INTEGRATION OF ROBOTIC PERCEPTION, ACTION, AND MEMORY

A Dissertation Presented

by

LI YANG KU

Submitted to the Graduate School of the
University of Massachusetts Amherst in partial fulfillment
of the requirements for the degree of

DOCTOR OF PHILOSOPHY

September 2018

College of Information and Computer Sciences

INTEGRATION OF ROBOTIC PERCEPTION, ACTION, AND MEMORY

A Dissertation Presented

by

LI YANG KU

Approved as to style and content by:

Erik Learned-Miller, Co-chair

Rod Grupen, Co-chair

Subhransu Maji, Member

Erik Cheries, Member

James Allan, Chair
College of Information and Computer Sciences

DEDICATION

To my family, friends, and those who dance to music ♫♪

ACKNOWLEDGMENTS

This dissertation is like a hilltop cabin on the path of my lifelong journey on understanding intelligence. Here I can re-examine what I have discovered on the path and thank those who have helped me reach here. Many have led me on this journey and without their guidance I would likely still be lost in the woods.

I first thank my advisers, Rod Grupen and Erik Learned-Miller. Rod is my trip manager, he pointed me possible paths to the cabin and showed me the whole map. Many ideas in this dissertation are spawn from inspirational discussions with Rod and I learned from him how to think from the big picture. Erik is my trip guide, he made sure my every step is on solid ground and no running on slippery banks. I still remember during the early years of this trip, Erik and I will debate for hours until both of us are comfortable with every equation we wrote down. I learned from Erik the beauty of crystal clear thinking. The robots, Robonaut-2 and uBot-6, are my jeep and bike in this trip; without them, I would only be able to reach the cabin in simulation. I then thank my mentors and colleagues at NASA, Julia Badger, Philip Strawser, Jonathan Rogers, Will Baker, Vienny Nguyen, Logan Farrell, Kimberly Vana, Evan Laske, et al., who are the local guides and mechanics that help me pass through difficult terrains while keeping the jeep running. I would then like to thank previous lab members, Steve Hart, who signed me up for this trip, and Shiraj Sen, whose previous work is the bus that took me to the starting point. I also thank my committee members Subhransu Maji and Erik Cheries who gave me directions and light up the end of the path.

I am also grateful to friends and colleagues that act as fellow hikers and forest animals that accompanied me throughout this journey. Specially, I would like to thank

my housemates, Kevin Winner, Misha Badov, Myungha Jang, Keen Sung, Amanda Gentzel, Larkin Flodin, et al., my robotics labmates, Dirk Ruiken, Tiffany Liu, Scott Jordan, Takeshi Takahashi, Mike Lanighan, Eric Wilkinson, et al., my vision labmates, Tsung-Yu Lin, Aruni Roy Chowdhury, SouYoung Jin, Pia Bideau, Jong-Chi Su, Hang Su, Huaizu Jiang, Chenyun Wu, et al., my other colleagues in computer science, Kyle Wray, Luis Pineda, Garrett Bernstein, Samer Nashed, Jarrett Holtz, Pinar Ozisik, JD DeVaughn-Brown, Joe Chiu, Steve Li, et al., and my Taiwanese friends, Hsin-Fei Tu, Hsin-Ting Huang, Celia Lin, Ning-Hsuan Tseng, Ming-Che Liu, Fang-Ling Yeh, Shao-Yu Chen, et al. I would also like to thank Leeanne Leclerc, who is the cabin manager that made sure I eventually checked in, and Laurie Downey who managed my resources for this trip.

I also have to thank people that guided me prior to this trip. My undergraduate adviser and mentor, Tian-Sheuan Chang and Nelson Chang, taught me how to hike while Sheng-Jyh Wang inspired me to hike. I am also grateful to my master degree adviser, Alan Yuille, who gave me a starting point for my previous trip. I also thank my former HRL colleagues, Ryan Uhlenbrock, David Payton, Heiko Hoffmann, Ken Kim, et al., who were guides of my previous journeys.

Lastly, I thank my family, who also represents my family in this journey metaphor, for sending me letters from home throughout this trip; without their trust and support, I would not have make it this far. As a non-native speaker, I often found myself limited in how I can express my gratitude by words. I hope the analogies I made above will not make my thanks by any means less sincere or less formal. Finally, as all of my publications do, I will end my acknowledgments with the following: This material is based upon work supported under a NASA Space Technology Research Fellowship. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author and do not necessarily reflect the views of the National Aeronautics and Space Administration.

ABSTRACT

INTEGRATION OF ROBOTIC PERCEPTION, ACTION, AND MEMORY

SEPTEMBER 2018

LI YANG KU

B.Sc., NATIONAL CHIAO TUNG UNIVERSITY

M.Sc., UNIVERSITY OF CALIFORNIA, LOS ANGELES

Ph.D., UNIVERSITY OF MASSACHUSETTS AMHERST

Directed by: Professor Erik Learned-Miller and Professor Rod Grupen

In the book "On Intelligence", Hawkins states that intelligence should be measured by the capacity to memorize and predict patterns. I further suggest that the ability to predict action consequences based on perception and memory is essential for robots to demonstrate intelligent behaviors in unstructured environments. However, traditional approaches generally represent action and perception separately—as computer vision modules that recognize objects and as planners that execute actions based on labels and poses.

I propose here a more integrated approach where action and perception are combined in a memory model, in which a sequence of actions can be planned based on predicted action outcomes. In this framework, hierarchical visual features based on convolutional neural networks are introduced to capture the essential affordances. These features in different hierarchies are associated with robot controllers of corresponding kinematic subchains to support manipulation. Through learning from

demonstration, both actions and informative features in the memory model can be learned efficiently. As more demonstrations are recorded and more interactions are observed, the robot becomes more capable of predicting the consequences of actions, thus, is better at planning sequences of actions to solve tasks under different circumstances.

TABLE OF CONTENTS

LIST OF TABLES

Table **Page**

LIST OF FIGURES

Figure **Page**

CHAPTER 1

INTRODUCTION

To act autonomously in an unstructured environment, it would be beneficial for a robot to integrate its perception and past experience to handle a wide variety of situations. However, traditional approaches generally represent action and perception separately—as object models in computer vision and as action templates in robot controllers. Due to this separation, the robot can only interact with objects based on learned models when the object label is identified. Interacting based on object labels is not only vulnerable to recognition errors but also limits how past experiences can be generalized to novel situations. In the book "On Intelligence" [33], Jeff Hawkins introduces the memory-prediction framework of intelligence and proposes that intelligence should be measured by the capacity to memorize and predict patterns. This dissertation extends on this concept and proposes a memory model that integrates action and perception. With this integrated model, a robot would be capable of solving novel tasks through predicting perceptual action consequences based on memory and observation.

Figure 1.1 shows a modified conceptual diagram of the neocortex taken from the book "On Intelligence". Blocks with the same vertical positions represent neurons of the same cortex layer and arrows represent the direction of the information flow based on neuron connections. A neuron in a higher layer represents more abstract notions while a neuron in a lower layer represents simpler features. For example, visual neurons in a higher layer have larger receptive fields, represent object categories, and change slower over time. In this figure, memory regions that connect sensory

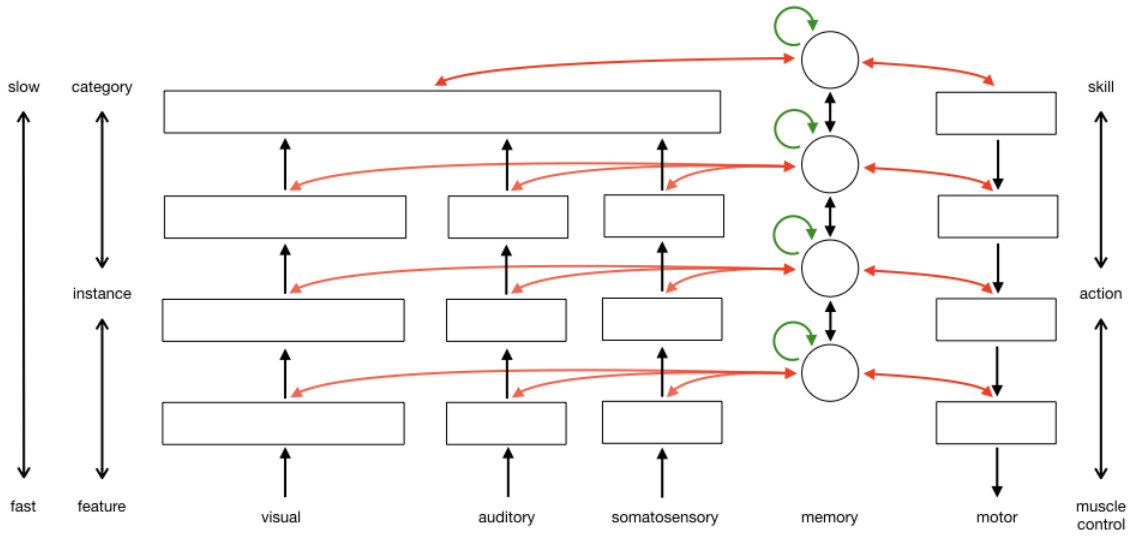

Figure 1.1. A proposed conceptual diagram of layers and connections in the neocortex.

neurons and motor neurons of the same layer is added to the original diagram. These memory regions associate neurons across modalities and can be used to infer bottom up signals that are missing. The connection loops within memory regions indicate predictions made based on observations, motor commands, and past memories. These connections not only allow motor commands to act based on memory and sensory feedback but can also explain mirroring effect of neurons that fire both when an agent acts and when it observes the same action. These memory regions have connections similar to the pyramidal neurons in the neocortex that have many connections within the same layer and an extended axon that sends signal to distant regions. However, these conjectured connections of the memory region are not based on neurological discoveries but on computational structures that shown to be practical in solving robotic tasks. In this work, a subset of this diagram is constructed and experimented on different robotic tasks.

This dissertation demonstrates that integrated memory models of action and perception can be learned efficiently from demonstrations and be used to complete tasks

under different situations. There are three major contributions in this dissertation: 1) a memory model that fuses perception and action information, 2) a hierarchical aspect representation that can be associated with controllers of different kinematic subchains, and 3) an approach that learns memory models and aspect representations from demonstrations efficiently. I describe them in the following.

1.1 Memory Model

In computer vision, there are two common types of object models used for identification. One represents objects in 2D and the other in 3D. However, neither of these incorporates information regarding how perceptions of objects change in response to actions. A robot that recognizes objects with traditional models knows nothing more than the label of the object. It is clear that humans have a different kind of object understanding—they can often predict the state and appearance of an object after an action. Incorporating actions into object models allows robots to gather information by interacting with objects and predict action outcomes. Instead of an independent object recognition system, I propose an integrated model called *aspect transition graph* (ATG) that fuses information acquired from sensors and robot actions to achieve better recognition and understanding of the environment. An ATG is a memory model that memorizes past experiences on how actions change *aspects*, observations stored in the model, and thus, maps observable states and actions to predicted future observable states.

1.2 Hierarchical Aspect Representation

An *aspect* is defined as an observation that is memorized by the robot. This concept is inspired by experiments done in the field of human psychophysics and neurophysiology, which suggest that humans memorize a set of canonical views of an object instead of maintaining a single object-centered model [17] [8]. In an ATG

model, an aspect representation that captures the affordance of the environment is crucial for executing actions robustly and generalizing to different situations.

Convolutional neural networks (CNNs) have attracted a great deal of attention in the computer vision community and have outperformed other algorithms on many benchmarks. However, applying CNNs to robotics applications is non-trivial for two reasons. First, collecting the quantity of robot data typically required to train a CNN is extremely difficult. Second, the final output of a CNN contains little location information from the observed object, which is essential for grasping. Based on the hierarchical nature of CNNs, the proposed *hierarchical CNN feature* captures the hierarchical support relations between filters in different CNN layer. This work demonstrates that by extracting these features from CNNs trained on ImageNet [76], a mapping from these features to grasp configurations of the robot hand/arm can be learned from a small set of grasping examples and generalize across different objects of similar shapes. In addition, the 3D positions of such features can be identified by tracing activations of high-level filters backwards in the CNN to discover the locations of important structures that can direct robot control. By associating features in different CNN layers with controllers that engage different kinematic subchains in the hand/arm systems and combining haptic information, a hierarchical aspect representation that supports manipulation and captures the essential affordances of an object can be built.

1.3 Learning From Demonstration

A robot that plans actions based on ATG models can only solve a task if ATGs related to the task exists in memory. While creating ATGs that represent tasks manually is tedious and not accurate in real environments, learning them from random exploration also has a small chance of success in high dimensional spaces. Learning from demonstration (LfD) is an appealing approach to teach robots new tasks due

to its similarity to how humans teach each other. However, most work on LfD has focused on learning the demonstrated motion, action constraints, and/or trajectory segments and has assumed that object labels and poses can be identified correctly. This assumption may be warranted in well-structured industrial settings, but does not hold, in general, for the kinds of uncertainty and variability common in everyday human environments.

This dissertation presents an approach for learning ATG models and its aspect representation from demonstrations efficiently. The proposed approach treats identifying informative sensory features as part of the learning process. This gives the robot the capacity to manipulate objects without fiducial markers and to learn actions focused on salient parts of the object. Instead of defining actions as relative movements with respect to the object pose, actions in ATG models are based on features that represent meaningful sensory milestones. With additional guidance provided by the operator, the informative features specific to an object instance can be identified automatically. I show that a challenging tool use task—tightening a bolt using a ratchet—can be learned from a small set of demonstrations. The proposed approach learns what part of the ratchet should be aligned with the bolt by recognizing consistent spatial relations between features among a set of demonstrations.

1.4 Document Overview

The document is organized as follows. Chapter 2 provides background on object representation and related work on grasping, error detection, convolutional neural network, learning from demonstration, etc. Chapter 3 provides an introduction to the proposed memory model, a discussion on how it can be used to handle uncertainty, and an example on how open loop and closed loop controllers can be combined in this model. In Chapter 4, a hierarchical aspect representation based on the proposed hierarchical CNN features is introduced and how these features can be associated

with controllers of different layers is described. Chapter 5 explains how the proposed memory model combined with the hierarchical aspect representation can be learned from demonstrations efficiently. Experimental results on drill grasping and bolt tightening based on this framework are analyzed. In Chapter 6, I draw connections from this proposed framework to the proposed conceptual diagram of the neocortex and discuss future directions of this work.

CHAPTER 2

BACKGROUND

In this chapter, research on object representation in human brains and background on aspect transition graphs (ATG) are first discussed. Different brain models are then compared to the ATG. Related work on affordance and how the ATG can be used to model actions that an object affords are further explained in Section 2.3. In Section 2.4 and 2.5, background on belief space planning, error detection, and surprise is described. In Section 2.6, 2.7, and 2.8, related work in vision-based grasping, visual servoing, and convolutional neural networks is compared. Lastly, background on plan generalization and learning from demonstration is discussed in Section 2.9 and 2.10.

2.1 Object Representation

In human psychophysics and neurophysiology, the study of visual object recognition is often motivated by the question of how humans recognize 3-D objects while receiving only 2-D light patterns on the retina [90]. Two types of models for object recognition have been proposed to answer this question. The structural description model represents each object by a small number of view-invariant primitives and their position in an object-centered reference frame [62]. Alternatively, viewer centered models represent each object as collections of viewpoint-specific local features. Since the development of these models, experiments in human psychophysics and neurophysiology have provided converging evidence for viewer centered models. In experiments done by Edelman and Bülthoff [17] [8], it is shown that when a new object is presented to a human subject, a small set of canonical views are formed

despite the fact that each viewpoint is presented to the subject for the same amount of time. Experiments on monkeys done by Logothetis et al. further confirm that a significant percentage of neurons in the inferior temporal cortex respond selectively to a subset of views of a known object [57]. However, how an infinite set of possible views can be effectively reduced to a smaller set of canonical views remains an open question. Different approaches such as view interpolation introduced by Poggio [73] and linear combinations of views introduced by Ullman [92] have been proposed.

Closely related to the viewer centered models in the field of psychophysics, aspect graphs are first introduced by Koenderink and Van Doorn as a way to represent 3-D objects using multiple 2-D views in the field of computer vision [45]. An aspect graph contains distinctive views of an object captured from a view sphere centered on the object. Research on aspect graphs have focused on the methodologies for automatically computing aspect graphs of polyhedra and general curved objects [26] [47]. The set of viewpoints on the view sphere is partitioned into regions that have the same qualitative topological structure as an image of the geometric contours of the object. However, work done in this field is mostly theoretical and is not applicable in real practice as discussed by Faugeras et al. [18]. One of the difficulties faced in this work concerned the large number of aspects that exist for normal everyday objects. An object can generate millions of different aspects, but many of these may be irrelevant at the scale of the observation. In this work, a consistent treatment for segmenting observations into aspects within a practically-sized subset of all possible aspects for most types of objects including deformable objects is proposed.

The Aspect Transition Graph (ATG) representation is an extension of these concepts. In addition to distinctive views, an ATG summarizes how actions change viewpoints or the state of the object and thus, the observation. Besides visual sensors, extensions to tactile, auditory and other sensors also become possible with this representation. ATGs were first introduced in Sen's work [81] as an efficient way of

storing knowledge of objects hierarchically. Sen's ATG was composed of nodes that are represented by dynamic controller statuses. The action that lead from one node to another is determined based on the differences in controller statuses implicitly. This work redefines aspect transition graph (ATG) as a directed *multigraph*, composed of a set of aspect nodes connected by a set of action edges that capture the probabilistic transition between aspect nodes. An aspect is defined as a multi-feature observation that is stored in the model and an action edge represents an action that transitions between aspect nodes. An ATG is a viewer centered model that summarizes empirical observations of aspect transitions in the course of interaction. This new ATG definition is more general and can model uncertainty and integrate with belief space planning more easily.

2.2 Brain Theories

The Memory-Prediction framework, a brain model that is consistent with neurological discoveries, is proposed by Hawkins in his book "On intelligence" [33]. This model emphasizes prediction from sequence memory based on the observation that humans recognize quotations and songs based on their sequences stored in memory. George and Hawkins further propose the Hierarchical Temporal Memory model that gives the Memory-Prediction framework mathematical foundations under Bayesian terms [23]. Lee and Mumford also suggests that based on findings on the early visual cortex activation, particle filtering and Bayesian-belief propagation algorithms might be used in cortical computations [52]. In this work, the concept of sequence memory is extended to recognizing objects. The relationship between a sequence of actions and a sequence of views are modeled not only to recognize objects, but also to provide robots with the capability to plan actions based on prediction.

2.3 Affordance

The term affordance first introduced by Gibson has many interpretations, I prefer the definition of affordance as "the opportunities for action provided by a particular object or environment" [25]. Affordance can be used to explain how the "value" or "meaning" of things in the environment is perceived. The proposed framework is based on this interactionist view of perception and action that focus on learning relationships between objects and actions specific to the robot. Some recent work in computer vision and robotics extended this concept of affordance and applied it to object classification and object manipulation [29] [44] [87]. Affordances can be associated with parts of an object, such as the work done by Varadarajan et al. where predefined base affordances are associated with surface types [95] [94]. In this work, models that inform inference in an extension of Gibson's original ideas about direct perception is built [24] [27].

Affordance describes the interaction between an agent and an object or environment. The proposed ATG is an affordance-based representation that is grounded in the robot's own actions and perceptions. Instead of defining object affordances from a human perspective, they are learned through direct interaction with objects from the robot's perspective. This work demonstrates that by exploiting these learned ATGs the robot can recognize objects and manipulate them to reach goal states.

