convey ideas, concepts, and thoughts among each other, ultimately becoming an integral part of human society. Therefore, it is of critical importance that robots can utilize this form of communication, especially as they are moving close to our everyday lives. Particularly when engaging with non-expert users who might not be able to program robots directly to teach them how to achieve new tasks, providing a language interface to the robot is of utmost importance.

Previous work on incorporating language with robot systems primarily focused on understanding verbal action descriptions to generate a control policy that achieves the described task. However, to fully integrate these robot systems into our homes, it is important to understand human instruction and close the loop by generating language descriptions of the robot's actions. In this chapter, we propose an approach that generates a verbal action description from images and an execution trace of the robot's motion. We also demonstrate this approach in a multi-robot scenario that shows our model's ability to generate concise action descriptions of various tasks. Figure 5.2 shows an overview of the proposed method. Given two images of the task, a pre-and post-action image, together with the motion trace of the two robots, our model can automatically generate a plausible description of the action that was

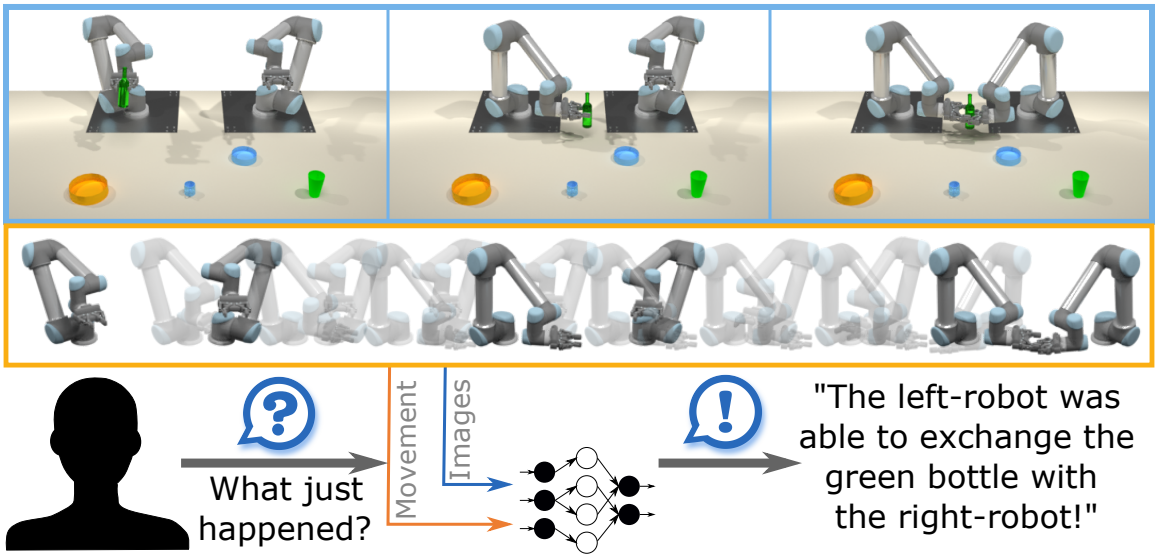

Figure 5.2: Multimodal Language Generation Introduction

Our model can generate a verbal description of the task that is completed by the two robots from a pre- and post-action image as well as a motion trace of the robots' joint movement.

completed by the robots. In this example of an exchange, the model describes the task as *"The left-robot was able to exchange the green bottle with the right robot"*. It is important to note here that the work in this chapter does not attempt to describe *why* the robots performed the action, but rather *what* has been performed. Such action descriptions have multiple use-cases. For example, these descriptions can be used when the robots can not directly be observed or to allow for better accessibility for people with visual impairments. Furthermore, language can provide additional context that cannot be extracted from vision alone. For example, from a single video of a robot completing a task, it is unclear if the system performed quickly or slowly; however, an additional explanation can provide the necessary context for downstream tasks or follow-up actions.

In summary, this chapter introduces the following contributions:

1. An approach capable of generating verb descriptions of tasks completed by multiple robots.

2. An empirical evaluation of our model, demonstrating its ability to generate more accurate verbal task descriptions, as compared to a current state-of-the-art image captioning baseline.

3. An approach to better understand the model's ability to generate sentences in multiple variations of the trained actions by reasoning about the model's uncertainty.

5.2 Background

The benefit of using natural language in conjunction with robotics has long been discovered as an intuitive communication channel between humans and robots (Winograd, 1972). Given this early approach, the work in Dillmann and Friedrich (1996) and Kress-Gazit *et al.* (2008) extended the idea of Winograd (1972) by proposing a programming-by-demonstration approach as well as a translations approach of verbal instructions to a plan specification, respectively. However, these approaches usually utilize symbolic language processing, semantic analysis, or formal gramars (Jurafsky and Martin, 2009). Following the recent advances in deep learning, allowing for more complex language models (Sutskever *et al.*, 2014; Vaswani *et al.*, 2017; Otter *et al.*, 2021). The usage of language in conjunction with robotics has also seen further advances, partially due to the surge of interest in collaborative robotics (Hicks and Simmons, 2019; Thomaz *et al.*, 2016). Especially in the domain of collaborative robotics, giving robots the ability to utilize language does not only remove the need

for other, potentially cumbersome interfaces but also has the potential of increasing the user's trust in the system (Martelaro *et al.*, 2016; Javaid and Estivill-Castro, 2021). While trust plays an important role in human-robot collaboration, most of today's focus on combining language and robotics falls into the category of instruction following Stepputtis *et al.* (2020b); Lynch and Sermanet (2021) with the goal of learning a policy from language and potentially other modalities that completes the described action. A complete overview of this type of task can be found in the surveys of Tellex *et al.* (2020) and Liu and Zhang (2019). In addition to an increased interest in collaborative robotics, the surge of deep learning in recent years enabled the usage of even more complex models that automatically learn various concepts in images and language. Especially the introduction of novel concepts like Transformers (Vaswani *et al.*, 2017) and self-attention, resulting in large language models like BERT (Devlin *et al.*, 2019b) allowed for a shift towards semantic image processing and language understanding (Stefanini *et al.*, 2021). One use-case of these advances is demonstrated in Huang *et al.* (2020b) where human action descriptions have been extracted from dynamic videos.

The approach presented in this chapter utilizes a multimodal approach that combines vision and motion in order to generate a verbal description of the executed task. In contrast to explainable AI (XAI) (Chakraborti *et al.*, 2021), the approach presented here strictly focuses on *what* has been done, rather than explaining *why* the robots have performed the actions they did. This approach allows users to utilize additional communication channels in order to advance human-robot interaction further. The approach discussed in this section is most similar to the work presented in Yoshino *et al.* (2020), but instead of K-means clustering, it uses an end-to-end learning approach that utilizes recent advances in attention models.

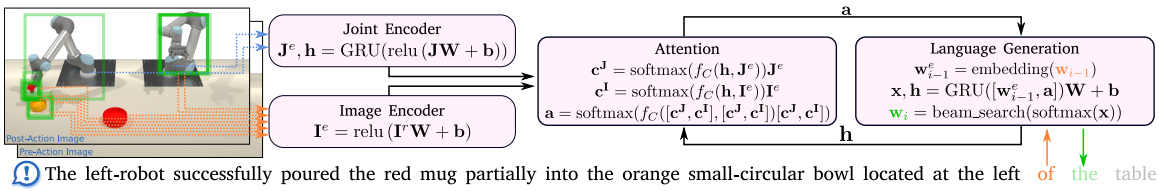

Figure 5.3: Language Generation System Overview

Our model generates a sentence from two images, one from before and one from after the action, as well as a motion trace of the two robots.

5.3 Problem Statement and Approach

The problem of generating a verbal task description $\boldsymbol{s}$ (a sequence of words) that concisely describes the action $a \in \mathcal{A}$ from the set of all actions $\mathcal{A}$ is defined as the problem of generating the next word $\boldsymbol{s}_{n+1}$ in sentence $\boldsymbol{s}$ as described in equation 5.1

$$\boldsymbol{s}_{n+1} = f_\Theta\left(\boldsymbol{I}_{pre}, \boldsymbol{I}_{post}, \boldsymbol{J}, \boldsymbol{s}_{0\to n}\right) \tag{5.1}$$

where the images $\boldsymbol{I}_{pre}$ and $\boldsymbol{I}_{post}$ are describing the image of the robots' immediate environment from before and after the action, respectively. Furthermore, our approach utilizes multiple modalities in order to generate a suitable task description, thus our model $f_\Theta(...)$ also receives an execution trace $\boldsymbol{J} \in \mathbb{R}^{T\times 14}$ containing T measurements of the robots' joint and gripper configuration during the action. Finally, in order to generate the next word $\boldsymbol{s}_{n+1}$, we also provide our model with the so-far generated sentence $\boldsymbol{s}_{0\to n}$ of length n. Each sentence is initialized with the start-of-sentence token $\langle sos \rangle$ and completes when either the maximum sentence length of 28 words is reached, or the end-of-sentence token $\langle eos \rangle$ is generated. For each word $\boldsymbol{s}_n \in \boldsymbol{s}$, our problem is defined as a classification task over N classes, where N is the size of our dictionary $\mathcal{D}$.

After the training of our model is completed, our model can generate a suitable

task description given a set of two new images $\boldsymbol{I}_{pre}$ and $\boldsymbol{I}_{post}$ as well as a motion trace $\boldsymbol{J}$ for any task $a \in \mathcal{A}$.

5.3.1 Data Pre-Processing

Instead of providing our model directly with raw images, we are utilizing Detectron2 (Wu *et al.*, 2019) in order to generate regions of interest (RoI) in the images. These regions capture all potentially relevant objects and robots in the environment image. In order to make the model applicable to our specific actions $\mathcal{A}$ and the objects used within the tasks, we fine-tuned Detectron2 on 10.000 randomly generated environments.

To allow our model to understand language, we represent each word w as a trainable embedding $\boldsymbol{w}^e \in \mathbb{R}^{256}$. In addition to the words used in our training data, expand the dictionary with the $\langle sos \rangle$, $\langle sos \rangle$, and $\langle padding \rangle$ tokens to form our dictionary of size N. In addition to the $\langle sos \rangle$ and $\langle sos \rangle$ tokens, the $\langle padding \rangle$ token is used to pad each sentence in the training dataset to its maximum length of 27 words.

5.3.2 Sequence to Sequence Language Generation

Equation 5.1 defines our problem: Predicting the next word in sentence $\boldsymbol{s}$ that describes the robots' actions shown by two images $\boldsymbol{I}_{pre}$ and $\boldsymbol{I}_{post}$ and a motion trace $\boldsymbol{J}$. We formulate our problem as a classification problem over the dictionary $\mathcal{D}$ of size N. The sentence that needs to be generated by the model is initialized with only the $\langle soc \rangle$ token, and our model generates one more word in each iteration. Figure 5.3 provides a general overview of the approach, starting with independent embeddings for each modality before applying our attention approach and language generation head.

Encoding Modalities

Images are processed by our pre-trained Detectron2, extracting the 25 image regions that have the highest confidence according to the detection algorithm. The feature vector of each of the 25 regions from each image are concatenated into a single matrix $\boldsymbol{I}^r \in \mathbb{R}^{R \times 1024}$ where $R = 50$ is the total number of image regions. Each row of $\boldsymbol{I}^r$ is then transformed with a trainable non-linear transformation $\boldsymbol{I}^e = \text{relu}\left(\boldsymbol{I}^r\boldsymbol{W} + \boldsymbol{b}\right)$ with trainable variables $\boldsymbol{W}$ and $\boldsymbol{b}$ of dimension 1024.

In order to encode the robots' motion $\boldsymbol{J}$, which is a sequence of robot states, a single GRU cell is used, returning the sequence of intermediate results $\boldsymbol{J}^e$ as well as the final hidden state $\boldsymbol{h}$ after re-scaling each element in $\boldsymbol{J}$ non-linearly to the common size of 1024. Equation 5.2 formally describes the joint encoding with trainable weights $\boldsymbol{W}$ and $\boldsymbol{b}$.

$$\boldsymbol{J}^e, \boldsymbol{h} = \text{GRU}(\text{relu}\left(\boldsymbol{J}\boldsymbol{W} + \boldsymbol{b}\right)) \tag{5.2}$$

Cross- and Self-Attention

After having generated the individual joint representations $\boldsymbol{I}^e$ and $\boldsymbol{J}^e$, as well as the hidden state $\boldsymbol{h}$, we employ two stages of cross attention and one layer of self-attention to generate a task context $\boldsymbol{c} \in \mathbb{R}^{C \times 1024}$ that is later used to generate the next word in $\boldsymbol{s}$. Generally, equation 5.3 describes the general process of attention used in the cross and self-attention parts of our model.

$$\begin{aligned} \boldsymbol{C} &= f_C(\boldsymbol{Q}, \boldsymbol{X}) \\ &= \tanh\left(\left(\boldsymbol{Q}\boldsymbol{W}^q + \boldsymbol{b}^q\right) + \left(\boldsymbol{X}\boldsymbol{W}^x + \boldsymbol{b}^x\right)\right)\boldsymbol{W} + \boldsymbol{b} \end{aligned} \tag{5.3}$$

The variables $\boldsymbol{W}, \boldsymbol{W}^x$, and $\boldsymbol{W}^q$ as well as $\boldsymbol{b}, \boldsymbol{b}^x$, and $\boldsymbol{b}^q$ are trainable and of common dimension 1024. While the function $f_C(...)$ is used multiple times, the dimensionality

of the weights remains the same. Further note that depending on the inputs, the resulting context $\boldsymbol{C^J} = \text{softmax}\left(f_C(\boldsymbol{h}, \boldsymbol{J}^e)\right)\boldsymbol{J}^e$ and $\boldsymbol{C^I} = \text{softmax}\left(f_C(\boldsymbol{h}, \boldsymbol{I}^e)\right)\boldsymbol{I}^e$ have a dimensionality of $\mathbb{R}^{T\times 1024}$ and $\mathbb{R}^{Rx1024}$ respectively. For the last step, self attention, which utilizes $f_C(...)$ again, the sequences need to be padded to a common length of 296, which is the maximum possible length of the robots' motion. We then generate the final context vector $\boldsymbol{C} = \text{softmax}(f_C([\boldsymbol{C^J}, \boldsymbol{C^I}], [\boldsymbol{C^J}, \boldsymbol{C^I}]))[\boldsymbol{C^J}, \boldsymbol{C^I}]$.