2.4 Belief Space Planning

Planning based on belief is introduced by Sondik and Smallwood for solving the optimal control problem characterized by the Partially Observable Markov Decision Processes (POMDPs) [85] [84]. The value iteration algorithm for solving POMDP is further improved by many authors such as [42] and [68] to solve larger problems. However, the maximum number of states these algorithms can handle is still largely

restricted. Some recent work [70] [65] [55] in computing optimal solutions for the POMDP problem have focused on solving this problem in Gaussian belief spaces where beliefs are modeled as Gaussian distributions. However, Gaussian distributions are not suitable for modeling beliefs in problems where states are defined as aspects of objects. In [72] a sample-based approach to belief space planning is introduced to handle non-Gaussian belief state. In this work, the non-Gaussian beliefs for all states are propagated through a history of observation. The belief update is simplified by using one collection state that represents all unknown states for each object model. One of the difficulties in belief space planning is to predict future observations. In work done by Platt et al., future observation is modeled based on Gaussian distribution under the assumption that the maximum likelihood observation is always obtained [70]. In this work, the probability of an observation is estimated based on past observations stored in ATG models. The prediction gets more accurate as more information is memorized.

2.5 Error Detection and Surprise

In the work done by Rodriguez et al. [74] a classifier is used to predict whether a grasp is a successful grasp for completing a task based on haptic feedback. If it is classified as a failed grasp the robot aborts and retries. The introduced ATG model can also be used to detect errors based on a similar concept, but instead of running a classifier at a specific step, a general framework that constantly checks if the observation is within expectations is introduced. In research done by Donald [16], a theory for error detection and recovery strategies based on geometry and physical reasoning are introduced. In this work, error detection and recovery is based on an observation-based model of the environment.

Baldi introduces a computational theory of surprise where surprise is defined by the relative entropy between the prior and the posterior distribution of an observer

[4]. This formula is shown to be consistent with what attracts human gaze in natural video stimuli [39]. However, this theory of surprise would identify informative robot actions that reduces the entropy over models significantly as surprising. In this work, a simpler definition that agrees better with intuition is proposed.

Behaviors based on intrinsic motivations have been well studied in the field of psychology. The concept of "drives" such as hunger, pain, sex, or escape in human behavior has been introduced by Hull [37] and applied to manipulation [31] and explorations [59]. Berlyne further proposed a number of other intrinsically motivating factors such as novelty, habituation, curiosity, surprise, challenge, and incongruity [6]. In this work, surprise is used as an intrinsic motivator to learn new models of the environment.

2.6 Grasping Based on Vision

A lot of work has been done on generating robotic grasp plans from visual information. In work done by Saxena et al., a single grasp point is identified using a probabilistic model on a set of visual features such as edges, textures, and colors [78]. Similar work uses contact, center of mass, and force closure properties based on point cloud and image information to calculate the probability of a hand configuration successfully grasping a novel object [79]. Platt et al. uses online learning to associate different types of grasps with the object's height and width [71]. A shape template approach for grasping novel objects was also proposed by Herzog et al. [34]. A shape descriptor called a height map that captures local object geometry is used for matching part of a point cloud generated by a novel object to a known grasp template. Another work uses a geometric approach for grasping novel objects based on point clouds [63]. An antipodal grasp is determined by finding cutting planes that satisfy geometric constraints. A similar approach based on local object geometry was also introduced [104]. In the work done by Lenz et al., a deep network trained on

1035 examples is used to determine a successful grasp based on RGB-D data [53]. Grasp positions are exhaustively searched and evaluated. In this work, the proposed hierarchical CNN features are associated to a known grasp and both the local pattern and the higher level structure are considered. This approach localizes features in a pre-trained convolutional neural network and can generate grasp points based on a small set of grasping examples.

2.7 Visual Servoing

Visual servoing can be classified into two major types: position-based servoing, where servoing is based on the estimated pose; and image-based servoing, where servoing is based directly on visual features [38]. The image-based servoing approach has the advantage that it performs with an accuracy independent of extrinsic camera calibration and does not require an accurate model of the target object or end effector. The visual servoing approach introduced in this work uses an image-based servoing technique inspired by Jägersand and Nelson [40], in which Broyden's method is used to estimate the visuomotor Jacobian online. This work uses a similar update approach but is implemented on top of a changing set of features. Some other work in visual servoing has also investigated approaches that do not rely on a predefined set of features. In [82], a set of robust SIFT features are selected to perform visual servoing. In [35] moments of SIFT features that represent six degrees of motion are designed. An approach that is based on the image entropy was also introduced in [13]. However these approaches all assume a setting in which the camera is mounted on the end effector. In this work, the setting is more similar to human manipulation. Unlike a system where the camera is mounted on the end effector, only part of the observed features move in correspondence with the end effector. The propose visual servoing algorithm is used to guide the robot end effector, within the field of view, to a pose

that is defined relative to an object that was memorized. The features that are controllable are learned and reused.

2.8 Convolutional Neural Networks

Convolutional neural networks (CNNs) are a class of deep neural networks that contain more then one convolutional layers introduced by Lecun and Bengio [51]. In the 2012 ImageNet Large Scale Visual Recognition Challenge, the CNN based approach proposed by Krizhevsky et al. generated results that surpassed other methods by a large margin [48]. CNN based approaches have since outperformed other approaches on most benchmarks in computer vision. Several authors have also applied CNNs to robotics. In the work done by Levine et al., visuomotor policies are learned using an end-to-end neural network that takes images and outputs joint torques [56]. A three layer CNN is used without any max pooling layer to maintain spatial information. In this dissertation, multiple convolution layers are also used; but unlike the previous work, relationships between layers are used to define a feature. Finn et al. use an autoencoder to learn spatial information of features of a neural network and demonstrate that the robot can learn tasks with reinforcement learning [21]. In [20], Finn and Levine further demonstrated that robots can learn to predict the consequences of pushing objects from different orientations and execute pushing actions to reach a given object pose based on a neural network structure with nine convolutional layers. In research done by Pinto and Gupta, a CNN is used to learn what features are graspable through 50 thousand trials collected using a Baxter robot [69]. The final layer is used to select 1 out of 18 grasp orientations. CNN is also used in the deep Q-network that generates control commands for agents that play Atari games using reinforcement learning [58]. The authors show that the deep Q-network agent is able to achieve professional human gamer performance on 49 games.

Many research have been done on understanding the relationship between CNN filter activations and the input image. In the work done by Zeiler and Furgus, deconvolution is used to find what pixels activate each filter [102]. In another work, a saliency map that can be used for object localization is obtained by calculating the derivative with respect to the image [83]. An approach called guided backpropagation that adds guidance signal to obtain a better visualization of higher level filters is also introduced [86]. In this work, backpropagation is performed on a single filter per layer to consider the hierarchical relationship between filters. Recent work by Zhang et al. introduces the excitation backprop that uses a probabilistic winner-take-all process to generate attention maps for different categories [103]. This work localizes features based on similar concepts.

Some authors have explored using intermediate filter activation in addition to the the response of the output layer of a CNN. Hypercolumns, which are defined as the activation of all CNN units above a pixel, are used on tasks such as simultaneous detection and segmentation, keypoint localization, and part labelling [30]. The hierarchical CNN feature proposed in this work groups filters in different layers based on their hierarchical activation instead of just the spatial relationship. In [80], the last two layers of two CNNs, one that takes an image as input and one that takes depth as input, are used to identify object category, instance, and pose. In [97], the last layer is used to identify the object instance while the fifth convolution layer is used to determine the aspect of an object. This work considers a feature as the activation of a lower layer filter that causes a specific higher layer filter to activate and plans grasp poses based on these features.

2.9 Plan Generalization

The idea of generalizing plans can be traced back to the STRIPS-style robot planners [19], where macro operations are generated by combining primitive actions

and generalized through replacing constants with variables. Since then, temporal abstraction that combines actions of smaller time scales to abstractions of larger time scales has become an active research area in planning and reinforcement learning [89] [46]. In this work, abstraction over perception is considered instead of identifying macro operations. A lot of work have also focused on the action model learning problem, where the goal is to learn the preconditions and effects of an action given a set of observations [3] [100] [96]. However, this series of work mostly focus on solving problems in a symbolic world. Instead, the proposed approach is grounded in the observation, where preconditions and action effects are defined by visual or haptic features.

Similar approaches have also been applied on robotic tasks. In work by Wörgötter et al., generalization is done through finding plans similar to the current situation [99]. A replacement object can also be chosen based on a manually defined table of object attributes. In this dissertation, instead of finding similarities based on symbols defined in human language, generalization is done based on visual appearance.

2.10 Learning From Demonstration

Much research has focused on methods for "learning from demonstration (LfD)," in which robots acquire approximate programs for replicating solutions to sensory and motor tasks from a set of human demonstrations. In work by Calinon et al. [11] [10], Gaussian mixture models are used to model multiple demonstrated trajectories by clustering segments based on means and variances. A Gaussian mixture regression is used to generate motions for different start and goal states during execution. In work by Pastor et al. [64], dynamic movement primitives are used to generalize trajectories with different start and end point. Instead of modeling trajectories in terms of motion invariants, the learning from demonstration approach introduced in this dissertation

focuses on learning consistent perceptual feedback that provides informative guidance for actions.

Approaches that learn from multiple demonstrations often require an experienced user to show a variety of trajectories in order to estimate task information like state variables, task constraints, relative frame, etc. In work by Alexandrova et al. [2], instead of generalizing from multiple examples, the user demonstrates once and provides additional task and feature information via a user interface. This is similar to the proposed approach where the user specifies action types and informative features are identified automatically.

In the work by Phillips et al. [67], experience graphs are built from demonstration and used to speed up motion planning. A manipulation task such as approaching a door and opening it can be planned in a single stage by adding an additional dimension that represents the state of the object. However, the demonstrated tasks are restricted to cases where the object can be manipulated in a one dimensional manifold that is detectable based on the position of a single contact point. In this work, demonstrations are stored as ATG models. ATGs are directed multi-graphs composed of aspect nodes that represent observations and edges that represent action transitions. Aspect nodes represent observations directly and can, therefore, be used to model a higher dimensional space.

In the work by Akgun et al. [1], a demonstrator provides a sparse set of consecutive keyframes that summarizes trajectory demonstrations. Pérez-D'Arpino and Shah [66] also introduced C-Learn, a method that learns multi-step manipulation tasks from demonstrations as a sequence of keyframes and a set of geometric constraints. In this work, aspect nodes that contain informative perceptual feedback play a similar role as keyframes that guide the multi-step manipulation. Instead of considering geometric constraints between an object frame and the end effector frame, relations between visual features and multiple robot frames are modeled.

CHAPTER 3

MEMORY MODEL

Memorizing past experiences is crucial for a robot to plan actions in a new environment. In the book "On Intelligence" [33], Hawkins states the following *"Your brain receives patterns from the outside world, stores them as memories, and makes predictions by combining what it has seen before and what is happening now"*. This work extends on the same concept and considers an agent that interacts with the environment and memorizes the consequences of actions. As more memories are recorded and more interactions are observed, the agent becomes more capable of predicting the consequences of actions and better at planning sequences of actions to solve tasks.

Object manipulation is an essential skill for a general purpose robot, and recognizing known objects is often a first step in manipulation tasks. In computer vision and robotics, object recognition is often defined as the process of labeling segments in an image or fitting a 3-D model to an observed point cloud. The object models used to accomplish these tasks usually include information about visual appearance and shape. However, what these object recognition systems provide is merely a label for each observed object. The sequence of actions that the robot should perform based on the object label are often manually defined. Without linking actions to object labels, these object models themselves have limited utility to the robot.

In this chapter, a memory model that tightly couples perception with action is introduced. This integrated model allows the robot to act directly on observations instead of object labels and can be used to predict action outcomes based on past

experience. The advantages of this memory model is shown in partially observed and non-deterministic environments.

This chapter begins with introducing the aspect transition graph (ATG) memory model and its definition on convergence and completeness. Under what condition can a sequence of closed loop and open loop actions in an ATG model guarantee success is analyzed. In Section 3.2, the benefit of ATGs in a partially observable environment is discussed. An information theoretic planner and results of handling uncertainty with belief space planning are also described. Section 3.3 explains how visual servoing can be used to construct a funnel-slide-funnel structure in an ATG that improves action accuracy significantly. An ATG-based approach that allows the robot to interact with objects without explicit pose information is further introduced in Section 3.4. In Section 3.5, how ATG models can be used to detect and handle errors early in a non-deterministic environment is explained. An equation for determining a surprising situation is also proposed.

3.1 The Aspect Transition Graph

The aspect transition graph (ATG) explained in this section is a representation that memorizes how actions change observations of objects or the environment. It is an extension of the original concept of an aspect graph used in the field of Computer Vision [45]. In addition to distinctive views, the ATG object representation summarizes how actions change viewpoints or the state of the object and thus, the observation. The term "observation" is defined to be the combination of all sensor feedback of the robot at a particular time and the "observation space" as the space of all possible observations. This limits the representation to a specific robot, but allows the representation to present object properties other than viewpoint changes. Extensions to tactile, auditory and other sensors is possible with this representation. An ATG of an object can be used to plan manipulation actions for that object to

achieve a specific target aspect. For example, in order for the robot to pick up an object, the target aspect is a view where the robot's end effector surrounds the object. It is expected that this view will be common to many such tasks and that it can be the target outcome of a sequence of actions executed by open-loop and closed-loop controllers.

3.1.1 Definitions

Different from prior work on aspect graphs, this work defines "aspect" as a single observation that is stored in the object representation. This usage is consistent with the term "canonical view" coined in the psychophysics literature for image-based models discussed in Section 2.1. An ATG is represented using a directed *multigraph*[1] $G = (\mathcal{X}, \mathcal{U})$, composed of a set of aspect nodes $\mathcal{X}$ connected by a set of action edges $\mathcal{U}$ that capture the probabilistic transition between aspects. An action edge u is a triple (x_1, x_2, a) consisting of a source node x_1, a destination node x_2 and an action a that transitions between them. Note that there can be multiple action edges (associated with different actions) that transition between the same pair of nodes.

In contrast to aspect graphs and image-based models that differentiate views based on visual appearance, I argue that, in general, discriminating between object observations should depend on whether the actor is capable of manipulating the object such that the observation converges to a target aspect without using prior knowledge of the object. That is, aspects are determined by functions of the visual servoing and action abilities of the robot.

Figure 3.1 shows an example of an ATG that contains two aspects x_1, x_2 and one action edge u connecting the two aspects in the observation space. An aspect is represented as a single dot in the figure. The ellipses around x_1, x_2 represent the ϵ-region of the corresponding aspect. Inside the ϵ-region, the observation is close

[1] A multigraph allows multiple edges between a given pair of vertices.

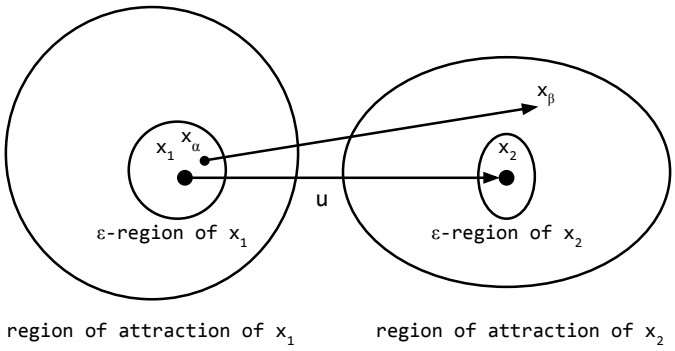

Figure 3.1. An ATG containing two aspects x_1 and x_2, each a likely result of applying a closed-loop action within their respective regions of attraction. The edge labeled u is a model-referenced open-loop action that reliably maps the ϵ-region of x_1 to the interior of the region of attraction of x_2.

to the target aspect, and the closed-loop action is considered to have "converged." The ϵ-region is task dependent; a task that requires higher precision such as picking up a needle will require a smaller ϵ-region. Each aspect x is located in the ϵ-region but does not have to be in the center. The location and shape of the ϵ-region also depends on the given task since certain dimensions in the observation space might be less relevant when performing certain tasks.

The larger ellipses surrounding the ϵ-regions are the region of attraction of a closed-loop controller referenced to aspects x_1 and x_2. Observations within the region of attraction converge to the ϵ-region of the target aspect by running this closed-loop controller that does not rely on additional information from the object model. For example, a visual servo can be implemented to perform gradient descent to minimize the observation error. The region of attraction for using such a controller is the set of observations from which a gradient descent error minimization procedure leads to the ϵ-region of the target aspect.

The arrow in Figure 3.1 that connects the two aspects is an action edge (x_1, x_2, a) that represents a transition. Action a is an open-loop controller that causes aspect

transitions. Instead of converging to an aspect, open-loop controllers tend to increase uncertainty in the observation space. An example is an end point position controller that moves to a relative pose with respect to a visual feature on an object point cloud. Under situations when there is no randomness in observation, action execution and the environment, executing action a from aspect x_1 will transition reliably to aspect x_2.

3.1.2 Convergence

The arrow in Figure 3.1 that connects the observation x_α within the ϵ-region of x_1 to observation x_β represents a scenario where action a is executed when x_α is observed in a system in which actions have stochastic outcomes. ϵ_u is defined as the maximum error between the aspect x_2 and the observation x_β when action a is executed while the current observation is within the ϵ-region of aspect x_1. ϵ_u can be caused by a combination of kinematic and sensory errors generated by the robot or randomness in the environment. If the region of attraction of the controller that converges to aspect x_2 covers the observation space within ϵ_u from x_2, by running the convergent controller it is guaranteed to converge within the ϵ-region of aspect x_2 under such an environment.

3.1.3 Funnel Slide Metaphor

Burridge et al. describe a controller as a funnel that guides the robot state to convergence; multiple controllers can be combined to funnel robot states to a desired state that no one single controller can reach alone [9] . However under certain situations the goal state may not be reachable through a combinations of controllers that act like funnels. For example, a visual servoing controller can control the end effector to a certain pose based on the robot hand's visual appearance. However, to reach the goal state, a controller that transitions from a state where the robot hand is not visible to one in which the visual servoing controller can be executed is required.

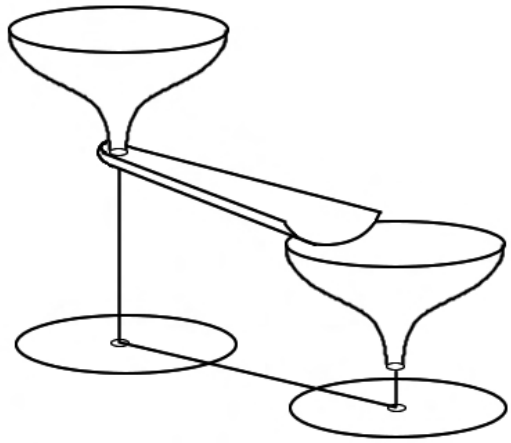

Figure 3.2. Funnel-slide-funnel structure. The funnel metaphor introduced by Burridge et al. [9] is used to describe a closed-loop controller or a track control action that converges to a subset of states, while the slide metaphor is used to describe an open-loop controller or a search control action that causes state transitions.

Such a controller can be an open-loop controller that moves the end effector to a memorized pose and may not necessarily converge to a certain state like a funnel.

The notion of a *slide* as a metaphor for the kind of action that transitions from one set of states to another is introduced in this work. Uncertainty of the state may increase after transitioning down a slide, but may still reach the goal state if a funnel-slide-funnel structure is carefully designed. As long as the end of the slide is within the region of attraction of the next funnel, convergence to the desired state can be guaranteed even when open-loop controllers are within the sequence. Figure 3.2 illustrates the funnel-slide-funnel concept using the same style of figure demonstrated by Burridge et al. [9]. In Section 3.3, the action accuracy of a funnel-slide-funnel structure where visual servoing is used as a closed-loop controller is analyzed.

3.1.4 Completeness and Sufficiency

An ATG is *complete* if the union of the regions of attraction over all aspects cover the whole observation space and a path exists between any pair of aspects. A complete ATG allows the robot to manipulate the object from any observation to one

of the aspects. Complete ATGs are informative but often hard to acquire and do not exist for irreversible actions. On the other hand, it is not always necessary to have a complete ATG to accomplish a task. For example, a robot can accomplish most drill related tasks without memorizing the bottom of the drill. Therefore, an ATG is defined to be *sufficient* if it can be used to accomplish all required tasks of the object. This work focuses on sufficient ATGs.

3.2 Handling Uncertainty With Belief Space Planning

This section describes how ATGs can be used to handle uncertainty in a partially observable environment by addressing the dual problem of modeling and reasoning. A belief space planner that takes into account the uncertainty across aspects and objects to plan efficiently is presented. The goal is to have the robot build up a set of object models by interacting with random objects one at a time. The task is evaluated based on whether the robot can identify novel objects and recognize which object model it corresponds to if it have been observed in the past. Once the object is recognized, the robot can manipulate the object to reach a goal aspect by finding the shortest path from the current aspect node to the target aspect node in its ATG model.

In this section, how objects are modeled as an ATG is first described. An information theoretic planner that picks the most informative action is then introduced. This planner with ATG object models is further tested on a dual problem of modeling and reasoning and shown to outperform a baseline approach. How this framework can be used to reach a given goal aspect is also discussed. Table 3.1 lists the symbols used in this section.

Table 3.1. Notations

Notation	Definition
x_t	the aspect at time t
z_t	the observation at time t
a_t	the control data at time t
$bel(x_t)$	$p(x_t \mid z_{1:t}, a_{1:t})$
$\overline{bel}(x_t)$	$p(x_t \mid z_{1:t-1}, a_{1:t})$
$\mathcal{M}$	the current memory pool
G_i	an ATG in memory pool, $G_i \in \mathcal{M}$
$\lvert G_i \rvert$	the number of total aspects in an ATG G_i
O_i	the object given to the robot at the ith trial
o_j	the object labeled id j
$\mathcal{O}$	the set of objects in the world, $\forall j \; o_j \in \mathcal{O}$
$\lvert \mathcal{O} \rvert$	the total number of objects in the world
$\mathcal{S}_T$	the set of objects given to the robot up to the Tth trial, $O_i \in \mathcal{S}_T \quad i = 1 \ldots T$
$\mathcal{X}_j$	the set of robot states that represents o_j
$\mathcal{U}_j$	the set of action edges in o_j
$\lvert \mathcal{F} \rvert$	the number of possible features

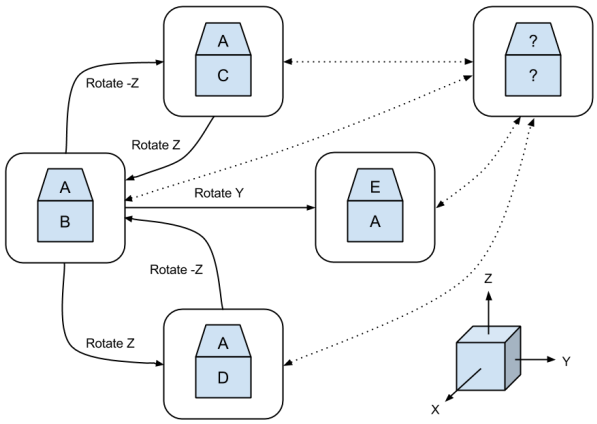

Figure 3.3. Example of an incomplete aspect transition graph (ATG) of a cube object. Each aspect is consists of observations of two faces of the cube. The lower right figure shows the coordinate of the actions and the aspect with question marks is the collection node representing all unknown aspects of the object. Each solid edge represents an observed action edge while each doted edge represents a set of unobserved action edges.

3.2.1 Modeling Objects with ATG

An object in the proposed framework is represented using an ATG where each aspect node x represents an observation from an unique view point of an object and each edge represents an action a that causes transition between aspects. z is used to denote an observation.

The memory pool $\mathcal{M}$ is defined as a set of ATGs that the robot created through past observations. Each ATG in the memory pool represents a single object given to the robot in the past. The ATG of an object is complete if it contains all possible aspect nodes and node transitions. However, in practice, when ATGs are learned through exploration they are almost always incomplete. In addition, an object might be represented by multiple (incomplete) ATGs. A complete model is more informative but harder to learn autonomously. This work focuses on handling incomplete object models. Figure 3.3 shows an example of an incomplete ATG on a cube object with a character on each face. The action edges in the ATG describe what action allows the robot to transition from one aspect node to another aspect node.

Assuming that an object has a total of $|G|$ aspects, if the robot has already observed $|\mathcal{X}|$ aspects on this object, a naive way to build an incomplete object model is to add $|G| - |\mathcal{X}|$ unknown aspects to the model and connect them with possible action edges. To make the calculation more efficient, each ATG model has a single collection node representing all unobserved aspects. The belief of a collection node is defined as the probability that the robot is currently viewing an unobserved aspect of the object this ATG model represents. By specifying the transition probability between an observed and unobserved aspect, the belief of each state can be updated using the Bayes Filter Algorithm (Figure 3.4)

For each ATG in the memory pool $\mathcal{M}$, a conditional update is applied after observing each new measurement z_t. If the new observation tuple (z_{t-1}, a_{t-1}, z_t) cannot be generated by the current ATG, the ATG is augmented to keep track of what the

ATG would be if it matches the observation. If a new aspect node is created during the conditional update to match the new observation, the belief associated with the collection node representing all unobserved aspects will be transitioned to this newly created node. If a new observation tuple is in conflict with existing nodes or edges in the ATG, the new observation is discarded and the belief of the collection node is reset to zero.

Ideally, if there is high certainty that the given object is identical to the object an ATG in $\mathcal{M}$ represents, saving the augmented ATG representing this object might be beneficial. However, it is unlikely that the robot can be 100% sure that the two objects are identical with a finite number of observations. In this work, the problem is simplified by not saving the augmented nodes and edges to avoid a false match that might contaminate the memory pool.

An ATG is added to the memory pool $\mathcal{M}$ only if the presented object is judged to be novel. A novel object is defined as an object that has not been presented to the robot in the past. Although the robot might not have seen all the objects or all the aspects of each object, a limiting assumption that the robot knows that $|\mathcal{O}|$ objects exist in the environment and each object has $|G|$ aspects is made to simplify the problem. If the robot assumes that there are more objects in the environment or more aspects of an object then there actually are, it will bias the judgment toward novelty. Let $\mathcal{S}_{T-1}$ denote the set of objects that have been presented to the robot in the first $T-1$ trials. Given a sequence of observations $z_{1:t}$ and actions $a_{1:t}$ during trial T, the probability that the object presented during trial T, O_T, is novel can be calculated;

$$
\begin{aligned}
& p(O_T \notin \mathcal{S}_{T-1} | z_{1:t}, a_{1:t}, \mathcal{M}) \\
&= \sum_{o_i \notin \mathcal{S}_{T-1}} p(O_T = o_i | z_{1:t}, a_{1:t}, \mathcal{M}) \\
&= \sum_{o_i \notin \mathcal{S}_{T-1}} \sum_{x_t \in \mathcal{X}_i} p(x_t | z_{1:t}, a_{1:t}).
\end{aligned} \tag{3.1}
$$

Where o_i is an element of set $\mathcal{O}$ designating all of the objects in the environment. Element x_t of set $\mathcal{X}_i$ describes all the aspects comprising object o_i. The conditional probability $p(x_t|z_{1:t}, a_{1:t})$ of observing an aspect is inferred using a Bayes filter. Object O_T is classified as novel if $p(O_T \notin \mathcal{S}_{T-1}|z_{1:t}, a_{1:t}, \mathcal{M}) > 0.5$.

If a particular object is judged to be a previously observed object, it is associated with the ATG that is most likely to generate the same set of observations. The posterior probability of object o_i is calculated by summing the conditional probability of observing aspect x_t over all aspects comprising object o_i,

$$
p(O_T = o_i | z_{1:t}, a_{1:t}, \mathcal{M}) = \sum_{x_t \in \mathcal{X}_i} p(x_t | z_{1:t}, a_{1:t}). \tag{3.2}
$$

The posterior probability of an aspect is calculated after each measurement and control update using the Bayes Filter Algorithm ([91]). The algorithm is stated in Figure 3.4, where $\overline{bel}$ is the posterior probability $p(x_t|z_{1:t-1}, a_{1:t})$ after executing a new action a_t and bel is the posterior probability $p(x_t|z_{1:t}, a_{1:t})$ after observing a new measurement z_t. Line 3 is the control update step and line 4 is the measurement update step.

The initial belief over aspects is determined based on the number of aspect nodes and ATGs in the memory pool. Assuming that there are $|\mathcal{M}|$ ATGs in the memory pool and $|\mathcal{X}_i|$ aspect nodes are observed in G_i, the initial belief is given by

$$bel(x_0) = \frac{1}{|G_i| \cdot |\mathcal{M}|} \tag{3.3}$$

$$bel(x_0^u) = \frac{|G_i| - |\mathcal{X}_i|}{|G_i| \cdot |\mathcal{M}|} \quad , \quad x_0^u \in G_i, \tag{3.4}$$

where x^u is the collection node representing all unobserved aspects in G_i and $|G_i|$ is the number of total aspects in G_i. In this experiment, all ATGs are assumed to have the same $|G_i|$.

1: **procedure** BAYES FILTER($bel(x_{t-1}), a_t, z_t$)
2: **for** *all* x_t **do**
3: $\overline{bel}(x_t) = \sum_{x_{t-1}} p(x_t|a_t, x_{t-1}) \cdot bel(x_{t-1})$
4: $bel(x_t) = p(z_t|x_t) \cdot \overline{bel}(x_t)$
5: **end for**
6: $normalize(bel(x_t))$
7: **end procedure**

Figure 3.4. Bayes Filter Algorithm

Assuming that transitions between aspects are deterministic; given the current aspect, the same action always leads to the same next aspect. Therefore, each aspect only has one outward action edge of the same type. The transition probability $p(x_t|a_t, x_{t-1})$ in the control update step for each aspect can be calculated by counting how many possible aspect nodes (including the collection nodes) the current aspect can transition to.

To simplify the problem, it is assumed that there is no noise in the measurement data. Therefore, the observation probability $p(z_t|x_t)$ would be either 1 for a match or 0 for a mismatch. Note that, although there is no uncertainty in the measurement data, there is still uncertainty over aspects since the same observation z_t can be generated by different objects.

3.2.2 Information Theoretic Planner

The challenge of integrating task-level planners with incomplete representations requires handling partial observability of the state while building plans. Since the true state of the system cannot be observed, it must be inferred from the history of observations and actions. The proposed planner belongs to a set of approaches (such as [70]) that select actions to reduce the uncertainty of the state estimate maximally with respect to the task.

Object recognition can be viewed as one such task in which the uncertainty over object identities (as quantified by the object entropy) is reduced with each observation. Selecting the action a_t that minimizes the expected entropy of the distribution over elements of set O_T representing the object identity reduces the uncertainty over object identities the most after the next observation z_{t+1};

$$\begin{aligned}
&\underset{a_t}{\operatorname{argmin}}\ E(H(O_T|z_{t+1}, a_t, z_{1:t}, a_{1:t-1})) \\
&= \underset{a_t}{\operatorname{argmin}} \sum_{z_{t+1}} H(O_T|z_{t+1}, a_t, z_{1:t}, a_{1:t-1}) \cdot p(z_{t+1}|a_t, z_{1:t}, a_{1:t-1}).
\end{aligned}$$

Where H is the entropy associated with the random variable. The entropy is zero if the state is uniquely determined; it reaches its maximum if all states are equally likely;

$$\begin{aligned}
&H(O_T|z_{t+1}, a_t, z_{1:t}, a_{1:t-1}) \\
&= \sum_{o_i \in \mathcal{O}} p(o_i|z_{1:t+1}, a_{1:t}) \log p(o_i|z_{1:t+1}, a_{1:t}).
\end{aligned} \tag{3.5}$$

The posterior probability $p(o_i|z_{1:t+1}, a_{1:t})$ can be calculated by updating the existing belief using the Bayes Filter Algorithm. The prior probability $p(z_{t+1}|a_t, z_{1:t}, a_{1:t-1})$ of

observing z_{t+1} given past observations can be calculated by summing the probability of all aspects generating observation z_{t+1},

$$\begin{aligned} & p(z_{t+1}|a_t, z_{1:t}, a_{1:t-1}) \\ & = \sum_{o_i \in \mathcal{O}} \sum_{x_{t+1} \in \mathcal{X}_i} p(x_{t+1}|a_t, z_{1:t}, a_{1:t-1}) \cdot p(z_{t+1}|x_{t+1}). \end{aligned} \tag{3.6}$$

Where the posterior probability of an aspect $p(x_{t+1}|a_t, z_{1:t}, a_{1:t-1})$ is updated using the Bayes Filter Algorithm.

The runtime for calculating the expected entropy given an action is $O(|\mathcal{F}| \cdot |\mathcal{O}|^2 \cdot |\mathcal{X}|)$. To speed up the calculation, an approximate expected entropy for each action is calculated instead:

$$\begin{aligned} & E(H(O_T|z_{t+1}, a_t, z_{1:t}, a_{1:t-1})) \\ & \simeq \sum_{z_{t+1}} H(O_T|z_{t+1}, a_t, z_{1:t}, a_{1:t-1}) \cdot \\ & \quad p(z_{t+1}|a_t, z_{1:t}, a_{1:t-1}) \cdot \\ & \quad \mathbb{1}_{(threshold, \infty)}(p(z_{t+1}|a_t, z_{1:t}, a_{1:t-1})). \end{aligned} \tag{3.7}$$

Here $\mathbb{1}(\cdot)$ is the indicator function and the *threshold* is set to $1/|\mathcal{F}|$ in this experiment. The value of the indicator function is 1 if the input is greater than *threshold* and 0 otherwise.

If an observation z_{t+1} is unlikely to be observed, the entropy $H(O_T|z_{t+1}, a_t, z_{1:t}, a_{1:t-1})$ will not be calculated. The approximate expected entropy will be lower than the actual entropy, however it effects all estimates in the same way and should allow us to identify the action that leads to the minimum entropy most of the time.

3.2.3 Experiments

The proposed information theoretic planner combined with ATG object models is tested on the dual problem of modeling and reasoning formalized as simultaneous

object modeling and recognition (SOMAR). The goal of SOMAR is to have the robot build up a set of object models through interacting with random objects one at a time. The task is evaluated based on whether the robot can identify novel objects and recognize which object model it corresponds to if it have been observed in the past. How ATG object models can be used to reach certain goal aspect by recognizing objects is also demonstrated.

This problem is inspired by the simultaneous localization and mapping (SLAM) problem introduced in [54]. Instead of building a map while localizing the robot, the task requires performing object modeling and recognition at the same time. The SOMAR problem is equivalent to a modified SLAM problem where multiple incomplete maps are given to the robot and the goal is to locate the robot in one of the maps or identify that the robot is in none of these maps and start modeling the current environment.

3.2.3.1 Settings

The capabilities of the proposed model and planner is evaluated using the Robonaut-2 simulator shown in Figure 3.6 ([15]) and an exclusively kinematic simulator. The kinematic simulator runs much faster and is used to collect more data for comparing different planners. The simulation environment contains 100 unique objects called ARcubes that consist of a $28cm$ cube with unique combinations of ARtags on the six faces; 12 different ARtag patterns are used in this experiment. ARcubes are simple objects that share a common set of manual actions and introduce a challenging amount of hidden states from any single sensor viewpoint. In an ATG for an ARcube, an aspect consists of ARtag features observed on the top face and the front face. As in Figure 3.1 each ATG has 24 unique aspects and each aspect has 132 different pattern combinations. For the sake of simplicity, it is assumed that an object does not have two faces with the same ARtag.

In the Robonaut-2 simulator, the simulated Asus Xtion sensor located in Robonaut-2's head is used for visual and depth input. The ARtags located on the ARcubes are detected and recognized using the ARToolkit ([43]) and are classified as the top or front ARtag based on the 3 dimension position.

The robot can perform 3 different manipulation actions on the object: 1) flip the top face of the cube to the front, 2) rotate the left face of the cube to the front, and 3) rotate the right face of the cube to the front. The robot will be able to execute any of these actions under the condition that it observes both of the ARtags. If the ARcube is tilted, too far or too close to manipulate, the robot will try to adjust the cube until it is in the right position. These adjustment actions are not stored in the ATGs.

3.2.3.2 Results

Tables 3.2 and 3.3 show the result of using the planner to recognize the object presented. Each test involves 100 trials and starts with an empty memory pool $\mathcal{M}$. In each trial, the task is to decide which ATG in memory pool the given object corresponds to or to declare it as a novel object. For each trial, an object is chosen at random and presented to the robot. The robot observes the object and executes an action. This process is repeated 10 times in the first experiment and 20 times in the second experiment. At the end of each trial the robot determines the likelihood that the presented object is novel and the most likely existing object in memory pool is identified.

The last row in Table 3.2 and Table 3.3 presents the results averaged over all the tests. The success rate is the percentage of objects correctly classified, that is, correctly identified in memory pool or declared as a novel object. When 10 actions are performed per trial, the system correctly recognizes the object 90.7% of the time and correctly determines if the presented object is novel 81.6% of the time. The overall

Table 3.2. The success rate of an information theoretic planner in recognizing the object (10 actions per trial)

Test	Correct Identification	Correct Recognition	Success Rate
1	80/100	20/21	79%
2	79/100	25/27	77%
3	87/100	21/25	83%
4	78/100	26/28	76%
5	84/100	24/27	81%
average	81.6%	90.7%	79.2%

Table 3.3. The success rate of an information theoretic planner in recognizing the object (20 actions per trial)

Test	Correct Identification	Correct Recognition	Success Rate
1	100/100	34/34	100%
2	98/100	32/32	98%
3	98/100	40/40	98%
4	99/100	37/37	99%
5	99/100	32/32	99%
average	98.8%	100%	98.8%

success rate is 79.2% in this experiment. When 20 actions are performed per trial, the overall success rate reaches 98.8%.

The efficiency of the planner is also tested against a random policy. The number of actions executed per trial were varied from 4 to 20. Figure 3.5 shows how the success rate of a test varies with the number of actions executed per trial. As is evident from the plots, the information theoretic planner outperforms a random exploration policy for all cases except when the number of actions per trial is low. Both algorithms perform equally poor when not enough information is provided.

3.2.3.3 Reaching a Target Aspect

To demonstrate how ATGs can be used to reach certain goal state, an environment where three ARcubes are located in front of the simulated Robonaut-2 is set up as

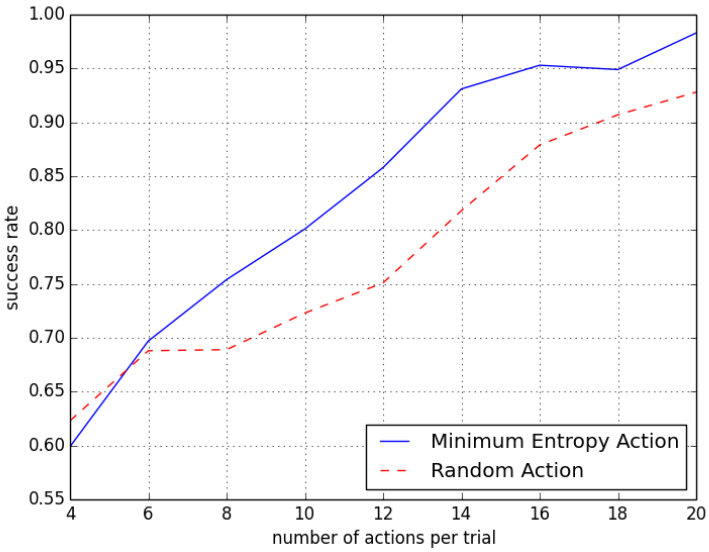

Figure 3.5. The plot shows the average success rate of 10 tests as the number of actions per trial are increased. Selecting actions that minimize entropy leads to a higher success rate then selecting actions at random.