Word Prediction

The word generation is implemented as a classification problem over N classes in dictionary $\mathbb{D}$. At each iteration of our model, the generated word is concatenated to the sentence $\boldsymbol{s}$ until either the maximum sentence length is reached or the $\langle eos \rangle$ is generated. Given the context $\boldsymbol{C}$, we generate the next hidden state $\boldsymbol{h}_{n+1}$ as well as the word embedding $\boldsymbol{x}$ as described in Equation 5.4.

$$\boldsymbol{x}, \boldsymbol{h} = \text{GRU}\left([\text{embedding}\left(\boldsymbol{s}_n\right), \boldsymbol{a}]\right)\boldsymbol{W} + \boldsymbol{b} \tag{5.4}$$

In contrast to the previous use of the trainable variables $\boldsymbol{W}$ and $\boldsymbol{b}$, the dimensionally here is reduced to the size of the dictionary $\mathcal{D}$ in order to allow for a classification task from $\boldsymbol{x}$. The GRU cell in Equation 5.4 has 1024 hidden units and also generates a new hidden state $\boldsymbol{h}$ that is used in Equation 5.3 during the generation of the context $\boldsymbol{C}$ in the next iteration. Finally, the next word is classified by $\boldsymbol{s}_{n+1} = \text{argmax}(\text{softmax}(\boldsymbol{x}))$.

While using argmax to decide on the next word in $\boldsymbol{s}_{n+1} \in \boldsymbol{s}$ results in reasonable sentences, we decided to utilize beam-search with a beam-width of five to generate the final sentence by always keeping the five most promising sentences. Empirically, using beam-search increases the variability of chosen templates during sentence generation, while the greedy approach using argmax directly mostly uses a single template to

generate a sentence. While both approaches are feasible, we decided to use beam-search for increased linguistic variability.

5.3.3 Training

We start our training process by fine-tuning Detectron2 on all of our objects and robots in the environment to be able to generate proper RoIs. To speed up the training of our language generation model, we pre-process the entire dataset with the fine-tuned Detectron2 model to generate the 25 most confident RoIs for each image in the dataset. While our model is capable of end-to-end inference, we chose this optimization during training such that the full-resolution images do not need to be part of the training dataset.

During training, we optimize the sparse categorical cross-entropy loss for a given one-hot encoding of the predicted word and the target word at each step. Between steps, teacher forcing is used by giving the correct word as input to the decoder instead of the predicted word from the previous time step until the maximum length of the target description is reached. Furthermore, we employ a 70% dropout in the attention layers of Equation 5.3 The model is optimized with an Adam optimizer using a learning rate of 0.001 over 25 epochs with a batch size of 32. On average, training the model until convergence takes about 60 minutes.

5.4 Evaluation and Results

By using our model, verbal descriptions of multiple tasks can be generated from two images and a motion trace of the robots' action. This ability is demonstrated on five different tasks implemented in a simulated table-top manipulation task utilizing two universal robots UR5 robots. Besides the six degrees of freedom (DoF) of each

robot, each robot utilizes a parallel jaw gripper to handle objects. The five tasks are defined as follows:

- **Picking:** A cup or mug on the table in front of the robots is picked up by one of the robots while the second robot is not performing any action.
- **Pouring:** A cup that is held by one of the robots is poured into one of the bowls on the table. In this task, the active robot can either pour a little or pour a lot.
- **Exchange:** An object that is held by one robot is exchanged with the other robot, resulting in the second robot holding the object
- **Open:** The saucepan object has a lid that can be opened and placed next to the pan.
- **Twist:** In this task, one robot acts as a fixture, holding a jar with a lid that can be unscrewed from it, while the second robot attempts to open the jar by holding onto the lid and opens it with a twisting motion.

All five tasks have been implemented with a motion planner in CoppeliaSim (Rohmer *et al.*, 2013; James *et al.*, 2019), allowing for precise physics simulation of our objects and robot motions. During the experiments, the simulator runs in real-time with a fixed step size of 50ms. The objects utilized in our task are shown in Figure 5.4, and all ten objects are available in six colors, except the saucepan, which is unique. All objects are monolithic in nature, except for the jar, which has a lid that can be unscrewed, and the saucepan, which also has a lid, but it can simply be lifted off. The objects in the top-right corner of the image are used for geometric variations

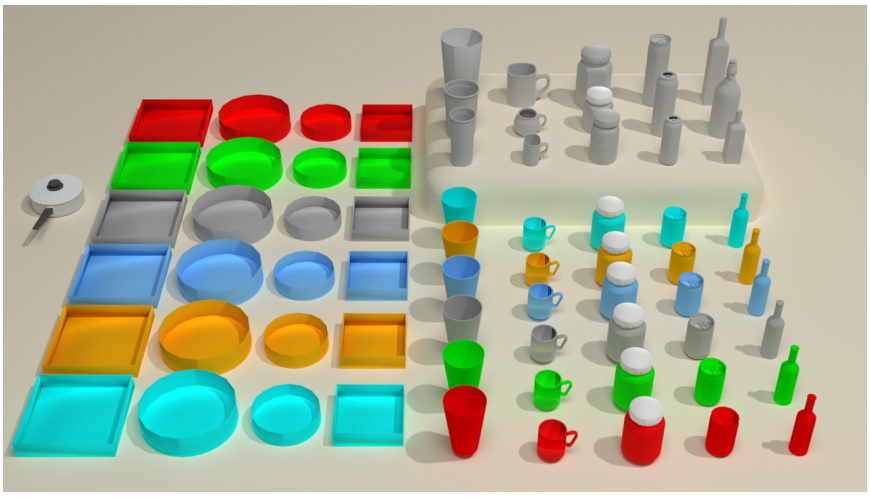

Figure 5.4: Overview of All Possible Scene Objects

All our objects exist in all six colors and the objects in the back-right are used for geometric variation.

which is evaluated later in this section. Unlike shown in Figure 5.4, the objects used for geometric generalization are also available in six different colors.

5.4.1 Data Collection and Training

In order to collect the training data for the model, the five basic motions are executed in the simulator to collect the pre and post-action images as well as to record the robot motion for every time step. Overall, a dataset of 40000 demonstrations for training and an additional 1500 demonstrations for testing have been collected. In order to later evaluate unseen action-object pairs, we do not include the glass in the exchange action nor the jar in the pouring action. However, other actions include the glass and the jar, as well as the respective actions with other objects from the dataset. This held-out object-action combination allows us to later evaluate the model's ability to generalize the learned language generation.

The sentences in our demonstrations are generated automatically with a template system that can generate plausible task descriptions given an environment and desired task. Our sentences are generated from a set of five templates for each task, as well as a set of synonyms for commonly used words like colors, objects and actions. Generally, we are interested in describing six features in each sentence: action, agent, direction, object, color, and Quantity. However, depending on the described task, not all features are necessary. For example, the quantity is only used when describing a pouring task as it is not applicable for other actions.

5.4.2 Evaluation Metric

Our model creates a description for an observed task. The resulting task description is evaluated on multiple common NLP metrics (Sai *et al.*, 2020) including *BLEU*, *ROUGE*, *METEOR*, *CIDER*, as well as our own metric that. In this work, we decided to add our own metric to evaluate the six described features by using an inverse template generator that can take a generated sentence and converter it back into the used template as well as the features the model chose to describe the task. This additional metric provides further insights into the quality of the generated language while the general NLP scores might not pick up the nuances of our task. Further details on the NLP metrics can be found in our discussion in section 5.4.4.

For our own metric, we extract the features and respecive sentence template that the model generated and compare them with the training template and features. We calculate the percentage of correctly detected attributes under consideration of valid synonyms. Our inverse template generator is able to handle such synonyms and varying templates for all five tasks since the possible sentences the model can generate are limited, and no out-of-distribution words will ever be generated. In addition to

	Input Features		Action	Agent	Direction	Object	Color	Quantity	Object & Color	All
Model	Joints	Images	/1500	/1500	/900	/1800	/1200	/300	/1200	/1500
1 Ours	✗	✓	98.6%	96.2%	76.87%	99.94%	97.81%	98.67%	97.71%	83.41%
2 Ours	✓	✗	100.0%	99.8%	92.13%	50.08%	19.42%	100.0%	7.32%	47.1%
3 Ours	✓	✓	100.0%	98.6%	95.87%	99.89%	96.2%	98.0%	96.11%	93.2%
4 M^2T	✗	✓	100.0%	100.0%	86.43%	95.94%	98.83%	99.33%	92.76%	87.01%
5 M^2T	✓	✓	100.0%	100.0%	87.55%	95.89%	98.08%	99.67%	92.01%	87.14%

Table 5.1: Language Generation: Results and Baselines

the six feature metrics, we also add the *Object & color* and *ALL* metric to capture if an object was fully described and if all features of the entire task have been identified correctly. Table 5.1 reports the result of our metric based on the extracted features while Figure 5.6 compares the standard NLP scores.

5.4.3 Language Generation Results

We evaluate our model with our custom metric described in Section 5.4.2 and report the results in Table 5.1. Results are reported on 1500 test demonstrations that are equally distributed over the five different actions, resulting in 300 tests per action type. However, some attributes overlap between different actions, potentially increasing the number of test cases for each feature. For example, each action involves one robot and as a result, the *Agent* feature in Table 5.1 is evaluated on all 1500 test-cases. On the other hand, the *Quantity* feature is only applicable in the pouring task and is thus only evaluated in the 300 pouring actions. Yet, maybe counterintuitively, the object feature is evaluated in 1800 cases even though there are only 1500 test cases. This is due to the pouring task in which the poured and target objects are referenced, thus increasing the number of objects by 300.

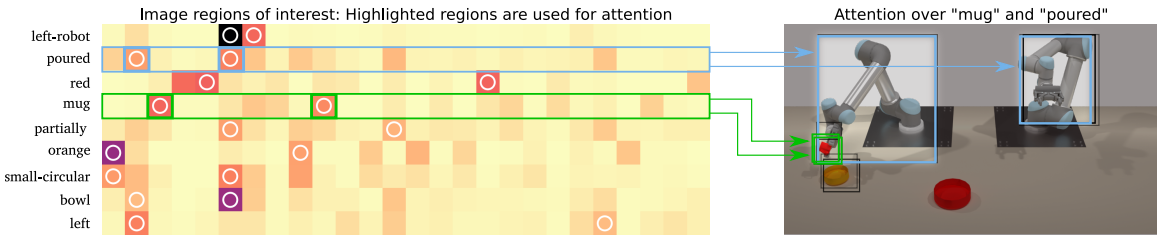

Figure 5.5: Attention to Image Regions During Task Generation

The heat-map shows the attention to certain image regions when generating key-words of a task description: *"The left-robot successfully poured the red mug partially into the orange small-circular bowl located at the left of the table"*. Darker colors indicate higher attention, and circles indicate the two highest weights.

The main contribution of this work is the combination of multiple modalities with the hypothesis that the combination of modalities yields more accurate task descriptions. The main results of our model when combining both modalities are shown in line 3 of Table 5.1 while the results with only one of the modalities are shown in lines 1 and 2. Combining the modalities provides a semantically correct sentence in 93.2% of the test cases where most of the failures come from not identifying the direction of the movement or color of the target object correctly. This is a drastic performance increase over the single modality models with 83.41% and 47.1% accurate sentences when only using images or joints, respectively. As expected, detecting the color of objects from the robot's motion alone results in only 7.32% accuracy in line 2, while detecting the direction of the two images alone is difficult for the image-only model. Overall, these results clearly indicate the benefit of using multiple modalities to describe robot actions as they complement each other.

Figure 5.5 shows our model's attention over the various image regions on a post-condition image of the pouring task. In order to generate the sentence *"The left*

robot successfully poured the red mug partially into the orange small-circular bowl located at the left of the table", the model attends various image regions for the most relevant words in the generated sentence. The two regions with the highest attention are highlighted with a white circle, while darker colors generally indicate higher attention. Most noticeably, the system correctly identifies the image regions for the *pouring* action by observing the state of each robot (highlighted in blue). Furthermore, the image regions containing the mug have been clearly identified and highlighted in green. Other concepts, like the word *red*, also have clear image regions but are not directly shown in the picture.