Figure 3.6. The simulated Robonaut-2 interacting with an ARcube.

shown in Figure 3.6. The goal is to rotate the cubes until certain faces are observable. The robot starts with a memory pool learned through interacting with 20 different ARcubes including the three ARcubes located in the test environment. To achieve the goal state, Robonaut-2 manipulates the object to condense belief over objects. Once the object entropy is lower than a threshold, Robonaut-2 tries to execute the sequence of actions that is on the shortest path from the current aspect node to the goal aspect node in the corresponding ATG if such goal aspect node exists. In this demonstration, the simulated Robonaut-2 successfully reaches the goal state by manipulating the cubes so that the observed aspects match the given goal aspects.

3.3 Funnel-Slide-Funnel Structure

Designing robots that can model unstructured environments and act predictably in those environments is challenging. This section addresses the issue of how to make actions repeatable in a noisy environment. Subsection 3.1.3 describes how open-loop and closed-loop controllers can be constructed into a funnel-slide-funnel structure (Figure 3.2) such that under certain conditions a sequence of actions that include open-loop controllers can guarantee success. In this section, a novel visual servoing algorithm that can be used to converge to a certain aspect node within the ATG is first introduced. This visual servoing alogrithm is then used to construct a funnel-slide-funnel structure and have shown to improve action accuracy significantly on a tool grasping task in the Robonaut-2 simulator.

3.3.1 Visual Servoing

Hutchinson et al. classify visual servoing approaches into two major types: position-based servoing, where servoing is based on the estimated pose; and image-based servoing, where servoing is based directly on visual features [38]. In this section, a novel image-based visual servoing algorithm that can be used to converge from an obser-

vation within the region of attraction to the ϵ-region of the corresponding aspect in an ATG is introduced. This visual servo controller falls under the control basis framework [36] and can be written in the form $\phi|_{\tau}^{\sigma}$, where ϕ is a potential function, σ represents sensory resources allocated, and τ represents the motor resources allocated. The control basis framework provides a means for robot systems to explore combinations of sensory and motor controls. In this experiment, the visual servoing controller is used to control the end effector of the robot to reach a pose relative to a target object using visual sensor feedback. Unlike many visual servoing approaches, this visual servoing algorithm does not require a set of predefined visual features on the end effector or target object nor does it require an inverse kinematic solution for the robot. The only information required is the current observation and the target aspect. Figure 3.7 shows a trial of the visual servoing algorithm converging to a stored target aspect.

In the control basis framework, a potential function ϕ represents an error function that the controller minimizes. To reach minimum error, a closed loop controller performs gradient descent on the potential function. Artificial potential functions that guarantee asymptotically stable behavior are usually used to avoid local minima [32]. However, potential functions with a unique minimum often do not exist in visual servoing due to occlusion, lighting, and noisy sensory data. Instead of trying to define a potential function with a unique minimum, I define a potential function with possibly many local minima and call the region in which gradient descent converges to a particular minimum the region of attraction. If the current aspect is within the region of attraction, convergence to the target aspect can be guaranteed through gradient descent.

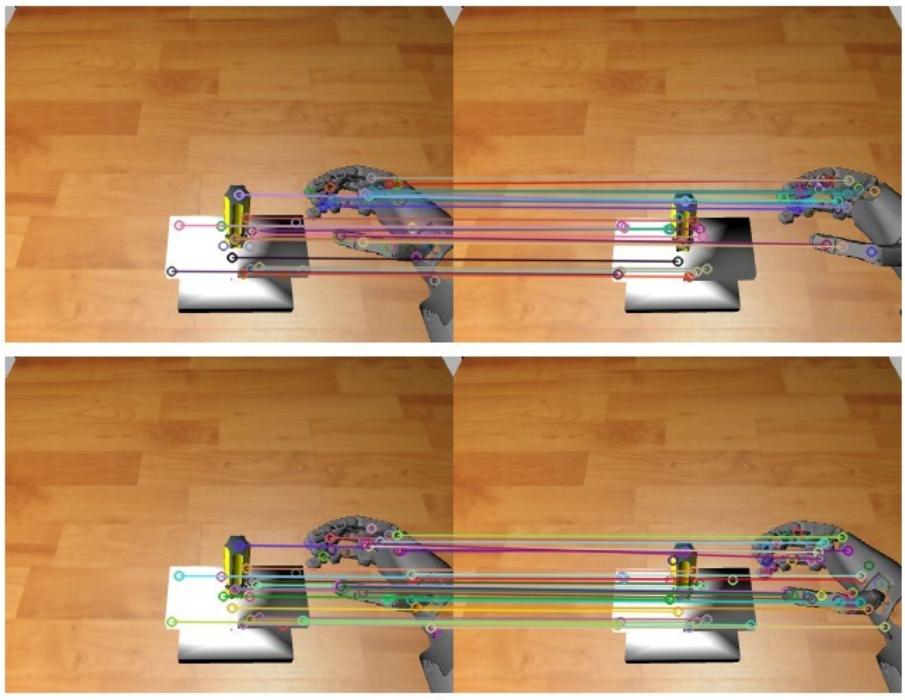

Figure 3.7. Visual servoing sequences. Each image pair shows the target aspect (left) and the current observation (right). A line in between represents a pair of matching keypoints. The top image pair represents the starting observation and the bottom image pair represents when the controller converged.

3.3.1.1 Potential Function

The potential function is defined as the weighted squared Euclidean distance between the signature of the current observation $\tilde{\boldsymbol{s}}$ and the signature of the target aspect $\boldsymbol{s}$. This approach can be used with most feature detectors and feature descriptors. In this work, the Fast-Hessian detector and the SURF descriptor [5] are implemented. A depth filter that uses the depth image is first used to filter out most keypoints that belong to the background. The first step to calculate the signature of an observation is to find a subset K of keypoints in the current observation that match to keypoints in the target aspect. The signature of an observation can then be calculated based on this subset K of keypoints. The signature is a combination of the distance signature vector $\boldsymbol{s}^D$ and the angle signature vector $\boldsymbol{s}^A$. $\boldsymbol{s}^D$ is a signature vector that consists of Euclidean distances s_{ij}^D between all pairs of keypoints (k_i, k_j) in K:

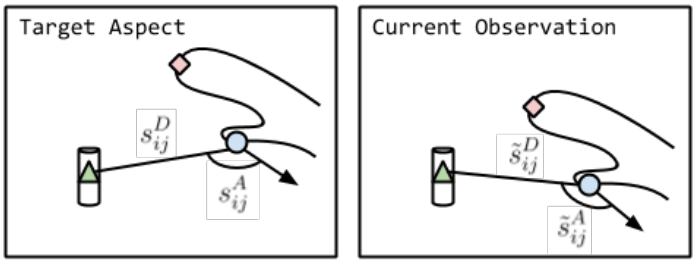

Figure 3.8. Components of the signature of the target aspect (left) and the current observation (right). The circle and the triangle represent the ith and jth matched keypoints.

$s_{ij}^D = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}$. Here x_i, y_i are the X Y image coordinates of keypoint $k_i \in K$. The angle signature vector $\boldsymbol{s}^A$ consists of angle differences s_{ij}^A between all pairs of keypoints (k_i, k_j) in K: $s_{ij}^A = \omega_{ij} - \theta_i$. Here ω_{ij} represents the orientation of the ray from keypoint k_i to keypoint k_j and θ_i represents the orientation of keypoint k_i. Figure 3.8 illustrates examples of s_{ij}^D and s_{ij}^A of the target aspect and the current observation.

The potential ϕ is then the scaled squared Euclidean distance between distance signature vectors of the target aspect $\boldsymbol{s}^D$ and the current observation $\tilde{\boldsymbol{s}}^D$ plus the weighted squared Euclidean distance between angle signature vectors of the target aspect $\boldsymbol{s}^A$ and the current observation $\tilde{\boldsymbol{s}}^A$;

$$\phi = \frac{1}{N_D} \cdot \sum_{\{i,j|k_i,k_j \in K\}} (s_{ij}^D - \tilde{s}_{ij}^D)^2 + \sum_{\{i,j|k_i,k_j \in K\}} w_{ij}^A \cdot (s_{ij}^A - \tilde{s}_{ij}^A)^2,$$

where $N_D = |K| \cdot (|K| - 1)/2$ and $w_{ij}^A = s_{ij}^D / \sum_{\{i,j|k_i,k_j \in K\}} s_{ij}^D$. Here $|K|$ is the number of matched keypoints between the current observation and the target aspect and w_{ij}^A is a normalized weight proportional to the keypoint pair distance s_{ij}^D in the target aspect. The purpose of w_{ij}^A is to weight angle differences more heavily for keypoints that are far apart.

3.3.1.2 Gradient Descent

In order to perform gradient descent on the potential function, the potential-motor Jacobian defined as

$$J = \frac{\partial \phi(\sigma)}{\partial \boldsymbol{\tau}}$$

need to be estimated. A seven degree freedom arm is used in this experiment, therefore $\boldsymbol{\tau} = [q_1, q_2, ..., q_7]$ where q_i represents the ith joint in Robonaut-2's right arm. The control signal that leads to the greatest descent can then be calculated by the expression:

$$\Delta \boldsymbol{\tau} = -c(J^{\#} \phi(\sigma)),$$

where c is a positive step size and $J^{\#}$ is the Moore-Penrose pseudoinverse [61].

In order to calculate the partial derivative of the potential function ϕ with respect to each joint q, the visuomotor Jacobian is defined as

$$J_v = \frac{\partial V}{\partial \boldsymbol{\tau}},$$

where V is the X Y positions and orientations of the set of keypoints detected in the current observation that match to keypoints in the target aspect based on its feature descriptor. Knowing $\Delta \boldsymbol{\tau}$ we can calculate the change in the keypoint positions and angles as follow:

$$\begin{bmatrix} \dot{x}_1 \\ \dot{y}_1 \\ \dot{\theta}_1 \\ \vdots \\ \dot{x}_n \\ \dot{y}_n \\ \dot{\theta}_n \end{bmatrix} = \begin{bmatrix} \frac{\partial x_1}{\partial q_1} & \frac{\partial x_1}{\partial q_2} & \cdots & \frac{\partial x_1}{\partial q_7} \\ \frac{\partial y_1}{\partial q_1} & \frac{\partial y_1}{\partial q_2} & \cdots & \frac{\partial y_1}{\partial q_7} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial \theta_n}{\partial q_1} & \frac{\partial \theta_n}{\partial q_2} & \cdots & \frac{\partial \theta_n}{\partial q_7} \end{bmatrix} \cdot \begin{bmatrix} \dot{q}_1 \\ \dot{q}_2 \\ \dot{q}_3 \\ \dot{q}_4 \\ \dot{q}_5 \\ \dot{q}_6 \\ \dot{q}_7 \end{bmatrix}$$

$$\Delta V = J_v \cdot \Delta \boldsymbol{\tau},$$

where x_i, y_i, and θ_i represent the X Y position and orientation of keypoint k_i that match to the ith keypoint in the target aspect; n is the number of keypoints detected in the target aspect. Given J_v we can estimate how changing joint values $\boldsymbol{\tau}$ will change the position and orientation of the matched keypoints on the image plane. Since the potential only depends on matched pairs, estimated potential for every joint value can be calculated.

3.3.1.3 Learning the Visuomotor Jacobian

This visuomotor Jacobian that models how features change with respect to joint values is inspired by work done in understanding how humans obtain a sense of agency by observing their own hand movements [93]. This approach learns that certain feature positions on the robot end effector are controllable while features in the background are not. The visuomotor Jacobians for each aspect are updated on-line using a Broyden-like method

$$J_{v_{t+1}} = J_{v_t} + \mu \frac{(\Delta V - J_{v_t} \Delta \boldsymbol{\tau}) \Delta \boldsymbol{\tau}^T}{\Delta \boldsymbol{\tau}^T \Delta \boldsymbol{\tau}},$$

where J_{v_t} is the visuomotor Jacobian at time t and $\mu \in (0, 1]$ is a factor that specifies the update rate [60]. When $\mu = 1$ the updating formula will converge to the correct Jacobian J_v after m noiseless orthogonal moves and observations, where m is the dimension of J_v. In this experiment, μ set to 0.1 is found to be more robust. The visuomotor Jacobians for each aspect are initialized randomly for the first run and memorized afterwards. The more trials the controller runs the more accurate the estimated J_v is on average. Using Broyden's method to estimate Jacobians on-line for visual servoing was first introduced in [40].

3.3.2 Experiments

The ATG in conjunction with the visual servoing algorithm are tested on a tool grasping task on the NASA Robonaut-2 simulator ([15]). The goal of the task is to control Robonaut-2's right hand to a pose where a screwdriver on a tool stand is in between the robot's right thumb, index finger and middle finger as shown in Figure 3.9. An ATG object model consisting of three aspects, that is sufficient for this task, is used in this experiment. It is shown that a "slide-funnel-slide-funnel" controller sequence decreases the average pose error over a "slide-slide" controller sequence.

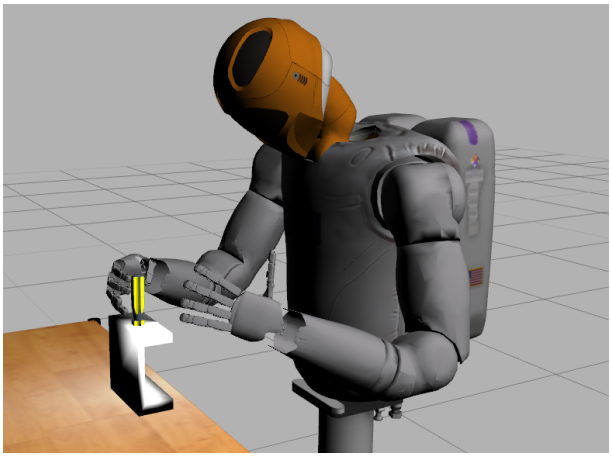

Figure 3.9. Robonaut-2 approaching a pre-grasp pose for a screwdriver on a tool stand in simulation.

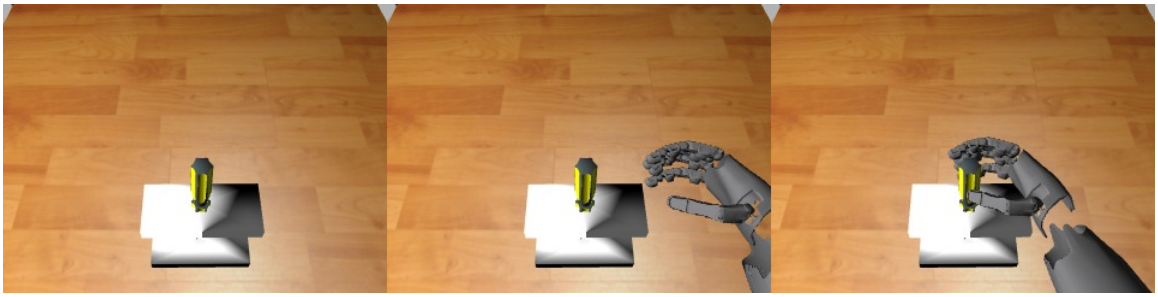

Figure 3.10. The first, second, and third aspect stored in the ATG through demonstration are shown from left to right. In the first aspect, the object on top of the table is a screwdriver on a tool stand. In the second aspect, the robot hand is in a position where a straight movement toward the screwdriver would lead to a pre-grasp pose. The third aspect represents a pre-grasp pose. This is the goal aspect for the pre-grasp task designed in this experiment.

3.3.2.1 Settings

The ATG used in this experiment consists of three aspects. The first aspect represents an observation in which the screwdriver is on a tool stand on a table and is 0.6 meters in front of the robot. The left image in Figure 3.10 is the corresponding observation of this aspect. The second aspect represents an observation where the robot's right hand is about 7 centimeters right of the screwdriver. The action edge between the first and second aspects represents an action that moves the robot's right hand to a pose relative to the center of the object point cloud observed in the first aspect. The middle image in Figure 3.10 is the corresponding observation of this aspect. The third aspect represents an observation where the robot's right thumb, index and middle finger surrounds the screwdriver handle. The right image in Figure 3.10 is the corresponding observation of this aspect. The action edge in between the second and third aspects represents an action that moves the robot's right hand to a pose relative to the right hand pose of the previous aspect.

3.3.2.2 Analyzing Region of Attraction

The region of attraction of the second and third aspect of the ATG with respect to the visual servoing controller can be analyzed. The region of attraction of an aspect is defined as the observation space in which a closed loop convergence controller that does not rely on the object representation can converge to the ϵ-region of the aspect. An aspect or observation lies in a high dimensional observation space and can be varied by multiple different parameters or noise. In this experiment, two types of noise are considered. 1) Noise in the relative pose between the robot hand and the object. This kind of noise can be caused by kinematic errors from executing an action or imperfect object positions calculated from a noisy point cloud and will result in a different end effector pose relative to the object. 2) Noise in the object position. This kind of noise can be caused by placing the tool stand and screwdriver in a different position than the position previous observed in the demonstration. This noise can cause the estimated object center position to vary and affect the visual servoing controller since the object and the robot end effector will look visually different from a different angle. In this experiment, our goal is to find the region of attraction of the second and third aspect with respect to these two kinds of noise.

These two kinds of noise are artificially added in the experiment and the number of gradient descent iterations required to reach the ϵ-region of the aspect are recorded. In this experiment, only noise on the X Y plane is considered for easier visualization and analysis. For each type of noise and each aspect, 289 different combination of noise in the X and Y axis roughly within the scale that the visual servoing controller can handle is tested. The results for adding noise in the relative pose between the robot hand and the object to the second aspect are shown in Figure 3.11. The plot on the left indicates how many iterations the visual servoing controller executed till convergence for different noise values. Each color tile is one single experiment and dark blue means the controller converges fast while orange means the controller took

longer to converge. A yellow tile means that the controller could not converge within the 1000 iteration threshold. The region of attraction is the set of observations that include the aspect plus the set of noise that corresponds to a non yellow tile connected to the origin. The plot on the right is a visualization of the same result in 3D which has some resemblance to the funnel metaphor used in Figure 3.2.

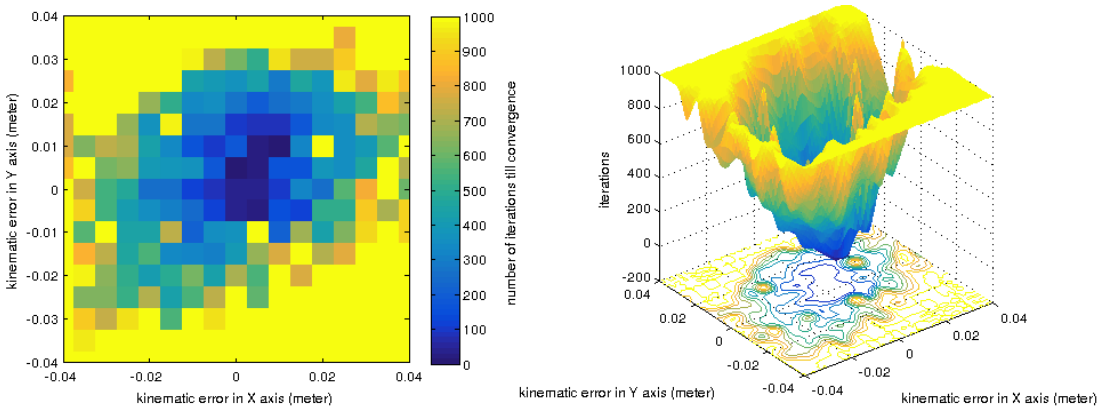

Figure 3.11. Iteration till convergence with respect to noise in the relative pose between the robot hand and the object for the second aspect.