Additionally, we also compared our model with a current state-of-the-art image-captioning baseline to show the benefit of using a multimodal approach. For this work, we chose Meshed-Memory Transformers (M^2T) (Cornia *et al.*, 2020) that we expanded to take two images as well as two images and sequence of joint positions by concatenating the additional inputs to the original input image. Table 5.1 shows the results of M^2T in line 4 and 5. While the image-only baseline in line 4 outperforms our image-only model in line 1 by 3.6%. However, comparing the multimodal baseline in line 5 with our multimodal model in line 3, the performance of our model increases by 6.06% over the baseline model. We attribute this additional performance gain to our attention approach that allows the modalities to be combined efficiently.

5.4.4 NLP Scores

In addition to our own metric, we also evaluate our model on seven common NLP metrics, namely BLEU-1/2/3/4, METEOR, ROUGE, and CIDEr (Sai *et al.*, 2020). Figure 5.6 shows the results when evaluating the generated sentences of our model and the baseline against the ground-truth sentence on common NLP baselines. In

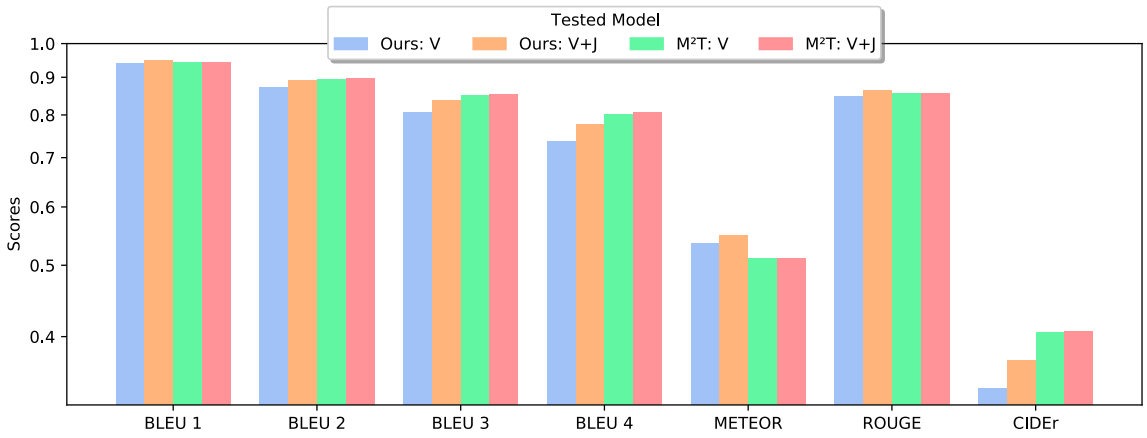

Figure 5.6: Evaluation of the Proposed Method on Common NLP Baselines Comparison of our model and the baseline on common NLP scores: BLEU, METEOR, ROUGE, and CIDEr. Note that all scores have been normalized between $[0, 1]$ according to their respective max values.

all metrics, using a multimodal approach increases the overall score of our as well as the baseline model. However, the baseline model outperforms our model when using the BLEU and CIDEr metrics, while our model marginally outperforms the baseline on METEOR and ROUGE. However, when looking at the results of our metric in Table 5.1, our model outperforms the baselines and single-modality models by large margins. We argue that this discrepancy is due to the way these metrics are evaluating sentences. Our task requires precise wording given the features we are interested in, yet these metrics are not well suited to capture such issues since a single mistake in, for example, determining the colors is still a grammatically correct statement, yet a semantically wrong sentence. However, unlike our own metric, failing to describe the task correctly will not result in a significantly lower score.

	Generalization	Action	Agent	Direction	Object	Color	Quantity	Object & Color	All
1	Light Change	100.0%	100.0%	96.72%	100.0%	78.46%	95.0%	78.46%	86.17%
2	Motion Change	98.67%	100.0%	72.86%	99.0%	68.08%	87.75%	66.67%	52.67%
3	Novel Object	100.0%	99.01%	88.23%	81.74%	86.84%	96.0%	67.10%	67.33%
4	Novel Object-Action	100.0%	100.0%	96.0%	93.33%	92.57%	92.0%	87.16%	81.0%

Table 5.2: Generalization to Environment Changes and New Objects

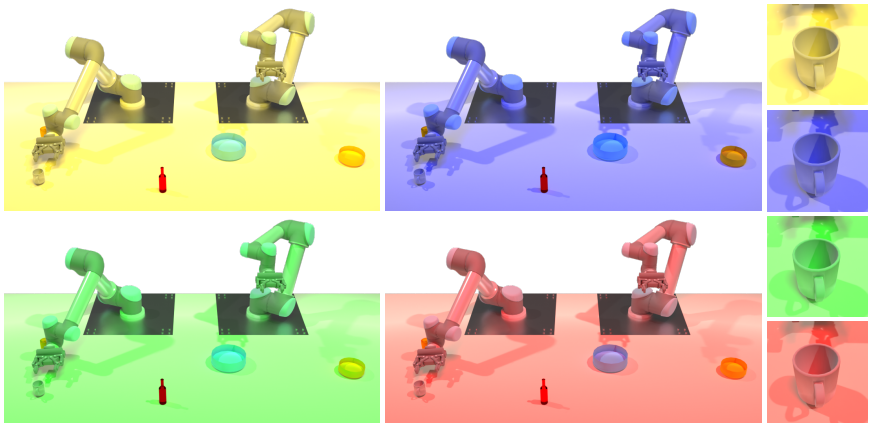

Figure 5.7: Generalization to Changing Lighting Conditions

Changes in the lighting conditions have a drastic effect on the gray mug, as its color changes noticeably with different environment illumination.

5.4.5 Generalization

In order to show our model's generalization capabilities, we show results on varying light conditions and robot motions, as well as changing objects and action-object pairs. To limit the generalization capabilities to our model only, Detectron2 always included the visual changes in order to provide our model with correct feature embeddings. With that, we expect our model from line 3 of Table 5.1 to perform in all the generalization tests.

Ideally, our model's generated sentence should be independent of varying lighting

conditions as well as modifications of the robot's motion assumed it still completes the same task. While we do not claim any contributions for object detection, Figure 5.7 shows four examples of our changed lighting conditions on a pickup task in which a gray cup needs to be lifted from the left side of the table. As can be seen in the figure, the cup appears to have a different color depending on the changed lighting conditions. Despite the color changes on our objects, our model still accurately describes 83.17% of the tasks as shown in line 1 of Table 5.2 with the main failures coming from the color prediction that loses about 18% accuracy when compared to line 3 in Table 5.1.

Line 2 in Table 5.2 reflects our model's performance when objects, and thus the grasping point, are rotated by ±45 degrees, causing the robot to perform slightly different motions. The model has been evaluated on 50 scenarios from the picking, pouring, and opening tasks each. The overall task performance in this scenario is only 52.67%, indicating that the model does not generalize well to the changing motions. While changing the motions does not completely fail, the root cause for the reduced performance still needs to be determined.

Lines 3 and 4 evaluate novel encounters in the environment by either adjusting the known objects' geometry or using a known object for a new task, respectively. The new objects are shown on the top-right of Figure 5.4 while line 4 demonstrates our model's ability to describe known objects in new tasks. For the latter, we are using the green glass for an exchange task and a jar for a pouring task. Given these changes, our model reaches 67.33% accuracy on novel objects while achieving 81% accuracy by using known objects for tasks they have not been used for in the training data. This shows that our model is generally able to understand and separate objects from actions as it can re-combine them; however, generalizing to entirely new objects remains challenging.

5.4.6 Model Uncertainty

Lastly, we evaluate our model's certainty when generating sentences by utilizing stochastic forward passes (Gal and Ghahramani, 2016). The general concept is to use stochastic forward passes Gal and Ghahramani (2016) by utilizing dropout during inference. In order to generate a distribution over $\boldsymbol{x}$ from Equation 5.4, we pass the same input 500 times through our network as a single batch and treat the resulting weights for each word of the dictionary as a distribution. The results can be seen in Figure 5.8 where the weights in $\boldsymbol{x}$ for each word are shown. However, for simplicity, we only show words that exceed a threshold-weight of 0.1 as the vast majority of the N words from the dictionary have a negligible or zero likelihood. In each plot, the most likely word is highlighted in blue; the second, third, and fourth choices are orange, green, and red, respectively. The generated sentence, as well as the alternative words, can be seen underneath each plot. The six plots show the model's language generation capabilities in the base-case in the first plot, as well as five generalization tests. The basic task is to pick up a red mug from the front of the table, and the task is exactly the same for all five generalization tests, except the respectively described variation. The first variation is a change in the robot's motion that has not been part of the training process. The second plot shows the word weights using a modified motion that grasps the object from a 45-degree angle. In comparison to the first plot that shows the pickup under ideal conditions, it can be seen that the robot and action still are identified correctly despite the changed movement; however, the wording changed from *front* to *top right* with an increased variance, which is a wrong description.

The third figure shows the word weight when the light in the scene changes by using a strong blue component. Here, we are evaluating our model's ability to handle

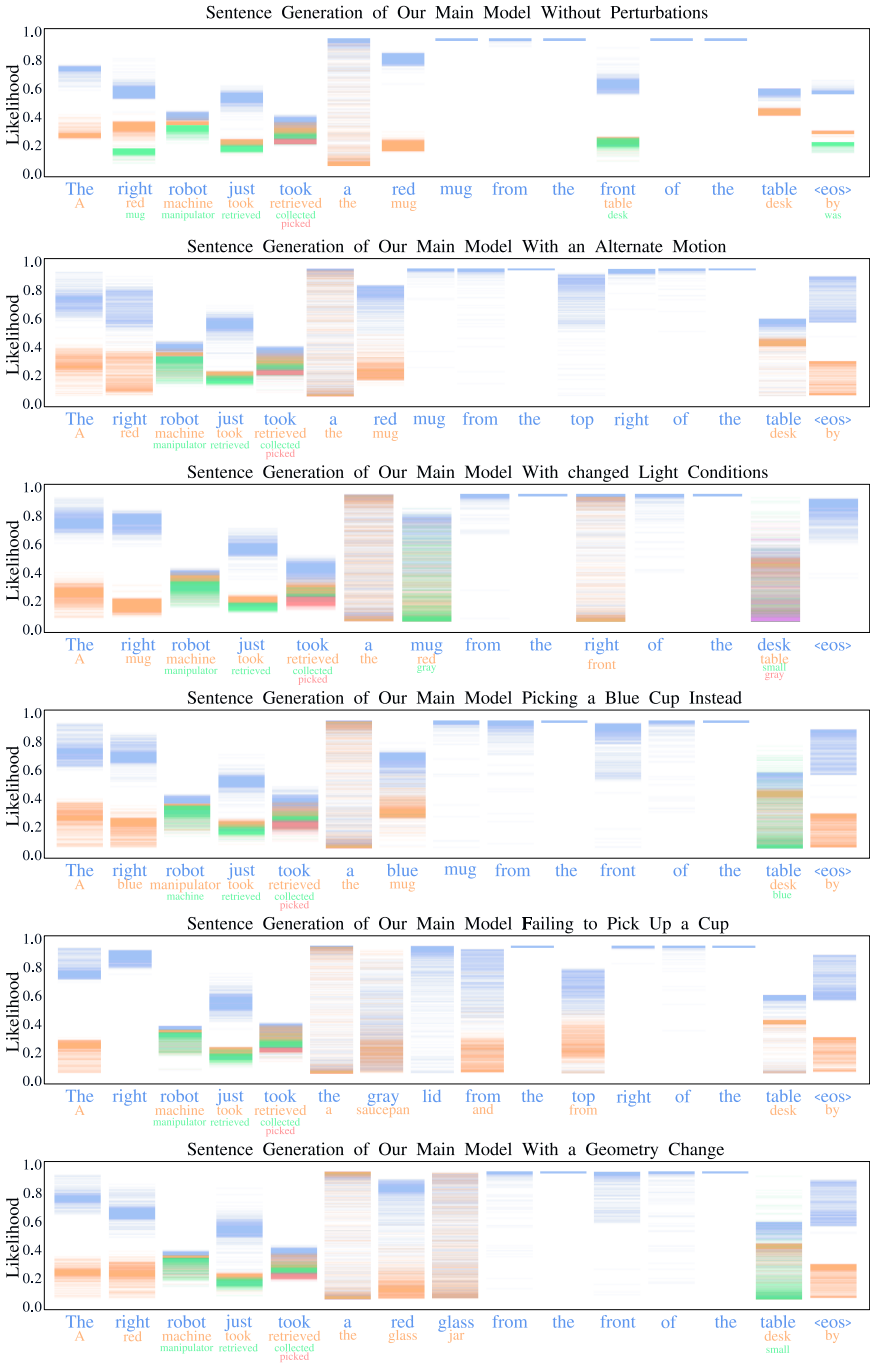

Figure 5.8: Stochastic Forward Passes for Language Generation

Most likely words are blue, where as second, third, and fourth choices are orange, green, and red, respectively.

visual perturbations. As an interesting result, our model is very uncertain about the object and decided not to describe it using a color word. However, the confusion is primarily due to the uncertainty of whether or not to use the color word *red*, which would regardless be a correct choice. This shows our model's robustness with respect to visual perturbations.

In the fourth graph, we tested the same picking task as in plot one; however, we changed the object's color from red to blue. Like the first plot, the model generates a correct sentence while being certain about the changed color, showing that our attention model is able to attend to the correct features as other components of the sentence have not been changed.