The results for adding noise in the relative pose between the robot hand and the object to the third aspect are shown in Figure 3.12. Note that this aspect has a smaller region of attraction with more tolerance in the direction perpendicular to the hand opening. If there is a large error in the Y axis the robot's hand may end up in front or behind the screwdriver. Under such situations, without additional information the visual servoing controller will not be able to avoid colliding with the screwdriver while trying to reach the goal. The results for adding noise in the object position are shown in Figure 3.13. Notice that the regions of attraction are much larger for this type of noise.

3.3.2.3 Convergence and Accuracy

By analyzing the observed regions of attraction of the visual servo controller that converges to the two aspects, the magnitude of noise this "slide-funnel-slide-funnel"

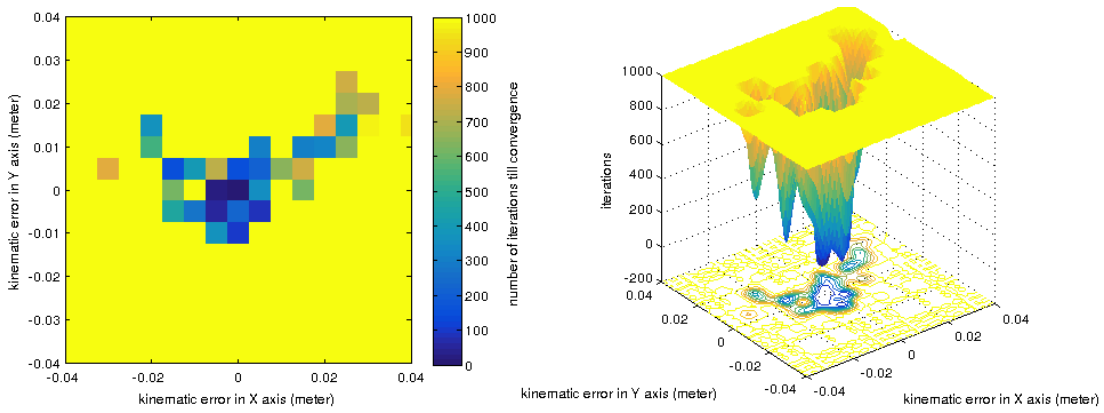

Figure 3.12. Iteration till convergence with respect to noise in the relative pose between the robot hand and the object for the third aspect.

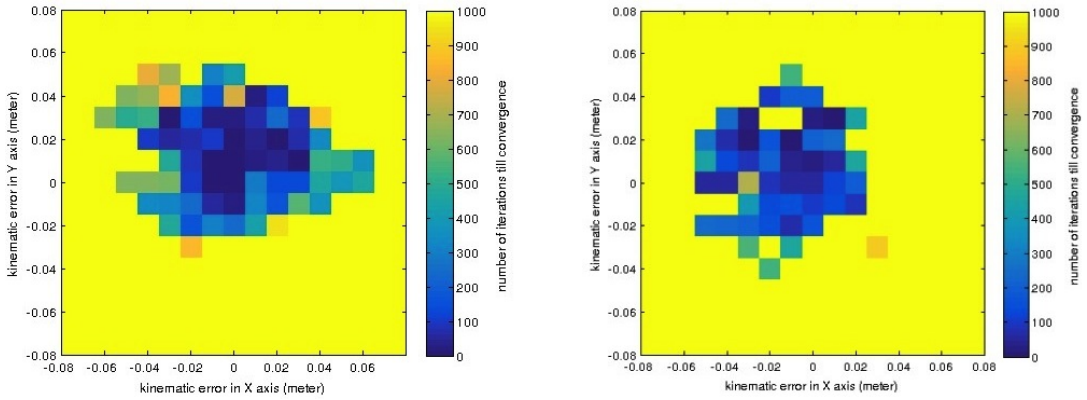

Figure 3.13. Iteration till convergence with respect to noise in the object position for the second aspect (left image) and the third aspect (right image).

controller sequence can tolerate can be estimated. As shown in Figure 3.11 and Figure 3.12, the visual servo controller has a region of attraction with about 1.5 centimeter radius of kinematic noise around the second aspect and about 0.5 centimeter radius of kinematic noise around the third aspect. These sequences of actions are evaluated by comparing the final end effector position in the X-Y plane to the demonstrated pose relative to the screwdriver. Noise of three different magnitudes are added to each open-loop action; 0.5, 1.0, and 1.5 centimeters for the action that transitions from the first aspect to the second aspect and 0.1, 0.2, and 0.3 centimeters for the action that transitions from the second aspect to the third aspect. For each combination of noise, eight uniformly distributed directions are tested. Among the 72 test cases 100% of them converged to the second aspect and 87.5% of them converged to the third aspect.

This experiment did not reach a 100% overall convergence rate for two possible reasons. First, in addition to the artificial noise, randomness in the action planner and simulator also exist in the system. Second, the region of attractions shown in the previous section are estimated based on visual similarity. Two observations can be visually similar but position wise quite different therefore causing a false estimate of convergence. Figure 3.14 shows the test cases that the controller fails to converge on; most of the failed test cases are located in the lower right corner. This is consistent with the shape of the region of attraction of the controller with respect to the third aspect shown in Figure 3.12. The final poses of the end effector relative to the screwdriver are recorded and compared to the demonstrated pose.

The results are further compared to a sequence of "slide-slide" controllers without visual servoing acting as a funnel. The average position error is shown in Table 3.4. The "slide-funnel-slide-funnel" structure reduces the error by 55.8% and has an average error of 0.75 cm in the X-Y plane when only considering test cases that converged.

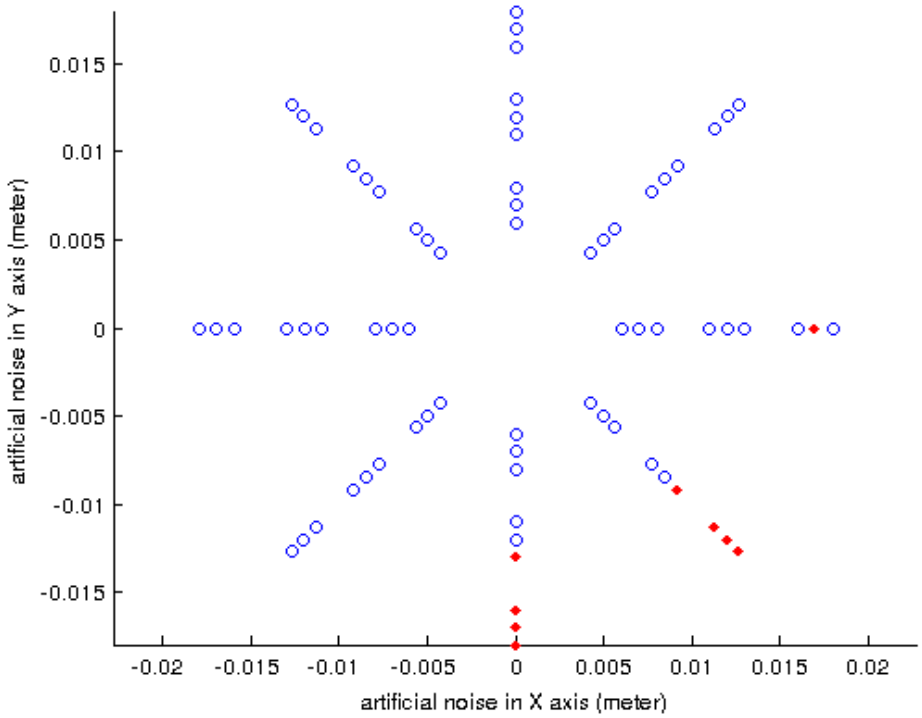

Figure 3.14. Convergence with respect to artificial noise added to the test cases. Each dot represents a test case where the X Y value represents the summed magnitude and direction of the manually added kinematic noise. A red diamond indicates that the controller fails to converge to the third aspect while a blue circle indicates that the action sequence converged.

Table 3.4. Average position error in the X-Y plane in centimeters.

	complete test set	converged test set
"slide-slide" structure	2.24 cm	2.06 cm
"slide-funnel-slide-funnel" structure	0.99 cm	0.75 cm

3.4 Acting without Explicit Pose Estimation

In most robotics tasks that require manipulating known objects, accurate pose estimation is often required before planning end effector trajectories. In the Willow Garage grasping pipeline [98], the iterative closest point algorithm is used to check how well a segmented point cloud matches to a stored mesh model. Pre-computed grasp points associated with the model are then used to generate a valid motion trajectory. However, object pose estimation is often computationally expensive and inaccurate. Using the ATG representation introduced in this work, an alternative approach that interacts with objects directly based on memory and observation is introduced.

3.4.1 Approach

Instead of storing a 3D representation that contains invariant features in the object frame, the ATG stores a set of viewpoint-specific observations. Since available actions from one observation to another are stored in action edges of an ATG, explicit object pose estimation is not required to interact with an object. Given a sufficient ATG, the robot can directly interact with objects based on memorized observations.

3.4.2 Experiments

Such approach is tested on a tool grasping task on Robonaut-2. The goal is to have Robonaut-2 use the drill directly with its left hand or grasp the drill from the top or the side with its left hand so that it can adjust the drill to a better grasping pose for the right hand. An ATG of a drill representing five different grasping trajectories for different drill orientations ranging from 0 to 180 degree is first created. One goal aspect that represents successfully grasping the drill is used to connect aspect nodes representing the end of the five grasping trajectories.

A grasping test is performed on 21 random drill poses ranging from 0 to 180 degrees and within 10 cm from the original training position. This approach successfully

grasped the drill 19 out of 21 times in this experiment. One of the two failures was caused by the planner failing to generate a valid trajectory to an intermediate aspect and the other was caused by failing to reach an intermediate aspect. Figure 3.15 shows examples of Robonaut-2 grasping the drill oriented at different poses during testing.

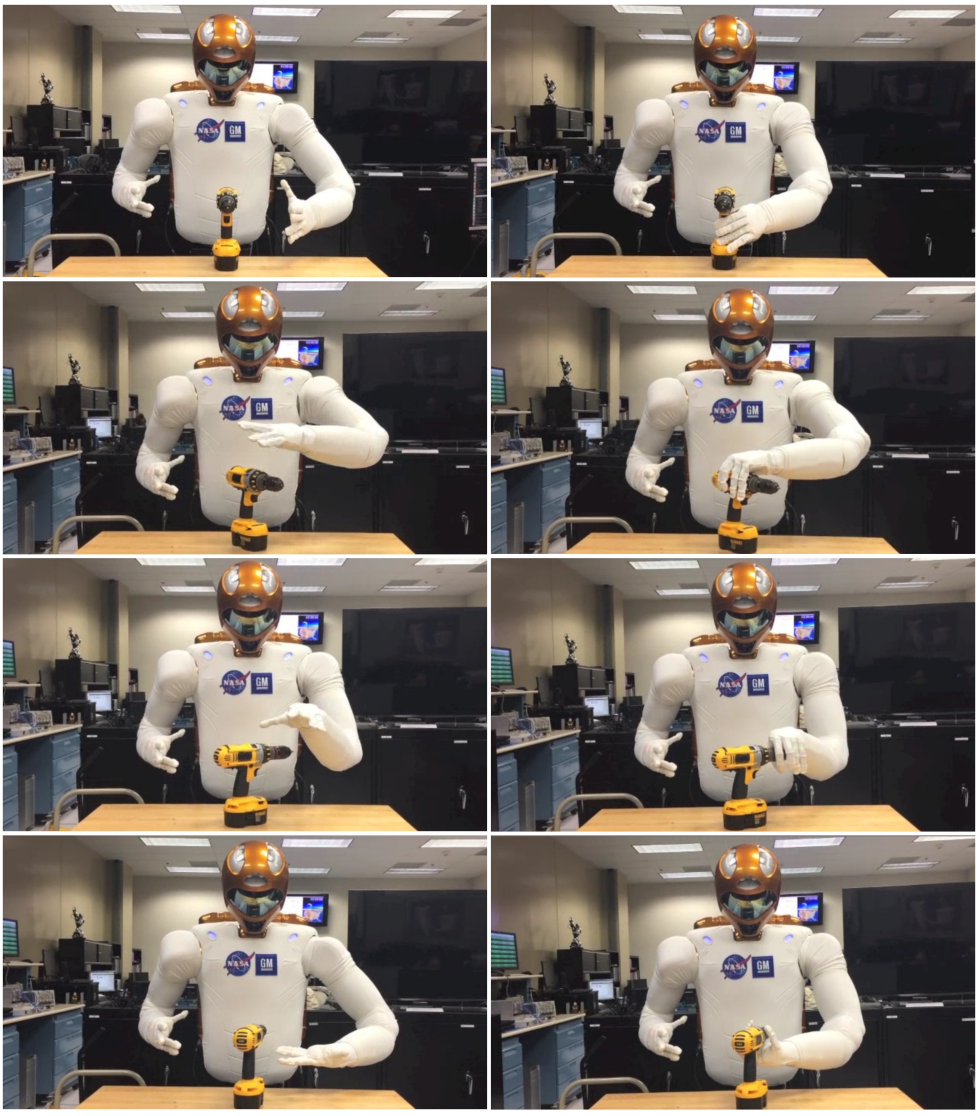

Figure 3.15. Robonaut-2 grasping the drill posed at different orientations. Image pairs in the same row represents the intermediate and final states of one drill grasping trial.

3.5 Error Detection and Surprise

Building robots that can handle uncertainty is necessary for autonomous robots to accomplish tasks in a non-deterministic environment. However, the vast majority of failures either evade detection completely or are detected only after a high level action fails to reach the target state. This makes robots inefficient and can lead to catastrophic failure if the robot continues to execute a plan when the actual state is quite different from what the robot expected. In Section 3.2, how ATG models can be used to handle uncertainty caused by partial observability is discussed. In this section, a framework that uses ATG models to tackle uncertainty caused by a non-deterministic environment is introduced. In this framework, plans are monitored during the execution of low-level actions to create a fine-grained observer for error detection. This allows the robot to compare observations to stored ATG models frequently and to handle unexpected outcomes immediately after they are detected.

In the book Probabilistic Robotics [91], environment, sensors, robots, models, and computation are considered as the five factors that give rise to uncertainty in robotic applications. This section focuses on how uncertainty is perceived instead of categorizing it based on the cause. Uncertainties a robot may encounter are classified into two categories: 1) random transitions that are within the model domain but may still be unlikely given transition probabilities, and, 2) "surprise" transitions that lead to outcomes that are not represented in the model domain. An example of a random transition is the outcome of rolling a loaded die that should come up 6 with high probability, but instead comes up 1. The robot will not be able to predict the exact outcome, but by having a model that captures low probability outcomes it is capable of handling each of them accordingly. An example of a surprise transition is if the outcome of rolling a 6 sided die is 13 while the robot's die model has possible outcomes from 1 to 6. In [12], Casti describes that "surprise can arise only as a consequence of models that are unfaithful to nature." Therefore I define a "surprise"

transition as when the robot observations are not explainable by any model in the robot memory. During the surprise transition a new model of the environment that can explain this surprising outcome is learned. Failures detected by the fine-grained observer are then recovered differently based on this classification.

In a non-deterministic environment it is inevitable that a robot will encounter randomness and surprises while performing tasks. In this section, how randomness can be modeled and handled is first described. A new formula for surprise is then introduced. In Subsection 3.5.3, experiments are tested on the balancing mobile manipulator uBot-6 [75]. It is shown that the framework can handle random transitions and recover from surprise transitions at fine-grained time scales.

3.5.1 Modeling Randomness

In an ATG model, an action edge u represents an action that causes an aspect transition. Actions can be implemented with open-loop or closed-loop controllers. For each action, its type, parameterization, and reference frame are stored. A robot has a set of actions available to interact with objects. These actions include visual and haptic servoing actions as well as gross motor (mobility) actions and fine motor (arm and hand) actions. While mobility actions can be used to bring an object into reach for manipulation, they also alter the view of an object, possibly revealing previously hidden visual features. Actions such as pick-up and lift can be used to manipulate objects, but they can also gather haptic feedback and change the current viewpoint to reveal otherwise unavailable features (e.g. surface markings on the bottom face of a box).

High-level manipulation actions can be represented as a sequence of primitive actions and aspect transitions. For example, the "flip" macro action is implemented as a sequential composition of the following four primitive actions: 1) reach, 2) grasp, 3) lift, and 4) place. These four actions connect five fine-grained aspect nodes that repre-

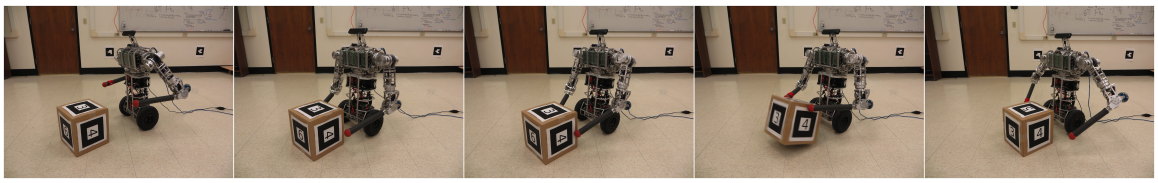

Figure 3.16. Fine-grained flip action. The photos shown from left to right are the five intermediate stages of a flip action. The robot checks whether the sub-action succeeded for each stage.

sent expectations for intermediate observations. Figure 3.16 shows these intermediate stages. These fine-grained aspect nodes allow us to monitor and detect unplanned transitions at many intermediate stages of the flip interaction.

I define a random transition as an outcome of an action that is non-deterministic but is within a previously seen set of outcomes that can be captured in the model. For example, due to uncertainty, the lift action may drop the object in the lifting process due to slippery hands, thus, causing a different observation in the terminating state. These random outcomes can be represented in the ATG using transition probabilities $p(x_t|a_t, x_{t-1})$, which is nonzero for all possible outcome states. As a consequence, the Bayesian filtering algorithm will update the belief accordingly. For simplicity, equal transition probabilities are assigned to all of the outcome states possible under action a_t. In general, each outcome of a randomized transition requires a different action to achieve the goal aspect. Figure 3.17 shows an ATG model that handles such randomness.

3.5.2 Recovery from Surprises

In [12], surprise is described as "the difference between expectation and reality." Although one might associate surprise or unexpected events with low probability events, observing a rare event does not always lead to surprise. For example, a person that observes a lottery drawing will likely not be surprised if the outcome is 29 - 41 - 48 - 52 - 54 even though this outcome has a low probability.

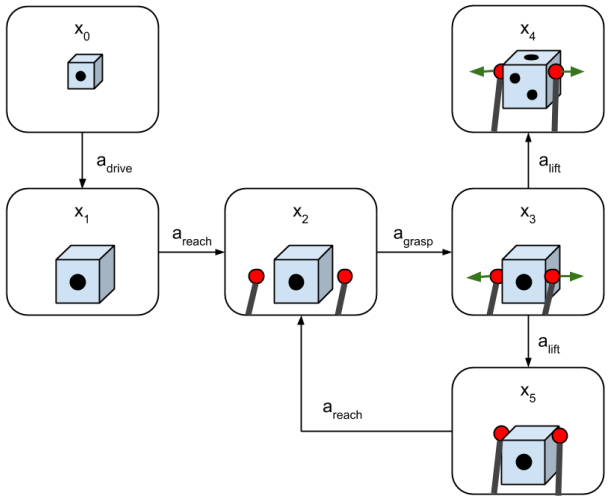

Figure 3.17. Part of an aspect transition graph model of a dice. The top right node indicates the observation when the robot successfully flipped the dice while the bottom right node indicates when the dice slipped. The red circles indicate the robot hands and the green arrows indicate haptic feedback.

Baldi [4] measures the degree of surprise in a Bayesian setting in which there is a well-defined distribution $P(M)$ over the space of models $\mathcal{M}$. He uses a measure S of how much the new information in an observation d changes the distribution over models. Specifically, he computes the KL-divergence between the current belief state $P(M)$ and the belief state $P(M|d)$ induced by the new observation:

$$S(d, \mathcal{M}) = \int_{\mathcal{M}} P(M) \log(P(M)/P(M|d)) dM.$$

An event is then defined to be a surprise if this divergence is greater than some threshold θ.