The fifth graph shows a situation in which the pickup action failed by throwing over the object. This has been achieved with our physic simulator and a slightly altered trajectory causing this failure. In this case, the generated sentence is still certain about which agent performed the task and what task might have been performed; however, the variances across the words describing the object location and object itself are significantly higher than in the other generalization experiments. This indicates that this model could successfully be used to inform the user about a generally increased variance and thus indicate that the currently executed task is not executed as expected. Especially as our model is forced to generate a sentence until the maximal length is reached or the end-of-sentence token is generated, this additional information can significantly increase the usability of the system.

Finally, we also evaluated our model's performance when using a slightly different object. In this case, a different 3D model of a mug was used whose appearance brought it closer to the one of a jar. In the sentence generation, it is still clearly visible that the only changes occur in words related to the object, ultimately confusing it with a

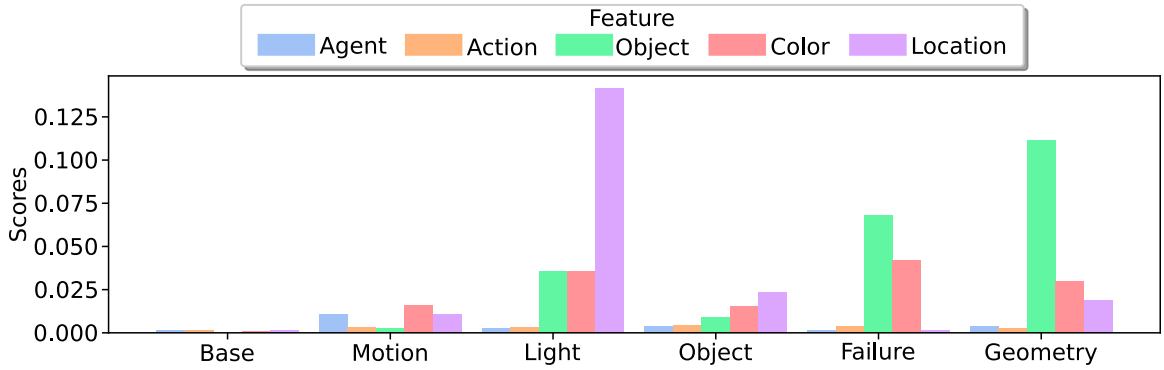

Figure 5.9: Variation of Generated Words

Variance of the chosen feature words in the five ablations as compared to the base case.

glass or jar. However, the uncertainty is drastically increased, which could be utilized to inform the user about an unexpected object.

Figure 5.9 shows the variance of the features relevant to the picking task shown in Table 5.1. In the generalization towards changed lighting conditions, changed geometry, and the failed picking tasks, significant increases in the variance can be observed. Except for the object's location in the lighting case, this increased variance is related to the object of interest. In contrast, the variance of the agent and the performed task remains low. Overall, using stochastic forward passes can identify problems during sentence generation that can hint at inexplicable and unexpected robot behavior or objects.

5.5 Conclusion

We present an approach for end-to-end language generation that combines language, vision, and control. After training, the resulting model generates a verbal task description of the robot's actions from a pre and post-action image as well as

the recorded motion trace of the robots. Using language to describe what the robot has done enables intuitive and efficient human-robot interaction and allows users to better understand what the robot is doing in case of occlusions, vision impairments, or teleportation. Empirically, the proposed methodology in this work significantly outperforms alternative methods by efficiently combining multiple modalities while also being able to generalize towards new combinations of objects and tasks. Our approach produces credible results explaining an action from a set of possible tasks and is able to detect situations in which the task execution diverges from the expected behavior by using Monte Carlo sampling. In future work, this approach could be used to not only explain successful tasks but also to describe potential failure cases to further close the gap between humans and robot collaborators.

Chapter 6

FUTURE WORK: LANGUAGE-CONDITIONED REWARD LEARNING

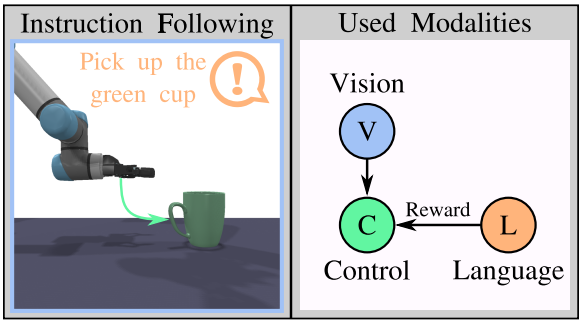

Figure 6.1: Inverse Reinforcement Learning Modalities
Overview of the used modalities in the instruction following task, trained with inverse reinforcement learning.

Natural language is an intuitive and efficient way to demonstrate new behaviors to robots. However, its integration into an imitation learning approach can be difficult due to its inherent complexity and dependency on context. In this work, Chapter 4 uses language as part of a translation process that generates a low-level control policy for the robot manipulator; however, that approach assumes that (a), the demonstrations provided by the user are ideal and (b) that language can be used in a meaningful process from words to low-level control parameters. To address both issues, this section introduces a different perspective on language integration by utilizing inverse reinforcement learning (IRL). Figure 6.1 introduces the relation of the used modalities in the same tabletop manipulation task as presented in Chapter 4. The key difference of the work presented in this chapter is that language is only used

as high-level guidance in the form of a generalizable reward function that can produce suitable rewards to the multi-task learner. Furthermore, the robot can learn an optimal policy without having to follow human demonstrations directly as demonstrations are only used to train the reward function that assigns rewards to execution traces of the learner based on a description of the desired task. In this work, we show preliminary results on learning a suitable reward function on a Mountain Car reinforcement learning problem as well as on the tabletop manipulation task proposed in Chapter 4.

6.1 Introduction

Existing inverse reinforcement learning (IRL) algorithms largely assume that each reward function represents the costs of a single task or behavior. In this chapter, we argue that a more general reward function can be learned when contextual information is considered. We present a generalization of the IRL problem that allows rewards for multiple tasks to be generated within a single, language-conditioned reward function. The major difference as compared to the approach presented in Chapter 4 is that language is not directly used for policy generation but only for selecting a task within the reward model and providing a suitable reward signal to the agent's action sequence. This allows the agent to explore the action space freely and enables it to learn an optimal control policy without being restricted by the demonstrations provided by a user. Furthermore, this approach can also be used to fine-tune an existing policy to, for example, overcome the correspondence problem between a demonstrator and the agent or learn how to recover from new states that were due to the agent's noisy control.

In this work, we explore the usage of natural language instructions provided

by the human expert to specify which agent behavior should be reinforced. We present a methodology for learning language-conditioned reward functions from multimodal demonstrations, i.e., example trajectories and verbal descriptions thereof. Once trained, a language-conditioned reward function can be used to generate a set of different agent policies that are responsive to new user input. Fundamentally, language is used to condition the reward function on the desired task and, further, to generate a reward signal that a subsequent reinforcement learning algorithm can use to train the agent. In order to generate a suitable reward function, it is of utmost importance that the learned reward model can separate the different tasks from each other but is also able to generate a meaningful reward signal for different instances of the same task. Preliminary results on two synthetic tasks, a mountain-car gym environment and a robot manipulation tasks, show that our IRL method is capable of generating meaningful rewards while clearly separating different tasks and can be utilized to successfully learn a policy on the mountain-car environment with comparable performance to a policy trained with the default reward.

6.2 Background

Reinforcement Learning (RL) is widely used to learn various tasks from grounding language (Chevalier-Boisvert *et al.*, 2019) to manipulation tasks (Misra *et al.*, 2017); however, it usually requires a large amount of feature and reward engineering. To alleviate parts of this problem, Inverse Reinforcement Learning (IRL) attempts to learn a reward function from a set of demonstrations (Abbeel and Ng, 2004; Osa *et al.*, 2018). Fundamentally, IRL can be seen as a form of Learning from Demonstration (LfD) (Hussein *et al.*, 2017) with the goal of assigning a scalar reward to a given execution trace. After training, the reward function can then be used in

a reinforcement learning pipeline to learn a policy that optimizes the reward function. Going beyond most IRL approaches that learn a reward function for a single task, the work presented in Tschiatschek *et al.* (2019) learns a single reward function that can generalize towards different instances of the same task while the approach in Chen *et al.* (2021) extends this idea by learning a generalizable reward function from in-the-wild videos of humans performing various tasks. In this work, however, we propose a method that learns a generalizable reward function from a high-level natural language description that describes one of the multiple captured tasks.

Incorporating language into the process of generating rewards is an appealing direction as language is an intuitive way for humans to describe the desired outcome of various tasks. At the same time, it allows the underlying learner to freely explore a vast action space to learn a policy that fulfills the desired action without being biased by human task demonstrations (Hussein *et al.*, 2017). Language can be used in conjunction with agent behaviors by either evaluating the final state of the task without the need of providing a full demonstration Fu *et al.* (2018); Singh *et al.* (2019), or by creating a classifier to predict whether or not a trajectory matches a verbal description of the desired task (Goyal *et al.*, 2019). In contrast to these approaches, which do not consider sharing knowledge between multiple tasks, we focus on learning a generalizable multi-task reward function for instruction following that can produce rewards for different tasks by conditioning on a natural language instruction of the desired task Stepputtis *et al.* (2020b); Anderson *et al.* (2018b). This approach has prominently been used in multi-task reward learning from videos of humans performing various tasks Chen *et al.* (2021); Fu *et al.* (2019); Schmeckpeper *et al.* (2019).

Going beyond multi-task learning, the work presented in Finn *et al.* (2017) and

Duan *et al.* (2017) can generate reward functions in a one-shot approach based on a large set of initial training tasks while Choi and Kim (2012) uses a clustering approach to rapidly learn reward functions upon observing new tasks. In this work, however, we rather focus on learning a reward function for a set of known tasks that can be conditioned on verbal descriptions of the desired task.

6.3 Problem Formulation and Approach

The goal of our approach is to learn a generalizable reward function that can be used to train a Markov Decision Process (MDP) given as tuple $\mathcal{M} := (\mathcal{S}, \mathcal{A}, \mathcal{T}, R, \gamma)$. Here, $\mathcal{S}$ and $\mathcal{A}$ are the set of states and actions, the transition function $\mathcal{T}$ is defined as $\mathcal{T} := \mathcal{S} \times \mathcal{A} \times \mathcal{S} \rightarrow [0, 1]$ and describes the probability $P(s'|s, a) = \mathbb{P}(s_{t+1} = s'|s_t = s, a_t = a)$, wile the reward function $R(s) \rightarrow \mathbb{R}$ provides a single reward at the end of each episode, and $\gamma \in [0, 1)$ is the discount factor. The goal of the MDP is to learn a policy $\pi : \mathcal{S} \times \mathcal{A} \rightarrow [0, 1]$ modeling the distribution $P(a_t|s_t)$ over actions the agent should take in any given state. An optimal policy π^* can be learned from a fully specified MDP that maximizes the agent's expected discounted reward: $\pi^* = \text{argmax}_\pi \left(\mathbb{E}_\pi \left[\sum_i \gamma_i R\left(s_i\right)\right]\right)$. However, specifying a suitable reward function $R(s)$ requires manual feature engineering and is difficult to formulate, especially when using complex or high-dimensional state representations. To mitigate this problem, IRL allows to learn a suitable reward function form a set of demonstrated behaviours.

In the IRL setting, the agent does not know the MDP's reward function and must first infer it from demonstrations $\mathcal{D} = \{d_1, ..., d_n\}$ provided by an expert. In this work, language is used together with a physical demonstrations of the desired motions and a binary reward specifying if the language matches the motion. Given this information, the goal is to learn a reward function $R(\tau, \boldsymbol{v}) \rightarrow \mathbb{R}$ taking a sequence of agent states τ

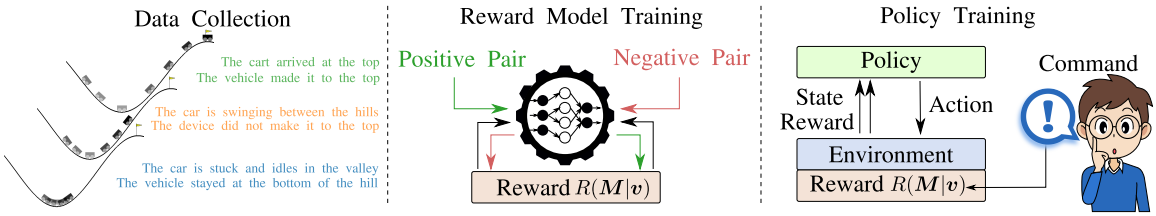

Figure 6.2: Inverse Reinforcement Learning from Language

Example of our proposed reward model. Given an observation of an agent's action and target instruction, the reward model evaluates the agents performance and produces a suitable reward.

and a verbal description of the desired outcome $\boldsymbol{v}$ to predict a scalar reward describing how well the agent's motion captured in τ matches the task described in $\boldsymbol{v}$.

6.3.1 Pre-processing Vision and Language

We first pre-process the language input to the reward model by converting the sentence s into a sequence of tokens $\boldsymbol{v}$ with the standard BERT tokenizer Devlin *et al.* (2019a). The pre-processing of the state tuple τ is done in two steps by first normalizing each dimension of each modality to a range of $[0, 1]$, which are then manually grouped into different channels based on their semantic meaning. Details of each state definition can be found in Section 6.4.1 for the *Mountain Car* experiment and Section 6.4.2 for the *Robot Manipulation* task. However, the general approach is to separate different input modalities into their own channels, e.g., agent positions and velocities are described in different channels. Especially for the *Robot Manipulation* task, our reward model contains an image sequence as part of its state definition. Each image $\boldsymbol{I} \in \mathbb{R}^{w \times h \times c}$ is recorded from an RGB top-down camera placed directly above the robot with a resolution of 569×320 and is then scaled to 128×128 pixels.

6.3.2 Reward Model

The goal of our reward model is to produce a scalar reward r from an execution trace $\boldsymbol{M}$ of the trained agent and a verbal description $\boldsymbol{v}$ of the desired task. Out model is able to utilize multiple input modalities, such as agent positions, velocities and visual representations from the environment. At its core, we use an attention mechanism that identifies the relations between multiple modalities, and we use a binary routing matrix $\boldsymbol{R}$ that defines which of the N modalities attend to each other. This allows for simplified training and limits the focus on a manually selected subset of feature relations. Especially for the robot manipulation experiment, the routing matrix significantly speeds up model training time and improves performance. An overview of our approach can be seen in Figure 6.2 in which our model learns a reward model from a positive and negative language-motion pair in the *Mountain Car* environment.

Data Preparation: In a first step, our model starts by splitting the verbal input $\boldsymbol{v}$ into individual tokens by using the BERT tokenizer and the resulting vector of tokens is used as the first modality τ_0. The agent's execution trace $\boldsymbol{M}$ is separated into its different modalities, depending on the experiments, e.g. agent positions, velocities, or environment state. See Sections 6.4.1 and 6.4.2 for experiment specific state processing. The state information is defined as a tuple $\mathcal{E} = (\boldsymbol{e}_0, \boldsymbol{e}_1, ..., \boldsymbol{e}_N)$ that contains the N modalities where $\boldsymbol{e}_0 = \tau_0$ is the verbal description. Note that each $\boldsymbol{e} \in \mathbb{R}^{s_n \times d_m}$ where s_n is the sequence length of the n'th modality.

In a first step, we convert each modality into a fixed-size feature representation with d_m dimensions by using a one-dimensional convolution for each modality: $\boldsymbol{F}_i = \text{Conv1D}(\tau_i) \forall i \in [0, ..., N]$ where each feature $\boldsymbol{F}_i \in \mathbb{R}^{s_i \times d}$ with sequence length s_i.

To recognize each feature, we add a class token $\xi_i \in \mathbb{R}1 \times d_m$ to the beginning of each sequence before adding the positional sequence embedding $\boldsymbol{P}_i \in \mathbb{R}^{s_i+1 \times d_m}$. The resulting embedding, $\boldsymbol{e}_i$, is then transferred to the routing process.

Information Routing: The purpose of information routing is to select which modalities attend to other modalities, e.g., language should attend to the robot's motion; however, having the robot's position attend to its velocity might not be as useful, especially since each added cross-modal attention increased the computational cost exponentially. The routing is defined as a binary matrix $\boldsymbol{R} \in \mathbb{R}^{N \times N}$ with elements $R_{i,j} \in \{0, 1\}$ overall N modalities. The routing matrix is used in a selection process that selects the relevant other modalities that a target modality should attend to, aggregated in the ordered set $\mathcal{C}$:

$$\mathcal{C}^n = \text{select}\left((\boldsymbol{e}_0, \boldsymbol{e}_1, ..., \boldsymbol{e}_N), \boldsymbol{R}_{n,:}\right) \forall n \in [0, ..., N] \tag{6.1}$$

A modality $\boldsymbol{e}_j$ with $0 \leq j \leq N$ will be part of $\mathcal{C}_n$ if and only if $\boldsymbol{R}_{n,j} = 1$. A given modality attends to γ_n other modalities where $\gamma_n = |\mathcal{C}_n|$, which is dependant on the number of modalities selected in $\boldsymbol{R}$ for each modality n.

Cross-Modal Attention: Cross-modal attention attends the modality $\boldsymbol{e}_\alpha$ to the other selected modalities in $\boldsymbol{C}_\alpha$. For simpler notation, we limited the following equations to highlight only one of the cross-modal attention blocks; however, the following operations are done overall N modalities.

The cross-modal attention operation is defined as follows for each attention head $h_i\mathbb{R}^{s_\alpha \times d}$ with dimension $d = d_m/H$ over H attention headers:

$$h_i = \text{CM}(\boldsymbol{Q}_\alpha, \boldsymbol{K}_\beta, \boldsymbol{V}_\beta) = \text{softmax}\left(\frac{\boldsymbol{Q}_\alpha \boldsymbol{W}_{Q_\alpha} \boldsymbol{K}_\beta^T \boldsymbol{W}_{K_\beta}}{\sqrt{d}}\right) \boldsymbol{V}_\beta \boldsymbol{W}_{V_\beta} \tag{6.2}$$

Queries $\boldsymbol{Q}_\alpha = \boldsymbol{e}_\alpha \boldsymbol{W}^Q_{\alpha\to\beta}$, keys $\boldsymbol{K}_\beta = \mathcal{C}^\alpha_\beta \boldsymbol{W}^K_{\alpha\to\beta}$ and values $\boldsymbol{V}_\beta = \mathcal{C}^\alpha_\beta \boldsymbol{W}^V_{\alpha\to\beta}$ are processed in a separate linear unit before being passed into the $\mathrm{CM}(\boldsymbol{Q}_\alpha, \boldsymbol{K}_\beta, \boldsymbol{V}_\beta)$ operation with individually trainable weights $\boldsymbol{W}$ per attention head i.

Next, the individual attention heads are concatenated together. Note, again, that these are just the attention heads for one particular $\alpha \to \beta$ pair with $\mathcal{C}^\alpha_\beta$ and each head represents the same combination, meaning that there are multiple crossmodal attention heads for each pair, that are concatenated and added to a residual connection with $\boldsymbol{e}_\alpha$. This results in $\boldsymbol{Z}_{\alpha\to\beta} = \mathrm{concat}(h_1, \ldots, h_H) + \boldsymbol{e}_\alpha$ where $\boldsymbol{Z}_{\alpha\to\beta} \in \mathbb{R}^{s_\alpha \times d_m}$. After layer normalization, we apply a two-layer feed forward network utilizing Gaussian error linear units (Hendrycks and Gimpel, 2020) such that $\boldsymbol{X}_{\alpha\to\beta} = \mathrm{FFN}(\boldsymbol{Z}_{\alpha\to\beta})$ where $\boldsymbol{X}_{\alpha\to\beta} \in \mathbb{R}^{s_\alpha \times d_m}$. Now, all cross-channels for the single modality $\boldsymbol{e}_\alpha$ over $\mathcal{C}^\alpha_\beta$ are concatenated together. Note that the dimensionality of this vector depends on the number of elements in $\gamma_i = |\mathcal{C}^i|$: $\boldsymbol{Y}_\alpha = \mathrm{concat}(\boldsymbol{X}_{(\alpha\to\beta)}, \ldots, \boldsymbol{X}_{(\alpha\to\eta)})$ where $\boldsymbol{Y}_\alpha \in \mathbb{R}^{s_\alpha \times d_m g_\alpha}$. η is used here to describe the attended modalities of $\mathcal{C}^\alpha_\beta$ the model is cross attending to. So this is $0 \leq \beta \leq \eta$.

Self Attention and Reward Prediction In a final step, we run self attention over every modality. This process is similar to the to the cross-modal attention, however, it now utilizes a single modality α for the query $\boldsymbol{Q}_\alpha = \boldsymbol{Y}_\alpha \boldsymbol{W}^Q_\alpha$, key $\boldsymbol{K}_\alpha = \boldsymbol{Y}_\alpha \boldsymbol{W}^K_\alpha$ and value $\boldsymbol{V}_\alpha = \boldsymbol{Y}_\alpha \boldsymbol{W}^V_\alpha$ pairs. Based on these input definitions, the process is the same as previously explained in the previous section. After generating $\boldsymbol{X}_\alpha$ for each modality in N, the class token ξ is pooled for each modality by selecting the first entry in each sequence and concatenated to our reward vector $\boldsymbol{r} \in \mathbb{R}^{d_m \sum R_{i,j}}$.

The vector $\boldsymbol{r}$ is then used in a two-layered feed forward network to determine the final scalar reward $r = \mathrm{MLP}(\boldsymbol{r})$ where $\mathrm{MLP}(\boldsymbol{r}) = \boldsymbol{W}_2 (\boldsymbol{W}_1 \boldsymbol{r} + \boldsymbol{b}_1) + \boldsymbol{b}_2$ with trainable

variables $\boldsymbol{W}_1, \boldsymbol{W}_2, \boldsymbol{b}_1, \boldsymbol{b}_2$.

The reward model is trained in an end-to-end approach and utilizes contrastive examples to distinguish different tasks from each other. Each training example $\mathcal{S}$ consists of the trajectory τ_a, text description $\boldsymbol{v}_a$, and the trajectory τ_b and text description $\boldsymbol{v}_b$. Our loss function is shown in equation 6.3 and describes a positive and negative loss, respectively:

$$\mathcal{L}_{p/n} = \mathbb{E}_{(\tau_a, \tau_b, \boldsymbol{v}_{a/b}) \sim \mathcal{D}} \left(-\log \left(\sigma \left(R_\Theta(\tau_{a/b}, \boldsymbol{v}_{a/b}) - R_\Theta(\tau_{b/a}, \boldsymbol{v}_{a/b}) \right) \right) \right) \tag{6.3}$$

The overall loss function is then calculated by computing the mean of the two samples $\mathcal{L} = (\mathcal{L}_p + \mathcal{L}_n)/2$. For optimization, we use the Adam optimizer with a learning rate of $1e-5$, training on a single GPU until convergence after 100 episodes. A detailed convergence plot can be seen in Figure 6.3.

6.4 Evaluation

We evaluate our approach with two different experiments to first show the feasibility of our proposed method on a gym environment before extending the approach to a more challenging environment of multi-task table-top manipulation using a 6 Degree of Freedom (DoF) robot arm. Both experiments utilize the approach described in section 6.3 with minor changes to the state space of the respective experiment.

6.4.1 Mountain Car

In the first experiment, we use the one-dimensional gym environment *Mountain Car* in which a car is located in a valley between two mountains and the task is to learn a policy that allows the car to go to the top of the mountain. A difficulty in this setup is that the car does not possess enough power to simply drive up the hill, thus an approach that swings between the two hills and using momentum is required to

successfully complete the task. We chose this experiment as an introductory example to highlight the properties of our reward model.

We collected 200 trajectories of the car attempting to ascend the hill, containing trajectories in which the car stays at the bottom of the hill, swings between the hills, and successfully reaches the top. These behaviours are grouped into three categories, where each category is defined by a threshold. We then asked ten human experts to provide descriptions for 20 trajectories for each category, where each description explains the behavior of the agent. Each collected description has been manually transcribed into text and checked for grammatical correctness and appropriateness by replacing sentences that are not relevant to the shown task. Based on these sentences, we build a template system that replaces synonyms in various sentence templates to automatically generate plausible sentences for arbitrary new execution traces.

State Definition

Our reward model is flexible with regards to the state definitions and can incorporate various modalities. For *Mountain Car*, the state τ is a list containing the tuple (c_{pos}, c_{vel}) where $c_{pso} \in \mathbb{R}$ is the car's position and $v_{vel} \in \mathbb{R}$ is its velocity. Furthermore, it contains the tuple (c_a) where $c_a \in \{0, 1, 2\}$ is a discrete representation of the car's actions. Actions are defined as accelerating to the left, right, or no accelerating at all.

Results and Ablations

Table 6.1 shows the results of our reward model in line 1 over the three different motion categories when the target behaviour is to reach the top of the mountain.

Table 6.1: Robustness of the Reward Model for *Mountain Car*

	Experiment	Described Top vs. Behavior		
		Top	Middle	Bottom
1	**Full Model**	9.40 ± 4.04	-5.92 ± 6.25	-10.91 ± 4.13
2	No Regularization	-13.12 ± 0.38	-12.68 ± 0.69	-12.34 ± 0.52
3	No Routing	9.12 ± 2.48	3.41 ± 2.77	9.61 ± 1.95
4	Verbal Generalization	8.16 ± 2.46	7.88 ± 3.75	8.26 ± 2.15
5	Motion Generalization	8.36 ± 4.05	-5.09 ± 6.63	-10.32 ± 4.34

Our reward model provides a clear positive reward of 9.4 with a variance of 4.04 for the correct behavior, while still being able to provide a distinction between the two other behaviors. While Table 6.1 only shows the target task of reaching the top of the mountain, Figure 6.3 shows the three categories introduced for mountain car, namely *Bottom*, *Middle*, and *Top*, indicating the behavior of the agent staying at the valley, moving between the mountains but not making it to the top and reaching the top of the mountain. The green violin plots show the reward and standard deviation if the agent's behavior matches the task description, while the orange plots show the average reward of the other two non-matching task descriptions. While the task of staying at the bottom and reaching the top has clear distinctions from the respective other tasks, the behavior of swinging between the hills has an overlapping decision area, resulting in a lower performance of the reward model. We attribute this loss in descriptiveness to the fact that we are using a threshold method to separate the tasks and tasks that are ending very close to the boundaries provide sub-optimal demonstrations to learn the sharp boundaries.