However, this formula considers transitions where an informative action reduces the entropy over models significantly as surprising. If there is a uniform distribution over models, observing an outcome that concentrates the posterior probability on one model should not be surprising. I offer a simpler definition of surprise that agrees better with intuition. This definition can be applied to a distribution over models,

but it also applies when there is just a single model M, and I start with this simpler case.

The entropy $H_M(D) = E[-\log P_M(D)]$ associated with a model M of observation D defines the expected negative log probability of an observation D. I define a specific observation d to be *surprising* if its *information*, $-\log P_M(d)$, is much greater than the entropy:

$$-\log(P_M(d)) \gg H_M(D).$$

Consider the example of a perfect fair die with six outcomes. No outcome can be surprising since the information of each outcome is equal to the entropy of the die. On the other hand, if we acknowledge that there is some minute probability of a die landing on its corner and balancing, then a better estimate of the probability of each face is $\frac{1}{6} - \epsilon$, and the probability of landing on a corner extremely small. If the die does in fact land on a corner, it is not the rarity of this event that makes it surprising, but the probability *relative to the entropy of the die*. That is, the event is thousands of times less likely than what we expected, which is $\frac{1}{6} - \epsilon$.

In the Bayesian setting, where the model M is unknown, this definition of surprise is extended by simply computing expectations over the unknown models:

$$E_{\mathcal{M}}[-\log(P_M(d))] \gg E_{\mathcal{M}}[H_M(D)].$$

Intuitively, a "surprising event" is still an outcome whose probability is far lower (and information is much higher) than what was expected. And again, when all events are a uniform low probability, there can be no surprise.[2]

In practice, rather than trying to define a precise probability for extremely rare events, events that are not part of a model are simply defined as having extremely

[2]This addresses the case of "television snow" that Baldi raises. When snow occurs in the context of a low entropy process, it is surprising, since it has much lower probability. However, when it occurs in the context of a snow distribution, it has probability equal to the other outcomes.

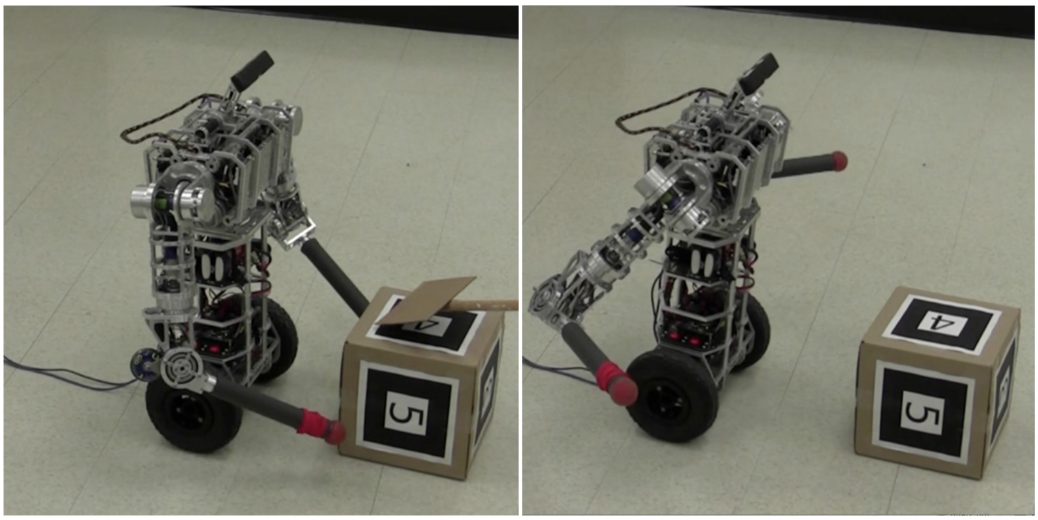

Figure 3.18. The uBot-6 mobile manipulator performing a "what's up?" gesture to convey that it is surprised after an unexpected event.

low probability. For example, while the robot is grasping an object if someone covered the robot's camera and such a transition is not modeled in any of the object models, the observation will be inconsistent with what the robot expects (i.e., has an extremely low probability) and therefore be classified as a surprise transition. During the surprise transition a new ATG model that includes this new transition is created by extending the ATG model with the highest belief in memory. If an identical transition reoccurs the robot will be able to expect the outcome with the new model.

Since surprise transitions are unpredictable and do not lie within a bounded set of possible outcomes, they are handled by resetting the belief among all aspect nodes to the prior distribution. If a possible plan that will lead to the goal still exists, the robot will continue to accomplish the task. In this experiment, when a surprise transition occurs the robot also shows a "What's up?" gesture that conveys confusion as in Figure 3.18. This gesture has proven to be useful for indicating the robot's current state in experiments.

The action selection algorithm that uses the ATG to achieve goals and handle surprise transitions is described in Figure 3.19. Here, $\mathcal{M}$ represents a set of ATG

models stored in the robot memory and x_{target} is the target aspect given to the robot. The function InitializeBelief($\mathcal{M}$) initializes prior probability specified in the set of ATG models $\mathcal{M}$. The function InferAspects($\mathcal{M}, \overline{bel}$) infers a set of possible aspects $X_{candidate}$ based on the belief $\overline{bel}$. The function GetObservation($X_{candidate}$) returns the current observation $z_{current}$ and the corresponding feature positions in R^3 f_{pose} from active feature detectors derived from the set of possible aspects $X_{candidate}$. The function BayesFilterObs($\overline{bel}, z_{current}$) refers to part of the Bayesian filtering algorithm that updates the belief given an observation. The function Surprise(bel) returns true if the belief in all aspect nodes are zero. If a surprise transition occurs, the robot shows the "What's up?" gesture, backs up and move arms to the ready pose. The function CreateNewModel($\mathcal{M}, \overline{bel}, a, z_{current}$) creates a new ATG model by adding a new aspect node representing $z_{current}$ and an action edge representing action a to the ATG model with the highest belief in $\mathcal{M}$. The belief over aspects are reset to the prior distribution through function ResetBelief($\overline{bel}, \mathcal{M}$). The function InferAction($\mathcal{M}, x_{target}, bel$) finds a shortest path to the target aspect x_{target}. The function then returns the first action on the shortest path. ExecuteAction(a, f_{pose}) executes the action based on information stored in the action edge a and feature positions f_{pose}. The function BayesFilterAct(bel, a) refers to part of the Bayesian filtering algorithm that updates the belief given an action.

3.5.3 Experiments

In order to evaluate the introduced techniques that handle randomness and uncertainty, two different experiments are performed. For both cases, the uBot-6 mobile manipulator has to perform simple manipulation tasks. The performance is compared with and without the presented techniques.

```
1: procedure ALGORITHM(x_target, M)
2:     bel = INITIALIZEBELIEF(M)
3:     while true do
4:         X_candidate = INFERASPECTS(M, bel)
5:         z_current, f_pose = GETOBSERVATION(X_candidate)
6:         bel = BAYESFILTEROBS(bel, z_current)
7:         if SURPRISE(bel) then
8:             WHAT'SUPGESTURE(())
9:             CREATENEWMODEL(M, bel, a, z_current)
10:            bel = RESETBELIEF(bel, M)
11:            BACKUPANDREADYPOSE()
12:            continue
13:        end if
14:        a = INFERACTION(M, x_target, bel)
15:        EXECUTEACTION(a, f_pose)
16:        bel = BAYESFILTERACT(bel, a)
17:    end while
18: end procedure
```

Figure 3.19. Algorithm for achieving goal aspect and handling surprise transitions.

3.5.3.1 Settings

The goal of task is to manipulate an ARcube until the requested features are observed on the corresponding faces. An ARcube is a cube box with a different ARtag on each of its face as shown in Figure 3.16. An ARtag is a square fiducial marker commonly used in robotics due to ease of use and robust detection. In this work an ATG model representing an ARcube has 174 nodes and 408 edges.

3.5.3.2 Surprise Recovery

For the first experiment, the model is provided with a fairly detailed ATG model of the handled ARcube. The robot has to reach the desired state with ARtags 1 and 2 on the top and front faces of the cube respectively. The robot has to first perform this task solely based on the provided model and the contained transitions. Despite the model being fairly complete, a small perturbation in the visual and haptics sensors, as well as slightly different dynamics when manipulating the ARcube can result in an observation not contained in the model or not agreeing with the model (e.g. an

action resulting in new outcome). Without the ability to recover, reaching such a state, the robot will not be able to complete the task.

The robot is repeatedly provided with random configurations of the ARcube and the performance of both methods is compared. The robot is able to complete the task reliably 10 out of 10 times when provided with the ability to recover from unforeseen observations. Without this capability, the robot succeeds 5 out of 10 runs.

3.5.3.3 Error Detection Through Fine-Grained Actions

In the second experiment, the robot is presented with an ARcube such that one flip action of the cube will lead to the desired goal state. The ARcube is then perturbed during the grasping action to cause a grasp failure. The performance of the robot is evaluated based on the number of actions it needed to complete the task despite this perturbation.

Two different setups are tested. In the first setup the robot has access to a "flip" macro which combines four primitive actions described in Subsection 3.5.1 into one macro action. In the second setup, the robot uses the fine-grained actions and aspects directly. The experiment is performed 10 times for each setup. Using the flip macro, the robot needs 16.1 ± 3.37 primitive actions to complete the task, whereas with the fine-grained actions it only needs 10.6 ± 0.69 primitive actions. The difference in the required number of primitive actions is due to the capability of detecting the error much earlier and reacting appropriately when using fine-grained actions and aspects.

3.6 Conclusions

This chapter introduces the aspect transition graph (ATG), a memory model that represents how actions change observations and can be used to capture the affordances of the environment. This viewer centered model categorizes different observations into a subset of aspects based on interactions instead of just visual appearance. Based on

this ATG model, an incremental learning framework for building memories through interaction can be constructed.

First, I discuss how this memory model can be used to handle uncertainty in a partially observed environment. A Bayes framework that performs inference over incomplete object models is presented. The strengths of combining the ATG representation with a belief-space planner is then demonstrated on a problem formalized as simultaneous object modeling and recognition. Second, I propose that a sequence of controllers that form a "funnel-slide-funnel" structure based on the ATG model can have high rates of success even when open-loop controllers are within the sequence. To demonstrate this, a visual servoing controller that funnels the current observation to a memorized aspect is introduced. This proposed structure decreases the average final position error significantly. Third, how interacting with objects can be performed without explicit pose information using the ATG model is described. This approach is tested on a drill grasping task on Robonaut-2. Last, I describe how failures in stochastic robotic actions can be detected and handled properly using ATG models. A general framework that stores fine-grained ATGs for early error detection is introduced. Detected failures are further categorized into random transitions or surprise transitions based on how it is perceived by the robot. These two types of uncertainties are then handled and recovered separately. It is shown that the overall framework increases the robustness towards handling uncertainties and decreases the number of actions needed on manipulation tasks experimented on the uBot-6 mobile manipulator.

CHAPTER 4

HIERARCHICAL ASPECT REPRESENTATION

In the previous chapter, the aspect transition graph (ATG), which represents how actions lead to new observations using a directed multi-graph consisting of aspect nodes and action edges, is introduced. An aspect is defined as an observation that is memorized by the robot. Aspect representations are generated by mapping observations to an aspect space and can provide controllers information for interacting with the environment. An aspect representation that captures the affordance of an observation is critical for the framework to handle a variety of situations beyond the given example. In this chapter, an aspect representation that supports manipulation in a hierarchical fashion and captures the essential affordances is proposed. A hierarchical correspondence where higher level features are associated with higher level actions that require less accuracy is introduced. This design allows the robot to plan more efficiently in different levels of abstraction based on the task requirement; an action that does not require high precision would only need a rough location reference.

In Section 4.1, the hierarchical convolutional neural network (CNN) feature that supports manipulation is introduced. An approach that associates these features with grasp configurations is then described. With this approach, Robonaut-2 can successfully grasp novel objects of the same class reliably. In Section 4.2, an aspect representation based on hierarchical CNN features is introduced. The robustness of this aspect representation is tested on the Washington RGB-D Objects Dataset and achieves state of the art result for instance pose estimation.

4.1 Hierarchical CNN Features

Although convolutional neural networks (CNNs) have outperformed other approaches on object recognition benchmarks, identifying the category of an object is not enough for manipulation. Knowing visual feature locations in 3D is also crucial for interacting with it precisely. The key observation is the following. A lower layer CNN filter often represents a local structure of a more complicated structure represented by a higher layer filter. The hierarchical CNN feature identifies local structures at each level that are related to a hierarchical "part-based" representation of an object that each afford opportunities for control and interaction. In this work, the arm controllers are associated with the hierarchical CNN features in the 4th convolutional layer while the hand controllers are associated with hierarchical CNN features in the 3rd convolutional layer. By associating features and controllers based on their corresponding hierarchy, combinatorial explosion can be avoided.

In this section, the hierarchical CNN features are first introduced. How consistent features can be identified and used to support manipulation is then explained in Subsection 4.1.2. Two sets of experiments are conducted. In Subsection 4.1.3, the difference between the calculated grasp points and the ground truth in the R2 grasping dataset is analyzed. In Subsection 4.1.4, this approach is tested on grasping novel objects using Robonaut-2 [14]. Results show that these features that consider hierarchical relationships can increase accuracy in cluttered scenario and outperform a point cloud based approach on an object grasping task.

4.1.1 Definitions

Convolutional neural networks (CNNs) are a class of deep neural networks that contains more then one convolutional layers. This work uses a CNN model similar to Alexnet [48] introduced by Yosinski [101] and implemented with Caffe [41]. This network is trained on the ImageNet [76] and has five convolutional layers followed

by two fully connected layers. A filter is defined as a kernel used for the convolution operation in each convolutional layer. In the following, f_i^m is used to represent the i^{th} filter in the m^{th} convolutional layer. The convolution operation between a filter and the previous layer generates a two dimensional grid of responses called the activation map. The final layer of a CNN or a set of activation maps of a single convolutional layer are often considered as CNN features that are used as inputs for a wide range of computer vision tasks.

CNN features are by nature hierarchical; a filter in a higher layer with little location information is a combination of lower level filters with higher spatial accuracy. For example, filter f_{87}^5 in the conv-5 layer in the CNN represents a box shaped object. This filter is a combination of several filters in the fourth convolution (conv-4) layer. Among these filters, f_{190}^4 and f_{133}^4 represents the lower right corner and top left corner of the box respectively. Filter f_{190}^4 is also a combination of several filters in the third convolution (conv-3) layer. Among these filters, f_{168}^3 and f_{54}^3 represents the diagonal right edge and diagonal left edge of the lower right corner of the box. The activation map of filter f_{168}^3 will respond to all diagonal edges in an image, but if we only look at the subset of units of f_{168}^3 that has a hierarchical relationship with f_{190}^4 in the conv-4 layer and f_{87}^5 in the conv-5 layer, local features that correspond to meaningful parts of a box-like object can be identified. Instead of representing a feature with a single filter in the conv-3 layer, the proposed approach uses a tuple of filters to represent a feature, such as (f_{87}^5, f_{190}^4, f_{168}^3) in the previous example. I call such features *hierarchical CNN features*. Within a hierarchical CNN feature, a higher level filter is called the parent filter of a lower level filter. Figure 4.1 shows the visualization of several hierarchical CNN features identified in cuboid and cylindrical objects, including the features in the previous example.

Given an observation, a set of hierarchical CNN features that have high activation can be identified. This set of features can be localized through one single forward

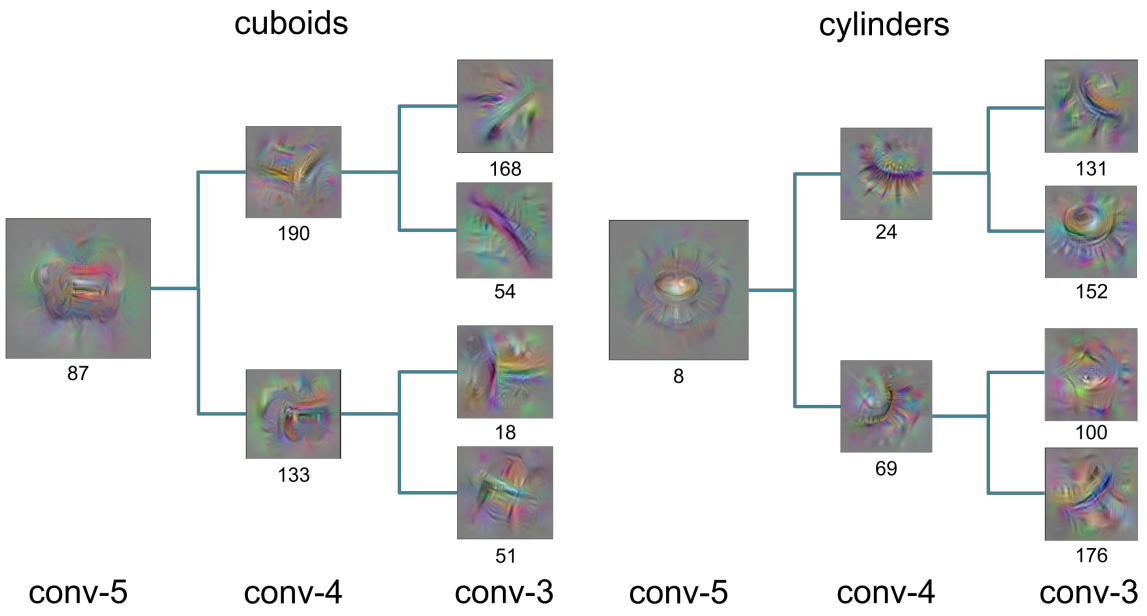

Figure 4.1. Hierarchical CNN feature visualizations among cuboid objects (left) and cylinders (right). Each square figure is the visualization of a CNN filter while the edges connect a lower layer filter to a parent filter in a higher layer. The numbers under the squares are the corresponding filter indices. Filters are visualized using the visualization tool introduced in [101]. Notice that the lower level filters represent local structures of a parent filter.

path and one backward path for each feature on the network. For each observed point cloud, the flat supporting surface is segmented using Random Sample Consensus (RANSAC) [22] and create a 2D mask that excludes pixels that are not part of the object in the RGB image. This mask is dilated to preserve the boundary of the object. During the forward pass, for each convolution layer the responses that are not marked as part of the object according to the mask is zeroed out. This approach removes filter responses that are not part of the object. The masked forward pass approach is used instead of directly segmenting the object in the RGB image to avoid sharp edges caused by segmentation.

During the backward pass, the top N^j filters in the j^{th} convolutional (conv-j) layer that have the highest sum of log responses are identified. For each of these filters, the maximum response of the activation map is identified and all other responses are

zeroed out. The max unit response is then backpropagated to the conv-$(j-1)$ layer. The top N^{j-1} filters that have the highest sum of log partial derivatives obtained from backpropagation is then identified. For each of these N^{j-1} filters, the gradient of backpropagation is calculated and everything except for the maximum partial derivative is zeroed out. The same procedure is used recursively to find N^{j-2} filters in the conv-$(j-2)$ layer. This process back traces along a single path recursively and yields a tree structure of hierarchical CNN features. This set of hierarchical CNN features can be located in 3D by backpropagating to the image and mapping the mean of the response location to the corresponding 3D point in the point cloud. The *response* of a hierarchical CNN feature is defined as the maximum gradient of the lowest layer filter in the tuple. Figure 4.2 shows an example of results of performing backpropagation along a single path to the image layer from features in the conv-5, conv-4, and conv-3 layers. The conv-3 layer features can be interpreted as representing lower level features such as edges and corners of the cuboid object.