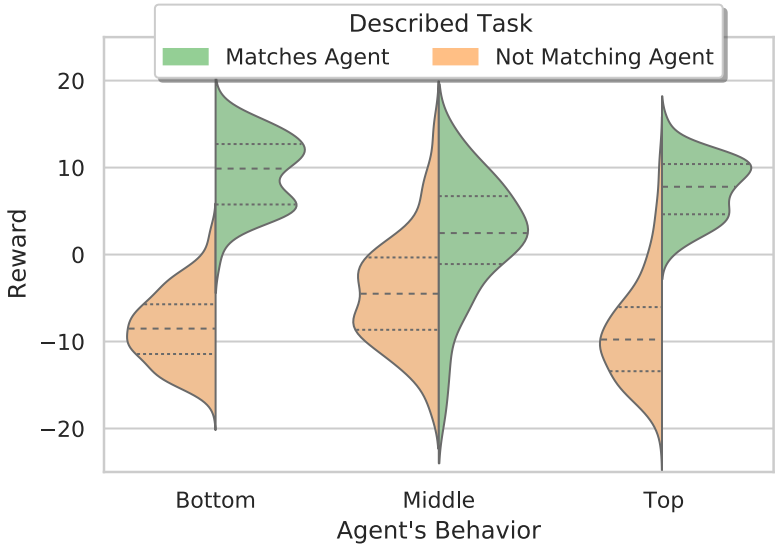

Figure 6.3: Task Separation Mountain Car

Comparison of the generated rewards for the *Mountain Car* experiment, contrasting matching and non-matching pairs of the verbal task description and the agent's behavior.

We attribute the success of our model to multiple design choices, including using a contrastive loss and information routing approach. Line 2 in Table 6.1 shows the result when using a simple mean-squared-error loss over our contrastive approach discussed in Equation 6.3, resulting in a strong negative reward overall tasks without having learned clear decision boundaries for the matching and non-matching task descriptions. Furthermore, we use a routing approach that allows us to pre-select modalities that are matched against each other in the self-attention model. While the ideal approach would be to extract information from every possible tuple combination in τ, the required computation increases exponentially for every additional modality. Instead, we introduce a routing approach to pre-select potential combinations to reduce the computational cost drastically. Line 3 in table Table 6.1 shows our reward

model's performance when attending to all possible combinations as compared to line 1 that uses the routing approach. It can be seen that limiting the number of combinations even has a positive effect on the model's performance as the *Bottom* behavior is not wrongly received a high reward.

After our reward model is trained, it is expected to generate suitable rewards for the agent that utilizes it to learn a suitable policy for the described task. A key requirement is that our reward model needs to generate a consistent reward when semantically equivalent, but different sentences are used to describe the same task; similarly, different trajectories from the same categories need to result in a similar reward. We are testing this capability by using the same motion while varying the sentence in line 4 of Table 6.1 and using the same sentence but varying motions in line 5. The results show the model's rewards for matching (Top) and non-matching behaviors (Middle and Bottom) are not descriptive anymore when varying the language, indicating that further work is needed to generalize the language properly. However, when fixing the language and providing the model with varying motions from the same categories, the results are comparable to line 1 of Table 6.1, indicating good generalization.

As our final experiment for the *Mountain Car*, we evaluate our reward model when used as part of a full reinforcement learning pipeline to show that a learned policy is not only able to converge but also to yield a policy that is successfully able to solve the *Mountain Car* experiment. We chose to train a policy with *Sarsa Lambda* and compare it to a policy trained using the standard sparse reward of the mountain car environment. Figure 6.4 shows the reward of the trained policies using our reward model and the standard reward given in the environment. Using our language conditioned reward model yields a better episodic reward signal, allowing

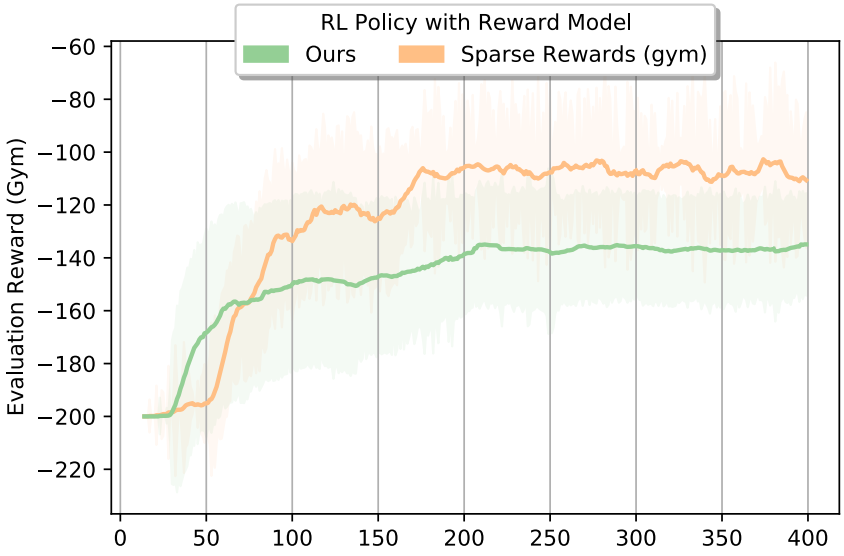

Figure 6.4: Convergence Ratio of Our Language Based Reward Model

Convergence of *Sarsa Lambda* using our reward model (green) and the standard sparse reward model of the gym environment (red). Both trained policies are considered successful based on the internal evaluation of the gym environment.

the policy to converge faster than using the sparse reward. With our model, the policy initially converges faster than the binary reward of the standard environment. However, while our final reward, as reported by the gym environment, is not as high as the one of the policy trained with a sparse reward, the environment reports a 100% success rate for both policies. It is important to note that we do not claim to produce better policies using our reward model, but only that using our language-conditioned reward model can be used to train a suitable policy for the task successfully.

6.4.2 Robot Manipulation

After having shown the feasibility of our approach in the *Mountain Car* environment, we apply our method of learning a reward function to a more challenging task.

In our robot manipulation experiment, the goal is to pick up one of the multiple cups placed in the work-space of the robot or to pour a specified quantity into one of the bowls in the environment, similar to the picking task in Chapter 4 or 5. In contrast to the previous mountain car experiment, the picking task requires control of a 6 Degree of Freedom (DoF) robot arm and an attached gripper. Furthermore, the reward model is required to understand the environment from an image $\boldsymbol{I} \in \mathbb{R}^{w \times h \times c}$ as an additional channel in τ.

This experiment is run in simulation using CoppeliaSim (Rohmer *et al.*, 2013) and uses the same dataset as the approach in Chapter 4. As a brief summary, we collected 100 human-labeled task descriptions for each of the actions from five human experts and used them to initialize an automatic, synthetic language generator. First, these descriptions are manually transcribed into text and are then converted into sentence templates and a list of possible synonyms for commonly used words. Based on these templates, we collected 22.500 further actions and automatically generated plausible descriptions for each of them based on the templating system.

State Definition

A contribution of our reward model is its ability to handle multi-modal input in its state tuple τ. For the robot experiment, the reward model needs to relate the visual input of the environment with the task described in the language command. The state space of the robot contains the robot configuration $r_{pos} \in \mathbb{R}^6$, robot velocity $r_{vel} \in \mathbb{R}^6$, gripper state $r_g \in \mathbb{R}^1$ and environment image $\boldsymbol{I} \in \mathbb{R}^{w \times h \times c}$.

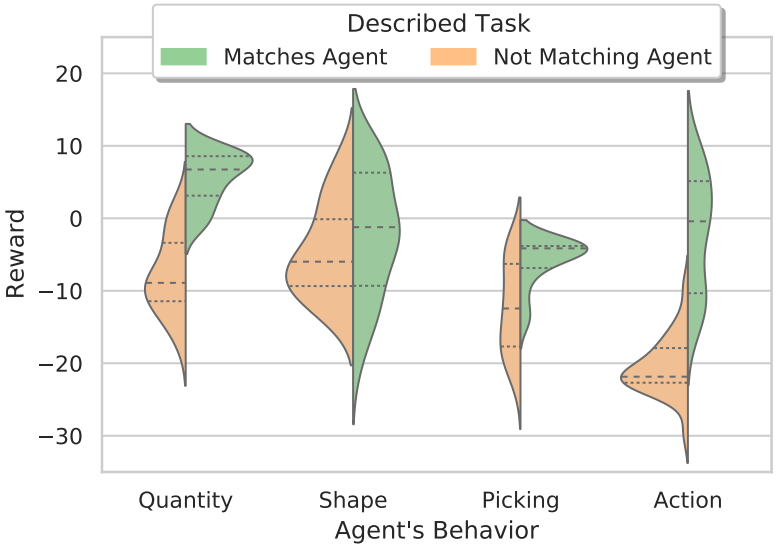

Figure 6.5: Task Separation Tabletop Manipulation

Our reward model when used on our robot manipulation task, consisting of picking and pouring actions. We compare our models ability to distinguish tasks based on the poured quantity, shape of the object, general object and general action.

6.4.3 Results on Tabletop Manipulation

In contrast to our previous task, the state space is more complex, and the routing of the various modalities is required to be feasibly trainable. Furthermore, we are distinguishing between four different tasks as defined below:

- Quantity: Pouring the correct quantity
- Shape: Pouring into an object of a different shape
- Picking: Picking the correct object from the table
- Action: Executing the correct action (picking vs. pouring)

The results of our reward model are shown in Figure 6.5. At the bottom of the

plot is the behavior the robot actually executed, while the assigned reward is on the y-axis. The green section of the respective violin plots shows the reward generated if the agent's action matches the described behavior. In contrast, the orange violin plot shows the reward assigned if the action does not match the described behavior. The results in the figure show the model's ability to distinguish different quantities, picked objects, and actions from each other; however, the reward model does not seem to be able to distinguish objects based on their shape clearly enough. We hypothesize that this is due to limitations in the vision pipeline as object differences are only visible in a small subset of pixels.

6.5 Conclusion

This work presents a novel approach to learning generalizable, language-conditioned reward functions for the mountain car environment and a tabletop manipulation task. After training, our model is able to distinguish multiple tasks from each other while providing a meaningful reward signal to a subsequent learner given a verbal description of the desired task. Preliminary experiments on the mountain car environment showed the feasibility of our approach by learning a policy that was successful in reaching the top of the mountain when employed in a full reinforcement learning pipeline. Furthermore, we showed the model's ability to distinguish different tasks in the more complex tabletop manipulation task. However, additional experiments need to be conducted to cover the entire action space of the manipulation task. Furthermore, while the results on the mountain car seem promising, it is yet to be determined if the reward model in the manipulation task can be used to learn a control policy for the robot.

Chapter 7

SUMMARY

Expanding the responsiveness and dexterity of robot manipulators by using multimodal observations and instructions stands at the core of this thesis. To this end, this work presents multiple approaches to combine language, vision, control, tactile sensing, and force/torque data to learn robust control policies while allowing for intuitive and effective training. Motivated by the human learning process, teaching robots how to manipulate objects in complex and contact-rich scenarios requires the system to fuse various sensor modalities to learn the correlations between them to complete the desired task successfully. Furthermore, many tasks can only be achieved successfully if the system understands the context it is acting in.

Instead of learning control policies from robot motion alone, additional sensor modalities allow the robot to gain additional insights about the environment it is interacting in. Starting with simple manipulation, Chapter 2 uses tactile sensors located at the robot's fingertips to perceive how a manipulated object is moving in its hand. Given this additional input, the robot learns how to utilize slippage to its advantage in order to perform dexterous manipulation. Utilizing slippage marks a significant departure from previous work. It was only seen as a negative side-effect of grasping; however, utilizing multimodal approaches allows the robot to learn additional skills that would not have been possible without it. Chapter 3 takes this idea to the next level by utilizing force/torque sensors and vision to complete a contact-rich bimanual insertion task. Similar to the previous task, force/torque and vision data are fused in order to generate the next action. However, given these

additional sensor data, the system can dynamically perform spatial and temporal adjustments that dramatically increase the success rate of the insertion.

While the previous tasks showed the advantages of using multimodal input data to increase the robot's repertoire of skills and allowed it complete tasks in contact-rich environments, conveying the context in which the system is performing a task is of utmost importance. Understanding the context of a task allows robots to physically engage with their environment in a safe and efficient manner and is an essential step towards human-robot interaction. Chapter 4 introduces an approach to learn language-conditioned visuomotor policies that can complete various tasks in a table-top manipulation setup. In this setup, language is used as an efficient and effective way to convey the desired task and context to the robot. At the intersection of language, vision, and control, the methods presented are cable to learn a single policy that is able to complete multiple tasks by using language to identify what, where, and how to perform the desired task. As a next step, Chapter 5 closes to loop to full human-robot interaction by extending the one-directional communication channel to a bidirectional channel that allows the system to explain the actions that are performed by the robots to a human partner by utilizing vision and motion traces to generate a verbal description of the performed actions. Finally, Chapter 6 introduces an alternative way to utilize language in order to learn task-specific control policies. Instead of directly translating language into robot control by using supervised learning, language is used as a high-level directive in the reward function of an inverse reinforcement learning task. This allows the system to independently explore what policies are suitable to complete the indent task.

From a learning perspective, this thesis shows how using multimodal approaches can be beneficial when learning manipulation tasks, ranging from dexterous manipu-

lation to pick-and-place and contact-rich bimanual insertion. Robots can safely and efficiently engage with their environment by utilizing these approaches, especially when adding language as an external input to convey tasks, contexts, and desired outcomes. These additional communication channels, be it an input to the system or output to a human collaborator, are an essential step towards human-robot interaction and bringing robots closer to our everyday lives.

REFERENCES

Abbeel, P. and A. Y. Ng, "Apprenticeship learning via inverse reinforcement learning", in "Proceedings of the Twenty-First International Conference on Machine Learning", ICML '04, p. 1 (Association for Computing Machinery, New York, NY, USA, 2004), URL https://doi.org/10.1145/1015330.1015430.

Abolghasemi, P., A. Mazaheri, M. Shah and L. Boloni, "Pay attention!-robustifying a deep visuomotor policy through task-focused visual attention", in "Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition", pp. 4254–4262 (2019).

Amor, H. B., O. Kroemer, U. Hillenbrand, G. Neumann and J. Peters, "Generalization of human grasping for multi-fingered robot hands", in "Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on", pp. 2043–2050 (IEEE, 2012).

Anderson, P., X. He, C. Buehler, D. Teney, M. Johnson, S. Gould and L. Zhang, "Bottom-up and top-down attention for image captioning and visual question answering", (2018a).

Anderson, P., A. Shrivastava, D. Parikh, D. Batra and S. Lee, "Chasing ghosts: Instruction following as bayesian state tracking", in "Advances in Neural Information Processing Systems 32", pp. 369–379 (Curran Associates, Inc., 2019).

Anderson, P., Q. Wu, D. Teney, J. Bruce, M. Johnson, N. Snderhauf, I. Reid, S. Gould and A. van den Hengel, "Vision-and-language navigation: Interpreting visually-grounded navigation instructions in real environments", (2018b).

Antol, S., A. Agrawal, J. Lu, M. Mitchell, D. Batra, C. Lawrence Zitnick and D. Parikh, "Vqa: Visual question answering", in "Proceedings of the IEEE international conference on computer vision", pp. 2425–2433 (2015).

Argall, B. D., S. Chernova, M. Veloso and B. Browning, "A survey of robot learning from demonstration", Robotics and autonomous systems **57**, 5, 469–483 (2009).

Campbell, J., S. Stepputtis and H. Ben Amor, "Probabilistic multimodal modeling for human-robot interaction tasks", in "Robotics: Science and Systems", (2019).

Cavallo, A., G. De Maria, C. Natale and S. Pirozzi, "Slipping detection and avoidance based on kalman filter", Mechatronics **24** (2014).

Chakraborti, T., A. Kulkarni, S. Sreedharan, D. E. Smith and S. Kambhampati, "Explicability? legibility? predictability? transparency? privacy? security? the emerging landscape of interpretable agent behavior", Proceedings of the International Conference on Automated Planning and Scheduling **29**, 1, 86–96, URL https://ojs.aaai.org/index.php/ICAPS/article/view/3463 (2021).

Chalodhorn, R., D. B. Grimes, K. Grochow and R. P. Rao, "Learning to walk through imitation.", in "IJCAI", vol. 7, pp. 2084–2090 (2007a).

Chalodhorn, R., D. B. Grimes, K. Grochow and R. P. N. Rao, "Learning to walk through imitation", in "Proceedings of the 20th International Joint Conference on Artifical Intelligence", IJCAI'07, p. 20842090 (Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2007b).

Chang, J., N. Kumar, S. Hastings, A. Gokaslan, D. Romeres, D. Jha, D. Nikovski, G. Konidaris and S. Tellex, "Learning deep parameterized skills from demonstration for re-targetable visuomotor control", (2021).

Chatzilygeroudis, K., B. Fichera, I. Lauzana, F. Bu, K. Yao, F. Khadivar and A. Billard, "Benchmark for bimanual robotic manipulation of semi-deformable objects", IEEE Robotics and Automation Letters **5**, 2, 2443–2450 (2020).

Chen, A. S., S. Nair and C. Finn, "Learning generalizable robotic reward functions from "in-the-wild" human videos", (2021).

Chevalier-Boisvert, M., D. Bahdanau, S. Lahlou, L. Willems, C. Saharia, T. H. Nguyen and Y. Bengio, "Babyai: A platform to study the sample efficiency of grounded language learning", (2019).

Chitnis, R., S. Tulsiani, S. Gupta and A. Gupta, "Efficient bimanual manipulation using learned task schemas", (2020).

Choi, J. and K.-e. Kim, "Nonparametric bayesian inverse reinforcement learning for multiple reward functions", in "Advances in Neural Information Processing Systems", edited by F. Pereira, C. J. C. Burges, L. Bottou and K. Q. Weinberger, vol. 25 (Curran Associates, Inc., 2012), URL `https://proceedings.neurips.cc/paper/2012/file/140f6969d5213fd0ece03148e62e461e-Paper.pdf`.

Chu, X., H. Fleischer, N. Stoll, M. Klos and K. Thurow, "Application of dual-arm robot in biomedical analysis: Sample preparation and transport", in "2015 IEEE International Instrumentation and Measurement Technology Conference (I2MTC) Proceedings", pp. 500–504 (2015).

Cirillo, A., P. Cirillo, G. D. Maria, C. Natale and S. Pirozzi, "Control of linear and rotational slippage based on six-axis force/tactile sensor", in "2017 IEEE International Conference on Robotics and Automation (ICRA)", pp. 1587–1594 (2017).

Codevilla, F., M. Müller, A. Dosovitskiy, A. López and V. Koltun, "End-to-end driving via conditional imitation learning", 2018 IEEE International Conference on Robotics and Automation (ICRA) pp. 1–9 (2018).

Cornia, M., M. Stefanini, L. Baraldi and R. Cucchiara, "Meshed-memory transformer for image captioning", (2020).

Cui, Y., Q. Zhang, A. Allievi, P. Stone, S. Niekum and W. B. Knox, "The empathic framework for task learning from implicit human feedback", (2020).

Cutkosky, M. R. and J. Ulmen, *Dynamic Tactile Sensing*, pp. 389–403 (Springer International Publishing, 2014).

Dafle, N. C., A. Rodriguez, R. Paolini, B. Tang, S. S. Srinivasa, M. Erdmann, M. T. Mason, I. Lundberg, H. Staab and T. Fuhlbrigge, "Extrinsic dexterity: In-hand manipulation with external forces", in "Robotics and Automation (ICRA), 2014 IEEE International Conference on", pp. 1578–1585 (IEEE, 2014).

Dauphin, Y. N., A. Fan, M. Auli and D. Grangier, "Language modeling with gated convolutional networks", (2016).

Devlin, J., M.-W. Chang, K. Lee and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding", (2019a).

Devlin, J., M.-W. Chang, K. Lee and K. Toutanova, "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding", in "Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)", pp. 4171–4186 (Association for Computational Linguistics, Stroudsburg, PA, USA, 2019b), URL `http://aclweb.org/anthology/N19-1423`.

Dillmann, R. and H. Friedrich, "Programming by demonstration: A machine learning approach to support skill acquision for robots", in "International Conference on Artificial Intelligence and Symbolic Mathematical Computing", pp. 87–108 (Springer, 1996).

Ding, Y., C. Florensa, M. Phielipp and P. Abbeel, "Goal-conditioned imitation learning", Advances in Neural Information Processing Systems URL `http://arxiv.org/abs/1906.05838` (2019).

Duan, Y., M. Andrychowicz, B. C. Stadie, J. Ho, J. Schneider, I. Sutskever, P. Abbeel and W. Zaremba, "One-shot imitation learning", (2017).

Fang, K., Y. Zhu, A. Garg, S. Savarese and L. Fei-Fei, "Dynamics learning with cascaded variational inference for multi-step manipulation", (2020).

Finn, C., T. Yu, T. Zhang, P. Abbeel and S. Levine, "One-shot visual imitation learning via meta-learning", (2017).

Fu, J., A. Korattikara, S. Levine and S. Guadarrama, "From language to goals: Inverse reinforcement learning for vision-based instruction following", (2019).

Fu, J., A. Singh, D. Ghosh, L. Yang and S. Levine, "Variational inverse control with events: A general framework for data-driven reward definition", in "Advances in Neural Information Processing Systems", edited by S. Bengio, H. Wallach,

H. Larochelle, K. Grauman, N. Cesa-Bianchi and R. Garnett, vol. 31 (Curran Associates, Inc., 2018), URL https://proceedings.neurips.cc/paper/2018/file/c9319967c038f9b923068dabdf60cfe3-Paper.pdf.

Gal, Y., *Uncertainty in Deep Learning*, Ph.D. thesis, University of Cambridge (2016).

Gal, Y. and Z. Ghahramani, "Dropout as a bayesian approximation: Representing model uncertainty in deep learning", (2016).

Geng, Z., F. Yang, X. Chen and N. Wu, "Gaussian process based modeling and experimental design for sensor calibration in drifting environments", Sensors and Actuators B: Chemical **216**, 321–331, URL https://doi.org/10.1016/j.snb.2015.03.071 (2015).

Gopalan, N., D. Arumugam, L. Wong and S. Tellex, "Sequence-to-sequence language grounding of non-markovian task specifications", in "Proceedings of Robotics: Science and Systems", (Pittsburgh, Pennsylvania, 2018).

Goyal, P., S. Niekum and R. J. Mooney, "Using natural language for reward shaping in reinforcement learning", (2019).

Hackett, D., J. Pippine, A. Watson, C. Sullivan and G. Pratt, "The darpa autonomous robotic manipulation (arm) program: A synopsis", Autonomous Robots **36** (2014).

Hart, S. G., "Nasa-task load index (nasa-tlx); 20 years later", Proceedings of the Human Factors and Ergonomics Society Annual Meeting **50**, 9, 904–908, URL https://doi.org/10.1177/154193120605000909 (2006).

Hart, S. G. and L. E. Staveland, "Development of nasa-tlx (task load index): Results of empirical and theoretical research", in "Human Mental Workload", edited by P. A. Hancock and N. Meshkati, vol. 52 of *Advances in Psychology*, pp. 139–183 (North-Holland, 1988), URL https://www.sciencedirect.com/science/article/pii/S0166411508623869.

Hendrycks, D. and K. Gimpel, "Gaussian error linear units (gelus)", (2020).

Hicks, D. J. and R. Simmons, "The national robotics initiative: A five-year retrospective", IEEE Robotics Automation Magazine **26**, 3, 70–77 (2019).

Huang, D., H. Zhan and C. Yang, "Impedance model-based optimal regulation on force and position of bimanual robots to hold an object", **2020**, 1–13, URL https://doi.org/10.1155/2020/3561807 (2020a).

Huang, G., B. Pang, Z. Zhu, C. Rivera and R. Soricut, "Multimodal pretraining for dense video captioning", in "AACL", (2020b).

Hussein, A., M. M. Gaber, E. Elyan and C. Jayne, "Imitation learning: A survey of learning methods", ACM Comput. Surv. **50**, 2, URL https://doi.org/10.1145/3054912 (2017).

Ijspeert, A. J., J. Nakanishi, H. Hoffmann, P. Pastor and S. Schaal, "Dynamical movement primitives: learning attractor models for motor behaviors", Neural computation **25**, 2, 328–373 (2013).

James, S., M. Freese and A. J. Davison, "Pyrep: Bringing v-rep to deep robot learning", arXiv preprint arXiv:1906.11176 (2019).

Jara, C. A., J. Pomares, F. A. Candelas and F. Torres, "Control framework for dexterous manipulation using dynamic visual servoing and tactile sensors feedback", Sensors **14**, 1, 1787–1804 (2014).

Javaid, M. and V. Estivill-Castro, "Explanations from a robotic partner build trust on the robots decisions for collaborative human-humanoid interaction", Robotics **10**, 1, URL `https://www.mdpi.com/2218-6581/10/1/51` (2021).

Jurafsky, D. and J. H. Martin, *Speech and language processing : an introduction to natural language processing, computational linguistics, and speech recognition* (Pearson Prentice Hall, Upper Saddle River, N.J., 2009), URL `http://www.amazon.com/Speech-Language-Processing-2nd-Edition/dp/0131873210/ref=pd_bxgy_b_img_y`.

Kim, H., Y. Ohmura and Y. Kuniyoshi, "Transformer-based deep imitation learning for dual-arm robot manipulation", (2021).

Kingma, D. and J. Ba, "Adam: A method for stochastic optimization", International Conference on Learning Representations (2015).

Krantz, J., E. Wijmans, A. Majumdar, D. Batra and S. Lee, "Beyond the nav-graph: Vision-and-language navigation in continuous environments", in "Computer Vision – ECCV 2020", edited by A. Vedaldi, H. Bischof, T. Brox and J.-M. Frahm, pp. 104–120 (Springer International Publishing, Cham, 2020).

Kress-Gazit, H., G. E. Fainekos and G. J. Pappas, "Translating structured english to robot controllers", Advanced Robotics **22**, 12, 1343–1359 (2008).

Kress-Gazit, H., G. E. Fainekos and G. J. Pappas, "Temporal-logic-based reactive mission and motion planning", IEEE transactions on robotics **25**, 6, 1370–1381 (2009).

Kuo, Y.-L., B. Katz and A. Barbu, "Deep compositional robotic planners that follow natural language commands", 2020 IEEE International Conference on Robotics and Automation (ICRA) pp. 4906–4912 (2020).

Le, T.-H.-L., A. Maslyczyk, J.-P. Roberge and V. Duchaine, "A highly sensitive multimodal capacitive tactile sensor", International Conference on Robotics and Automation (2017).