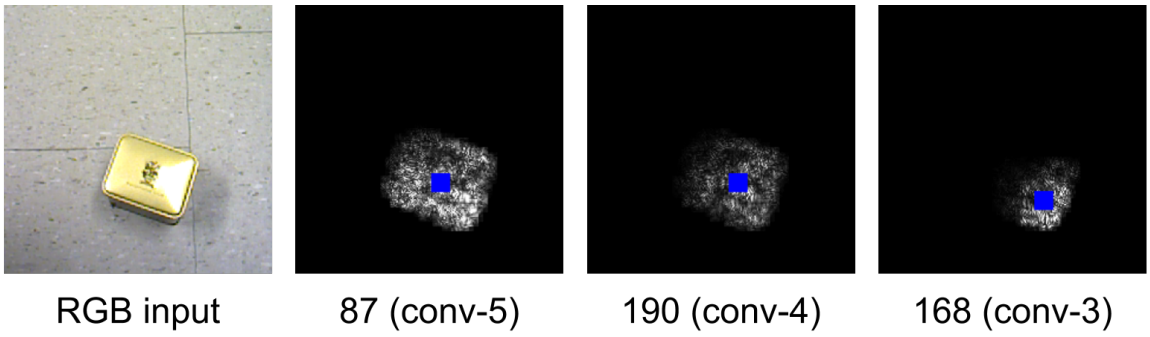

Figure 4.2. Localizing features in different layers. The color image is the input image. The following images from left to right represent the location map of hierarchical CNN features (f^5_{87}), (f^5_{87}, f^4_{190}), and ($f^5_{87}, f^4_{190}, f^3_{168}$) obtained from backpropagating the feature response along a single path to the image layer. The blue dot in each location map represents the mean of the response locations. The mean response locations of conv-3 layer features are located closer to edges and corners of the cuboid object compared to the conv-4 and conv-5 features. The conv-3 layer features can be interpreted as representing local structures of the cuboid object.

4.1.2 Supporting Grasping

In this work, a solution for posturing the robot hand and arm for grasping based on visual information is proposed. Experiments conducted by Goodale and Milner [28] have shown that humans are capable of pre-shaping their hands to grasp an object based exclusively on visual information from an object. There are, in general, many possible kinds of grasps for each object; here the focus is on generating a specific grasp that is similar to a demonstrated grasp. To learn how to grasp a previously unseen object, hierarchical CNN features that represent meaningful characteristics of the geometric shape of an object are identified to support grasping. In this work, the network is trained on ImageNet and not retrained for the grasping task. Therefore only require a small amount of examples. In this subsection, an approach that identifies hierarchical CNN features that activate consistently among the training data is first proposed. How grasp points, Cartesian grasp positions of end effectors, are generated based on offsets between features and robot end effectors is then further described.

4.1.2.1 Consistent Features

Given a set of grasp records that demonstrates grasping objects with similar shapes, the goal is to find a set of hierarchical CNN features that activate consistently. The assumption is that some of these features represent meaningful geometric information regarding the shape of an object that supports grasping. For each training example, a masked forward pass described in Subsection 4.1.1 is first processed. Next, filters that activate consistently over similar object types (*consistent filters*), in the fifth convolution (conv-5) layer are identified. The top N^5 filters that have the highest sum of sum of log responses $\sum_{e \in E} \sum_{a \in A_i^{5,e}} \log a$ among all the training examples $e \in E$ of the same type of grasp are identified, where $A_i^{5,e}$ is the activation map of filter f_i^5 of example e, and f_i^m represents the i^{th} filter in the m^{th} convolutional

layer. In this work, only one type of grasp is demonstrated for the same type of object. I observe that many filters in the conv-5 layer represent high level structures and fire on box-like objects, tube-like objects, faces, etc. Knowing what type of object is observed can determine the type of the grasp but is not sufficient for grasping, since boxes can have different sizes and presented in different pose. However, if lower level features such as edges or vertices exist, robot fingers can be placed relative to them. The proposed hierarchical CNN feature represented as a tuple of filters can be used to represent features that have such hierarchical relationship. A higher layer filter in the tuple can be used to represent object level features while a lower layer filter can be used to represent low level features that support manipulation. Defining a feature based on both low level and high level features restricts the low level feature to be part of the high level feature.

To identify lower layer filters that fire consistently and represent features that are part of what the top N^5 conv-5 layer filters selected represent, the max response of the activation map $A_i^{5,e}$ for each of the consistent filters f_i^5 in the conv-5 layer for each example is first identified. For each consistent filter f_i^5 and each example e, all responses except for the maximum $a_{max}^{5,e}$ are zeroed out and backpropagated to obtain the gradient of the max response with respect to the conv-4 layer: $G^{4,e} = \partial a_{max}^{5,e}/\partial$conv-4. The filters that contribute to f_i^5 consistently in the conv-4 layer are identified by picking the top N^4 filters that have the highest sum of sum of log gradients $\sum_{e\in E}\sum_{g\in G_j^{4,e}} \log g$, where $G_j^{4,e}$ represents components of the gradient $G^{4,e}$ that corresponds to filter f_j^4 for example e. With the same procedure, the top N^3 filters that contribute to f_j^4 consistently is identified by calculating the gradient with respect to the conv-3 layer: $G^{3,e} = \partial g_{max}^{4,e}/\partial$conv-3 for each example, where $g_{max}^{4,e}$ is the maximum partial derivative of $G_j^{4,e}$. This process traces backward along a single path recursively and yields a tree structure of consistent hierarchical features that are activated consistently among the same type of grasp within the training examples.

4.1.2.2 Generating Grasp Points

For each target object during testing, a set of grasp points for the index finger, thumb, and the arm are generated. Grasp points are the locations that the corresponding end effectors should be positioned for a successful grasp and are calculated based on a set of relative positions between 3D features and end effectors. For each type of object a set of possible grasp offsets for each end effector and the corresponding feature is stored. A grasp offset is determined by the relative position from a robot end effector to a feature based on a demonstrated grasp. Although modeling a type of grasp based on the offsets between grasp points and feature positions may not result in consistent grasp points when dealing with objects of different sizes at different orientations, there usually exists a subset of feature points that are close to the end effector consistently therefore result in smaller offset variances. In this work, the robot hand frame is associated to features in the conv-4 layer and the robot index finger and thumb is associated to features in the conv-3 layer. This mapping is inspired by the observation that humans are capable of moving their arms toward the object for grasp without recognizing detailed features; the precise locations of low level features are only needed when the finger locations need to be accurate. To my knowledge, this is the first work that associates features in different CNN layers with controllers that engage different kinematic subchains in the hand/arm systems. Figure 4.3 shows the overall architecture and how such hierarchical CNN features are mapped to a point cloud to support manipulation.

Not all features that fire consistently on the same object type are good features for planning actions. For example, when grasping a box, positioning the index finger, which contacts the back edge of the box, relative to the front edge of the box will result in positions with high variance, since the size of the box may vary. In contrast, positioning the index finger relative to the back edge of the box will result in a lower position variance. For each end effector position, the top N hierarchical CNN features

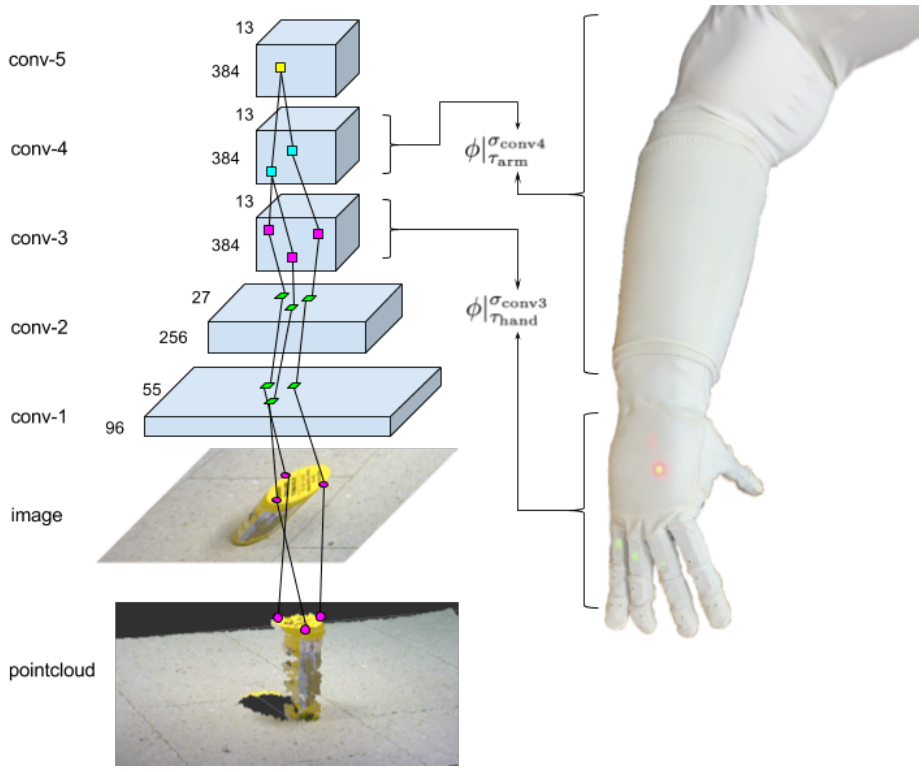

Figure 4.3. Overall architecture of the proposed system. The input for this example is the RGB image and point cloud of a yellow jar. Tuples of the yellow dots (conv-5), the cyan dots (conv-4), and the magenta dots (conv-3) represent hierarchical CNN features. These features can be traced back to the image input and mapped to the point cloud to support manipulation. $\phi|^{\sigma_{\text{conv3}}}_{\tau_{\text{hand}}}$ represents the function that controls hand motor resources τ_{hand} based on conv-3 layer features σ_{conv3} and $\phi|^{\sigma_{\text{conv4}}}_{\tau_{\text{arm}}}$ represents the function that controls arm motor resources τ_{arm} based on conv-4 layer features σ_{conv4}. ϕ represents the potential function computed from input σ whose derivative with respect to motor resources τ constitute a controller.

that have the lowest 3D position variance among training examples and have the same parent filter in the conv-5 layer are selected. The other features are then removed. The final set of features have the same conv-5 filter f_i^5. It is shown that restricting all the hierarchical CNN features to have the same parent filter allows this approach to perform well in a cluttered scenario.

During testing, the hierarchical CNN features associated with grasp offsets are first identified. The 3D positions of these features are then located through backpropagation to the 3D point cloud. A set of possible grasp positions are then calculated

based on the grasp offsets and the 3D positions of the corresponding hierarchical CNN features. The grasp points for the robot hand frame, and end points for the thumb and index finger are then determined by the weighted mean position of the corresponding set of possible grasp positions with the feature responses as weights. Figure 4.5 shows examples of the grasp points and the set of possible grasp positions on different objects.

4.1.3 Experiments on the R2 Grasping Dataset

In this subsection, the performance of the proposed approach on a grasping dataset is analyzed. First, the R2 grasping dataset is introduced. The accuracy of generated grasp points based on cross-validation is then evaluated. Last, approaches with and without the hierarchical CNN features are compared.

4.1.3.1 Dataset

In this work, the R2 grasping dataset that contains grasp records of each demonstrated grasp is created. A grasp record contains the point cloud and RGB image of the target object observed from the robot's viewpoint, and the Cartesian pose of each joint in the Robonaut-2 hand in the camera frame. The data is collected using an Asus Xtion camera and the Robonaut-2 simulator [15]. The object is placed on a flat surface where the camera is about 70 cm above and looking down at a 55 degree angle. The left robot arm and each finger of the left hand are manually adjusted so that the robot hand can perform a firm grasp on the object. For cuboid objects, the thumb tip and index finger tip are adjusted to the front and back faces of the cuboid and about 3cm away from the left edge of the face. For cylindrical objects, the thumb tip and index finger are adjusted to perform an enveloping grasp on the object.

A total of 120 grasping examples of twelve different objects are collected. Six of the objects are cylindrical and six of them are cuboids. The same object is presented at different orientations and under different lighting conditions. The collection interface

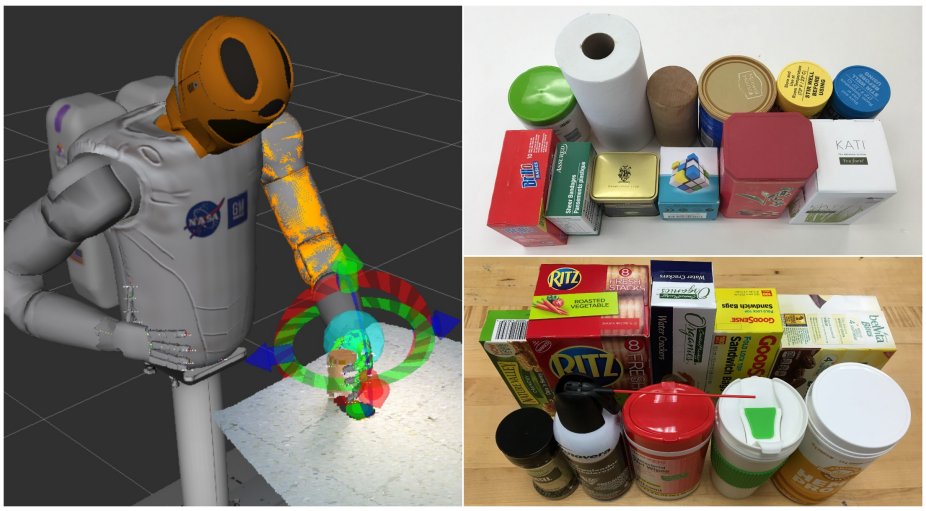

Figure 4.4. Left: the data collection interface where the robot arm and hand is adjusted to the grasp pose. Right top: The set of objects used in the R2 grasping dataset. Right bottom: The set of novel objects used in the grasping experiment.

and the objects used are shown in Figure 4.4. In addition to grasping examples with a single object, 24 grasping examples in cluttered scenarios are also created. Twelve of them include a single cylindrical object and twelve of them include a single cuboid object. The joint poses of the hand while grasping these objects are also recorded.

4.1.3.2 Cross-Validation Results

Cross-validation is applied by leaving out one object instance at a time during training and testing on the left out object by comparing the calculated grasp points to the ground truth. The distance between the example position and the targeted position of the hand frame, index finger tip, and thumb tip is calculated and shown in Table 4.1. The average grasp position error for the hand frame is higher than the thumb and index finger; this is likely because the positions of local features alone are not sufficient to predict an accurate position for the hand frame. However, since the hand frame is not contacting the object, its position is less crucial for a successful grasp. Figure 4.5 shows a few results of cross-validation on different objects with different pose and lighting. Similar to the training data, the cuboid objects are

grasped at positions closer to the left side while the cylindrical objects are grasped such that the fingers would wrap around the cylinder.

Table 4.1. Average grasp position error on cylindrical and cuboid objects in meters.

	cylindrical objects						
left hand	cetaphil jar	wood cylinder	maxwell can	blue jar	paper roll	yellow jar	average
thumb tip	0.0197	0.0164	0.0134	0.0202	0.0136	0.0147	0.0163
index tip	0.011	0.0111	0.023	0.017	0.0155	0.0126	0.015
hand frame	0.0128	0.0393	0.0234	0.0183	0.0328	0.0173	0.024
	cuboid objects						
left hand	cube box	redtea box	bandage box	twinings box	brillo box	tazo box	average
thumb tip	0.0094	0.0101	0.0093	0.0095	0.0088	0.0111	0.01
index tip	0.0135	0.0207	0.0278	0.0134	0.0098	0.0157	0.0168
hand frame	0.0241	0.0203	0.0195	0.0177	0.0177	0.0285	0.0213

4.1.3.3 Comparison

To evaluate the proposed hierarchical CNN feature (hier-feat), cross-validation results with four alternative approaches that do not consider relationships between layers are compared. The first approach is a baseline that uses the same set of features identified by the proposed approach but only considers the lowest level filter of the hierarchical CNN features, therefore removing relationship with higher level CNN filters. The grasp points are then generated based on offsets to these individual filters in each layer. The second approach (indv-filter) also associates grasp points with individual CNN filters instead of hierarchical CNN features but learns the set of filters that fire consistently instead of using the same set of features used by the proposed approach. Similar to the second approach, the third alternative (conv5-filter) identifies individual CNN filters instead of hierarchical CNN features but only considers filters in the conv-5 layer. The fourth approach (conv5-max) identifies the top five consistent filters in the conv-5 layer and uses the one that has the max

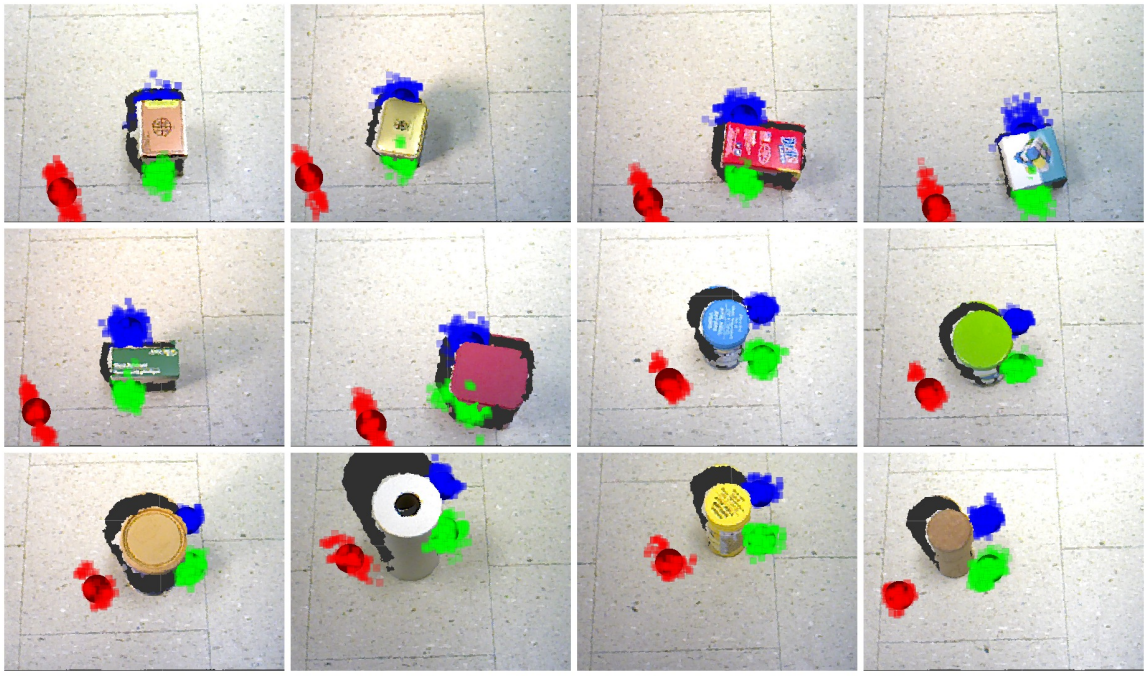

Figure 4.5. Sample cross-validation results for single object scenario. The red, green, and blue spheres represent the calculated grasp points for the hand frame and endpoint positions for the thumb and index finger of the left robot hand. The grasp points are the weighted mean of the colored dots that each represents a possible grasp position based on one training example. Notice that for the cuboid object the grasp points for the thumb and index finger are located on the opposing face and about 3cm away from the left edge of the face as it was trained. For the cylinder object the grasp points for the thumb and index finger are on the right side of the cylinder to form an enveloping grasp. The black pixels are locations behind the point cloud that are not observable.

response during testing. To make the comparison fair, filters other than the top N hierarchical CNN features that have the lowest position variance among training examples are also removed for the first three comparative approaches. $N^5 = N^4 = N^3 = 5$ and $N = 15$ is used in this experiment.