Liu, R. and X. Zhang, "A review of methodologies for natural-language-facilitated humanrobot cooperation", International Journal of Advanced Robotic Systems **16**, 3, 1729881419851402, URL `https://doi.org/10.1177/1729881419851402` (2019).

Lopatovska, I., K. Rink, I. Knight, K. Raines, K. Cosenza, H. Williams, P. Sorsche, D. Hirsch, Q. Li and A. Martinez, "Talk to me: Exploring user interactions with the amazon alexa", Journal of Librarianship and Information Science p. 096100061875941 (2018).

Lu, J., D. Batra, D. Parikh and S. Lee, "Vilbert: Pretraining task-agnostic visiolinguistic representations for vision-and-language tasks", in "Advances in Neural Information Processing Systems 32", pp. 13–23 (Curran Associates, Inc., 2019).

Lynch, C. and P. Sermanet, "Language conditioned imitation learning over unstructured data", Robotics: Science and Systems URL `https://arxiv.org/abs/2005.07648` (2021).

Ma, Y., Y. Huang, L. Mao, P. Liu, C. Liu and Y. Ge, "Pre-sliding detection in robot hand grasping based on slip-tactile sensor", in "2015 IEEE International Conference on Robotics and Biomimetics (ROBIO)", pp. 2603–2608 (2015).

Maeda, G., M. Ewerton, R. Lioutikov, H. B. Amor, J. Peters and G. Neumann, "Learning interaction for collaborative tasks with probabilistic movement primitives", in "Humanoid Robots (Humanoids), 2014 14th IEEE-RAS International Conference on", pp. 527–534 (IEEE, 2014).

Martelaro, N., V. C. Nneji, W. Ju and P. Hinds, "Tell me more designing hri to encourage more trust, disclosure, and companionship", in "2016 11th ACM/IEEE International Conference on Human-Robot Interaction (HRI)", pp. 181–188 (2016).

Matuszek, C., "Grounded language learning: Where robotics and nlp meet", in "Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI-18", pp. 5687–5691 (International Joint Conferences on Artificial Intelligence Organization, 2018), URL `https://doi.org/10.24963/ijcai.2018/810`.

Meyer, E., *The culture map: breaking through the invisible boundaries of global business* (Public Affairs, 2014).

Misra, D., J. Langford and Y. Artzi, "Mapping instructions and visual observations to actions with reinforcement learning", (2017).

Motoda, T., D. Petit, W. Wan and K. Harada, "Bimanual shelf picking planner based on collapse prediction", (2021).

Mülling, K., J. Kober, O. Kroemer and J. Peters, "Learning to select and generalize striking movements in robot table tennis", Int. J. Rob. Res. **32**, 3, 263279, URL `https://doi.org/10.1177/0278364912472380` (2013).

Mülling, K., J. Kober, O. Kroemer and J. Peters, "Learning to select and generalize striking movements in robot table tennis", The International Journal of Robotics Research **32**, 3, 263–279 (2013).

Osa, T., J. Pajarinen, G. Neumann, J. A. Bagnell, P. Abbeel and J. Peters, "An algorithmic perspective on imitation learning", Foundations and Trends in Robotics **7**, 1-2, 1179, URL http://dx.doi.org/10.1561/2300000053 (2018).

Otter, D. W., J. R. Medina and J. K. Kalita, "A survey of the usages of deep learning for natural language processing", IEEE Transactions on Neural Networks and Learning Systems **32**, 2, 604–624 (2021).

Paraschos, A., C. Daniel, J. R. Peters and G. Neumann, "Probabilistic movement primitives", in "Advances in Neural Information Processing Systems", edited by C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani and K. Q. Weinberger, vol. 26 (Curran Associates, Inc., 2013), URL https://proceedings.neurips.cc/paper/2013/file/e53a0a2978c28872a4505bdb51db06dc-Paper.pdf.

Pennington, J., R. Socher and C. D. Manning, "Glove: Global vectors for word representation", in "Empirical Methods in Natural Language Processing (EMNLP)", pp. 1532–1543 (2014), URL http://www.aclweb.org/anthology/D14-1162.

Pomerleau, D. A., "Alvinn: An autonomous land vehicle in a neural network", in "Advances in neural information processing systems", pp. 305–313 (1989).

Raman, V., C. Finucane and H. Kress-Gazit, "Temporal logic robot mission planning for slow and fast actions", in "2012 IEEE/RSJ International Conference on Intelligent Robots and Systems", pp. 251–256 (IEEE, 2012).

Ren, S., K. He, R. Girshick and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks", (2015).

Rohmer, E., S. P. N. Singh and M. Freese, "Coppeliasim (formerly v-rep): a versatile and scalable robot simulation framework", in "Proc. of The International Conference on Intelligent Robots and Systems (IROS)", (2013), www.coppeliarobotics.com.

Sai, A. B., A. K. Mohankumar and M. M. Khapra, "A survey of evaluation metrics used for nlg systems", (2020).

Scarselli, F., M. Gori, A. C. Tsoi, M. Hagenbuchner and G. Monfardini, "The graph neural network model", IEEE Transactions on Neural Networks **20**, 1, 61–80 (2009).

Schaal, S., "Is imitation learning the route to humanoid robots?", Trends in cognitive sciences **3**, 6, 233–242 (1999).

Schmeckpeper, K., A. Xie, O. Rybkin, S. Tian, K. Daniilidis, S. Levine and C. Finn, "Learning predictive models from observation and interaction", (2019).

Shaw, J. S. and V. Dubey, "Design of servo actuated robotic gripper using force control for range of objects", in "2016 International Conference on Advanced Robotics and Intelligent Systems (ARIS)", pp. 1–6 (2016).

Singh, A., L. Yang, K. Hartikainen, C. Finn and S. Levine, "End-to-end robotic reinforcement learning without reward engineering", (2019).

Srivastava, N., G. Hinton, A. Krizhevsky, I. Sutskever and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting", Journal of Machine Learning Research **15**, 1929–1958, URL `http://jmlr.org/papers/v15/srivastava14a.html` (2014).

Stachowsky, M., T. Hummel, M. Moussa and H. A. Abdullah, "A slip detection and correction strategy for precision robot grasping", IEEE/ASME Transactions on Mechatronics **21**, 5, 2214–2226 (2016).

Stefanini, M., M. Cornia, L. Baraldi, S. Cascianelli, G. Fiameni and R. Cucchiara, "From show to tell: A survey on image captioning", arXiv preprint arXiv:2107.06912 (2021).

Stepputtis, S., C. Baral and H. Ben Amor, "Towards semantic policies for human-robot collaboration", in "Southwest Robotics Symposium", (2018a).

Stepputtis, S., C. Baral and H. Ben Amor, "Neural policy translation for robot control", in "Southwest Robotics Symposium", (2019).

Stepputtis, S. and H. Ben Amor, "Active slip control for in-hand object manipulation using deep predictive models", in "Workshop on Tactile Sensing for Manipulation: Hardware, Modeling, and Learning (RSS)", (2017a).

Stepputtis, S. and H. Ben Amor, "Deep predictive models for active slip control", in "Workshop on (Empirically) Data-Driven Robotic Manipulation (RSS)", (2017b).

Stepputtis, S., J. Campbell, M. Phielipp, C. Baral and H. Ben Amor, "Imitation learning of robot policies by combining language, vision and demonstration", in "Workshop on Robot Learning (NeurIPS 2019)", (2021).

Stepputtis, S., J. Campbell, M. Phielipp, S. Lee, C. Baral and H. Ben Amor, "Language-conditioned imitation learning for robot manipulation tasks", in "Advances in Neural Information Processing Systems", edited by H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan and H. Lin, vol. 33, pp. 13139–13150 (Curran Associates, Inc., 2020a).

Stepputtis, S., J. Campbell, M. Phielipp, S. Lee, C. Baral and H. Ben Amor, "Language-conditioned imitation learning for robot manipulation tasks", in "Advances in Neural Information Processing Systems", edited by H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan and H. Lin, vol. 33, pp. 13139–13150 (Curran Associates, Inc., 2020b), URL `https://proceedings.neurips.cc/paper/2020/file/9909794d52985cbc5d95c26e31125d1a-Paper.pdf`.

Stepputtis, S., Y. Yang and H. B. Amor, "Extrinsic dexterity through active slip control using deep predictive models", in "2018 IEEE International Conference on Robotics and Automation (ICRA)", pp. 3180–3185 (IEEE, 2018b).

Stern, K., "What is dynamic tactile sensing?", URL http://blog.robotiq.com/what-is-dynamic-tactile-sensing (2017).

Su, Z., K. Hausman, Y. Chebotar, A. Molchanov, G. E. Loeb, G. S. Sukhatme and S. Schaal, "Force estimation and slip detection/classification for grip control using a biomimetic tactile sensor", in "2015 IEEE-RAS 15th International Conference on Humanoid Robots (Humanoids)", pp. 297–303 (2015).

Sugita, Y. and J. Tani, "Learning semantic combinatoriality from the interaction between linguistic and behavioral processes", Adaptive Behavior **13**, 33 – 52 (2005).

Sutskever, I., O. Vinyals and Q. V. Le, "Sequence to sequence learning with neural networks", in "Advances in Neural Information Processing Systems", edited by Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence and K. Q. Weinberger, vol. 27 (Curran Associates, Inc., 2014), URL https://proceedings.neurips.cc/paper/2014/file/a14ac55a4f27472c5d894ec1c3c743d2-Paper.pdf.

Tellex, S., N. Gopalan, H. Kress-Gazit and C. Matuszek, "Robots that use language", Annual Review of Control, Robotics, and Autonomous Systems **3**, 1, 25–55, URL https://doi.org/10.1146/annurev-control-101119-071628 (2020).

Thomaz, A., G. Hoffman and M. Cakmak, *Computational Human-Robot Interaction* (Now Publishers Inc., Hanover, MA, USA, 2016).

Tschiatschek, S., A. Ghosh, L. Haug, R. Devidze and A. Singla, "Learner-aware teaching: Inverse reinforcement learning with preferences and constraints", (2019).

van Hoof, H., T. Hermans, G. Neumann and J. Peters, "Learning robot in-hand manipulation with tactile features", in "2015 IEEE-RAS 15th International Conference on Humanoid Robots (Humanoids)", pp. 121–127 (2015).

Vaswani, A., N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. u. Kaiser and I. Polosukhin, "Attention is all you need", in "Advances in Neural Information Processing Systems", edited by I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan and R. Garnett, vol. 30 (Curran Associates, Inc., 2017), URL https://proceedings.neurips.cc/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf.

Veiga, F., H. van Hoof, J. Peters and T. Hermans, "Stabilizing novel objects by learning to predict tactile slip", in "2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)", pp. 5065–5072 (2015).

Vogt, D., S. Stepputtis, B. Jung and H. Ben Amor, "One-shot learning of humanrobot handovers with triadic interaction meshes", Autonomous Robots **42**, 5, 1053–1065, publisher Copyright: © 2018, Springer Science+Business Media, LLC, part of Springer Nature. Copyright: Copyright 2018 Elsevier B.V., All rights reserved. (2018).

Wang, T., R. Liao, J. Ba and S. Fidler, "Nervenet: Learning structured policy with graph neural networks", in "ICLR", (2018).

Williams, T. and M. Scheutz, "The state-of-the-art in autonomous wheelchairs controlled through natural language: A survey", Robotics and Autonomous Systems **96**, 171–183 (2017).

Winograd, T., *Understanding Natural Language* (Academic Press, Inc., USA, 1972).

Wu, Y., A. Kirillov, F. Massa, W.-Y. Lo and R. Girshick, "Detectron2", https://github.com/facebookresearch/detectron2 (2019).

Xie, F., A. Chowdhury, M. C. D. P. Kaluza, L. Zhao, L. L. S. Wong and R. Yu, "Deep imitation learning for bimanual robotic manipulation", (2020).

Yoshino, K., K. Wakimoto, Y. Nishimura and S. Nakamura, "Caption generation of robot behaviors based on unsupervised learning of action segments", in "IWSDS", (2020).

Zhu, F., Y. Zhu, X. Chang and X. Liang, "Vision-language navigation with self-supervised auxiliary reasoning tasks", (2020).

Zhu, G., Z. Huang and C. Zhang, "Object-oriented dynamics predictor", (2018).

APPENDIX A

DISCLOSURE OF PREVIOUSLY PUBLISHED WORK

The work in Chapter 4 was previously published at the *Conference on Neural Information Processing Systems* in 2020:

> Stepputtis, S. , J. Campbell , M. Phielipp, S. Lee, C. Baral and H. Ben Amor, "Language-conditioned imitation learning for robot manipulation task" , in "Advances in Neural Information Processing System" , edited by H. Larochelle, M. Ranzato, R. Hadsell , M. F. BalcanandH. Lin, vol . 33, pp. 1313913150 (Curran Associates, Inc. , 2020b) , URL `https://proceedings.neurips.cc/paper/2020/file/9909794d52985cbc5d95c26e31125d1a-Paper.pdf`.

Furthermore, the work in Chapter 2 was previously published at the *International Conference on Robotics and Automation* in 2018:

> Stepputtis, S. , Y. Yang and H. B. Amor, "Extrinsic dexterity through active slip control using deep predictive models" , in "2018 IEEE International Conference on Robotics and Automation (ICRA)" , pp. 31803185

APPENDIX B

SOURCE CODE

The source code for the work presented in Chapter 4 can be found at `https://github.com/ir-lab/LanguagePolicies`