The results are shown in the first row of Table 4.2. The proposed approach performs better than all four alternatives. However the difference is not significant, this is because the lower level filter that has the maximum response is mostly the same with or without restricting it to have the same parent filter when only one object is presented. In the next test, it is shown that associating low level filters with high level filters has a greater advantage when low level features may be generated from different high level structures, i.e., when there is clutter present. The fact that the proposed approach outperforms the conv5-max approach shows the benefit of using lower level features to higher level features on planning actions. In the absence of clutter, the conv5-filter approach performs well because although filters in the conv-5 layer are more likely to represent higher level object structures, many of them also represent low-level features like corners and edges.

Hierarchical CNN features are most useful when the scene is more complex and the same lower layer filter fires at multiple places. Since the proposed approach limits the filters in the conv-3 layer to have the same parent filter in the conv-5 layer, only lower layer features that belong to the same high level structure are considered. These five approaches are further tested on the cluttered test set. A test case is considered successful if the distance errors of the thumb tip and index finger tip are both less than 5cm and the hand frame error is less than 10cm. The results are shown in the second row of Table 4.2. Figure 4.6 shows a few example results on cluttered test cases. The proposed approach performs significantly better than the baseline, indv-filter, and conv5-filter approaches since filters may fire on different objects without

constraining it to a single high level filter. Figure 4.7 shows two comparison results between the proposed approach and the baseline approach.

Table 4.2. Comparison on alternative approaches.

	hier-feat	base-line	indv-filter	conv5-filter	conv5-max
Single Object Experiment: cross validation average grasp position error (cm)	1.719	1.805	2.114	1.755	2.138
Cluttered Experiment: number of failed cluttered cases (24 total)	2	20	13	19	2

4.1.4 Experiments on Robonaut-2

This subsection describes the evaluation of the proposed pre-shaping algorithm based on the percentage of successful grasps on a set of novel objects on Robonaut-2 [14]. Details on the experimental setting, the hierarchical controller used for pre-shaping, and results are explained.

4.1.4.1 Settings

For each trial, a single object in the novel object set is placed on a flat surface within the robot's reach. Given the object image and point cloud, the robot moves its wrist and fingers to the pre-shaping pose. After reaching the pre-shaping pose, the hand changes to a pre-defined closed posture and tries to pick up the object by moving the hand up vertically. A grasp is considered to be successful if the object did not drop after the robot tries to pick it up. A total of 100 grasping trials on 10 novel objects with the proposed approach and a comparative baseline approach are tested. The novel objects used in this experiment are shown in Figure 4.4.

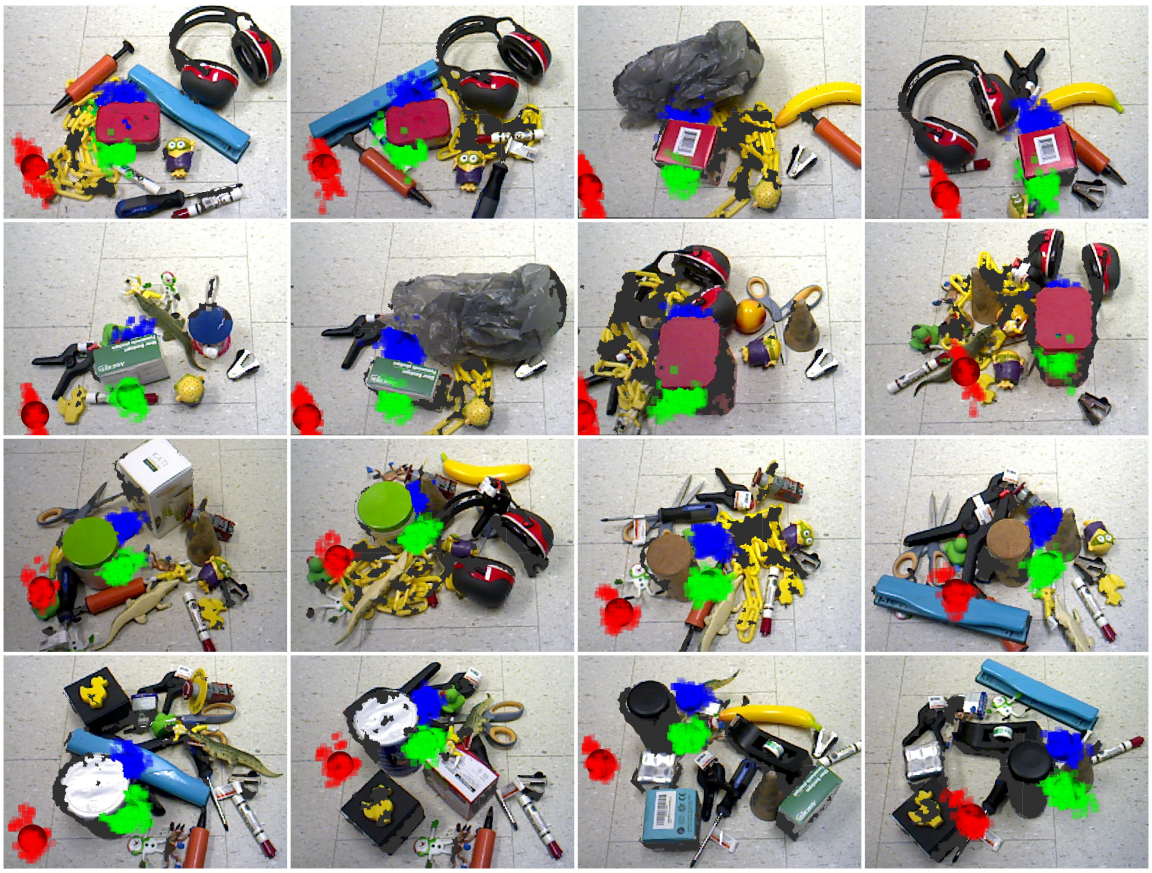

Figure 4.6. Examples of grasping in a cluttered scenario. The red, green, and blue spheres represent the grasp points of the hand frame, thumb tip, and index finger tip of the left robot hand. The grasp points are the weighted mean of the colored dots that each represents a possible grasp position based on one training example. The top two row is trained on grasping cuboid objects and the bottom two row is trained on grasping cylindrical objects. Notice that this approach is able to identify the only cuboid or cylinder in the scene and generate grasp points similar to the training examples.

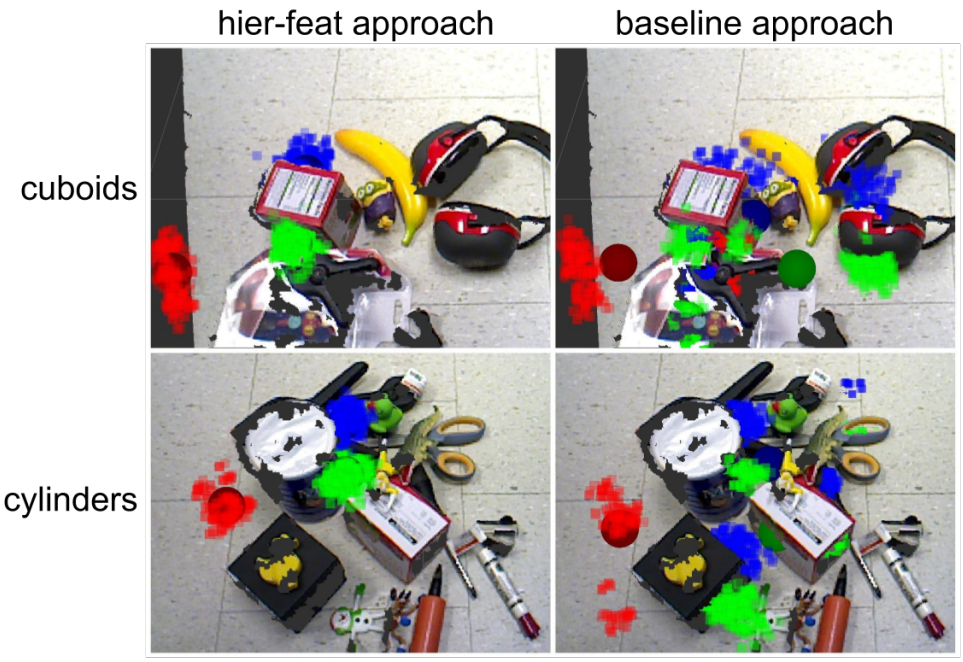

Figure 4.7. Comparison in a cluttered scenario. Notice that the colored dots are scattered around in the baseline approach since the highest response filter in conv-3 or conv-4 layer are no longer restricted to the same high level structure.

4.1.4.2 Hierarchical Controller

A hierarchical controller constructed from hierarchical CNN features in different CNN layers is implemented to reach the pre-shaping pose. Given the object image and point cloud, this approach generates targets for the robot hand frame, index finger tip, and thumb tip. The hand frame target is determined based on hierarchical CNN features in the conv-4 layer while the thumb tip and index finger tip target is determined based on hierarchical CNN features in the conv-3 layer. The pre-shaping is executed in two steps. First, the arm controller moves the arm such that the distance from the hand frame to the corresponding grasp point is minimized. Once the arm controller converges, the hand controller moves the wrist and fingers to minimize the sum of squared distances from the index finger tip and thumb tip to their corresponding target.

These controllers are based on the control basis framework [36] and can be written in the form $\phi|_{\tau}^{\sigma}$, where ϕ is a potential function that describes the sum of squared distances to the targets, σ represents sensory resources allocated, and τ represents the motor resources allocated. In this work, the hand controller is represented as $\phi|_{\tau_{\text{hand}}}^{\sigma_{\text{conv3}}}$, where τ_{hand} is the hand motor resources and σ_{conv3} is the conv-3 layer hierarchical CNN features; the arm controller is represented as $\phi|_{\tau_{\text{arm}}}^{\sigma_{\text{conv4}}}$, where τ_{arm} is the arm motor resources and σ_{conv4} is the conv-4 layer hierarchical CNN features.

4.1.4.3 Results

The proposed algorithm is compared to a baseline point cloud approach that moves the robot hand to a position where the object point cloud center is located at the center of the hand after the hand is fully closed. The results are shown in Table 4.3. Among the 50 grasping trials only one grasp failed with the proposed approach due to a failure in controlling the index finger to the target position. This demonstrates that the proposed approach has a much higher probability of success in grasping novel objects than the point cloud based approach. Figure 4.8 shows Robonaut-2 grasping novel objects during testing.

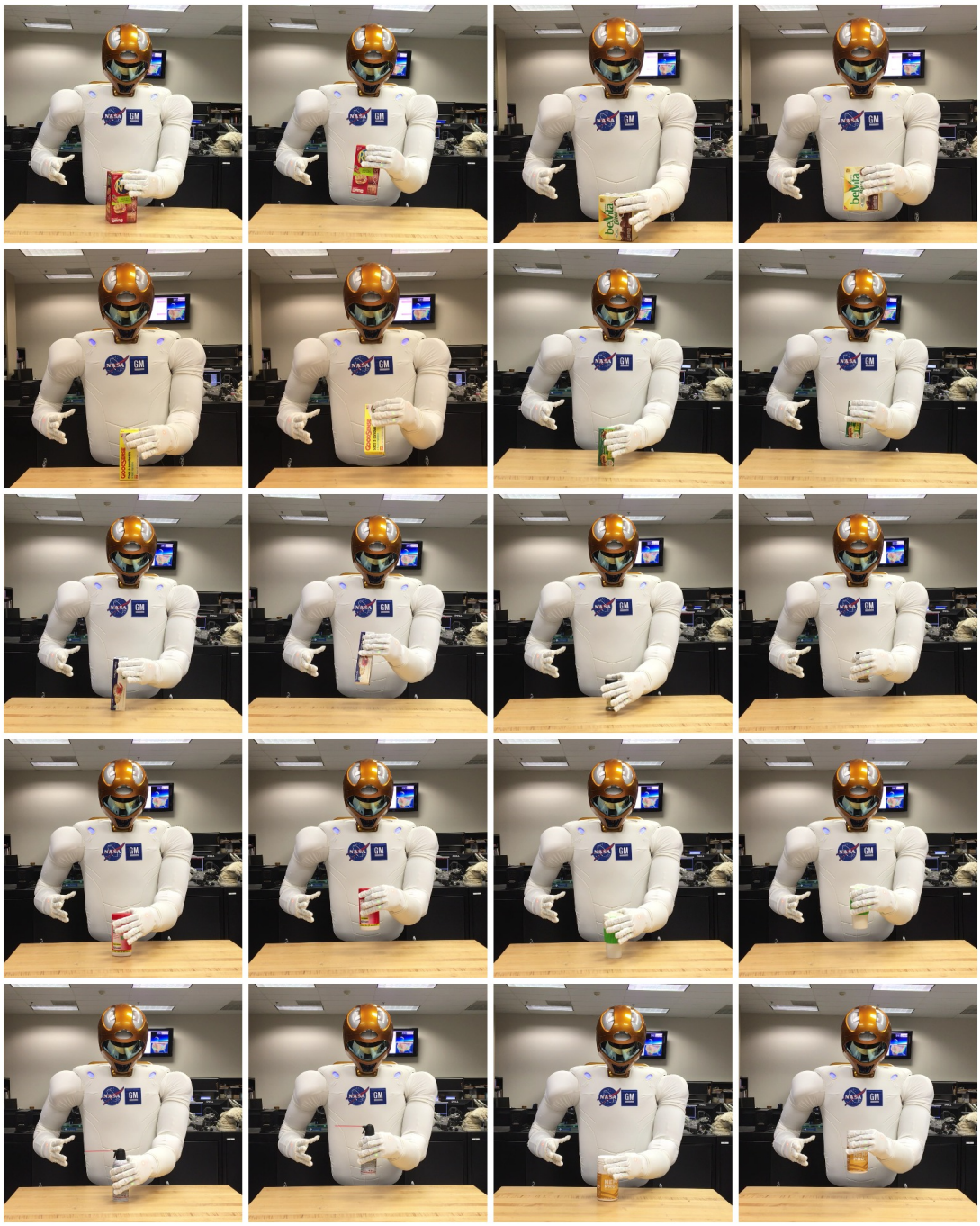

Figure 4.8. Robonaut-2 grasping 10 different novel objects. The first and third columns show the pre-shaping steps while the second and fourth columns show the corresponding grasp and pickup. The cuboid objects are grasped on the faces while the cylinder objects are grasped such that the object is wrapped in the hand.

Table 4.3. Grasp success rate on novel objects based on 5 trials per object.

	cylindrical objects					
	tumbler	wipe package	basil container	hemp protein	duster	average
point cloud approach	40%	60%	0%	20%	40%	36%
hier-feat (our's)	100%	100%	100%	100%	100%	100%
	cuboid objects					
	cracker box	ritz box	bevita box	bag box	energy bar box	average
point cloud approach	80%	20%	60%	60%	60%	52%
hier-feat (our's)	100%	80%	100%	100%	100%	96%

4.2 Aspect Representation

In this section, a novel aspect representation that supports manipulation and captures the essential affordances of an object based on sensory feedback is introduced. In a traditional planning system, robots are given a pre-defined set of actions that take the robot from one symbolic state to another. However symbolic states often lack the flexibility to generalize across similar situations. The proposed representation is grounded in the robot's observations and lies in a continuous space that allows the robot to handle similar unseen situations. This representation is based on the hierarchical CNN features introduced in the previous section and allows the robot to act precisely with respect to the spatial locations of individual features. A primary step of a system that uses ATGs to determine actions is to match the current observation to an aspect, a stored observation in memory, so that the robot can apply learned actions to the current situation. This work represents an aspect as a set of hierarchical CNN features, an appearance descriptor, a pose descriptor, a location descriptor, and a force descriptor. This representation is used to identify the aspect that affords

the same type of interactions given the current observation and allows the robot to manipulate the object based on the feature locations when combined with an ATG. In the following, how hierarchical CNN features are used to create descriptors is first described. This representation is then evaluated on the Washington RGB-D Objects Dataset and is shown to achieve state of the art results for instance pose estimation.

4.2.1 Descriptors

Based on the response and 3D location of the hierarchical CNN features extracted, this approach generates an appearance descriptor r, a pose descriptor q, a location descriptor l, and a force descriptor f for each aspect. The appearance descriptor r is a set of hierarchical CNN feature responses based on the feature tuple. The assumption is that aspects similar to the current observation have similar appearances and therefore similar hierarchical CNN features and responses.

The pose descriptor q is a set of relative 3D positions in the camera frame between each pair of hierarchical CNN features. If H is the set of all possible hierarchical CNN features and r contains $|h|$ responses of a subset $h \subset H$ of hierarchical CNN features, then q contains $|h| \times |h-1|/2$ XYZ differences. Assuming that aspects similar to the observation should have similar poses, the relative location of the features are used to further distinguish aspects that have the same features but are oriented differently.

The location descriptor l is a set of distances from the centroid of the hierarchical CNN features to a set of pre-defined robot frames. The robot shoulder and hand frames are used in experiments conducted on Robonaut-2 in this dissertation. The location descriptor captures the object position with respect to the robot and can provide information on the reachability of an object.

The force descriptor g is based on force feedback. The load cells in the Robonaut-2 forearms are used to receive force information in the ratchet experiment described in

section 5.4. The observed force values are projected to the body frame at 10Hz and averaged over the preceding one-second interval.

The following describes how to find the aspect x in memory most likely to match the current observation z based on the descriptors. Let $p(x|z)$ be the probability that aspect x is generated from the same state that generates observation z. This probability can be calculated through Bayes' rule $p(x|z) \propto p(z|x) \cdot p(x)$, which the likelihood is modeled as

$$p(z|x) = p(r^z|r^x) \cdot p(q^z|q^x) \cdot p(l^z|l^x) \cdot p(g^z|g^x). \tag{4.1}$$

Here r^z and r^x are the appearance descriptors of the observation z and the aspect x, q^z and q^x are the pose descriptors of z and x, l^z and l^x are the location descriptors of z and x, and g^z and g^x are the force descriptors of z and x.

The probability $p(r^z|r^x)$ is modeled as the geometric mean of the probabilities $p(r_n^z|r_n^x)$ of individual appearance descriptor values r_n^z and r_n^x. The probability $p(r_n^z|r_n^x)$ is modeled as a Generalized Gaussian Distribution (GGD) of the value difference between r_n^z and r_n^x scaled by their sum.

$$p(r^z|r^x) = \Big(\prod_{r_n^x \in r^x \vee r_n^z \in r^z} p(r_n^z|r_n^x) \Big)^{\frac{1}{N}}, \tag{4.2}$$

$$p(r_n^z|r_n^x) = GGD(r_n^x - r_n^z; \alpha = r_n^x + r_n^z), \tag{4.3}$$

where N is the number of appearance descriptors. A missing descriptor value $r_n \notin r$ is considered to be zero. Different individual appearance descriptors r_n refer to different hierarchical CNN feature tuples. The GGD function is defined as

$$GGD(y; \mu, \alpha, \beta) = \frac{1}{Z(\alpha, \beta)} \cdot e^{-(\frac{|y-\mu|}{\alpha})^\beta}, \tag{4.4}$$

where μ is the mean parameter, α is the scale parameter, β is the shape parameter, and $Z(\alpha, \beta)$ is the partition function. μ is set to zero and β is set to 0.1 in this work. A shape parameter β that produces a heavier tail is found to perform better than a standard Gaussian on the Washington RGB-D Objects dataset.

The pose descriptor likelihood $p(q^z|q^x)$, location descriptor likelihood $p(l^z|l^x)$, and force descriptor likelihood $p(g^z|g^x)$ are modeled similarly and the equations are shown in the following:

$$p(q^z|q^x) = \Big(\prod_{q_n^x \in q^x \wedge q_n^z \in q^z} p(q_n^z|q_n^x) \Big)^{\frac{1}{N}}, \tag{4.5}$$

$$p(q_n^z|q_n^x) = GGD(q_n^x - q_n^z; \alpha = C_q), \tag{4.6}$$

where q_n^x and q_n^z are the individual pose descriptor values, and N is the number of pose descriptors. Different individual pose descriptors refer to different pairs of hierarchical CNN features and XYZ coordinates in the camera frame. C_q is a constant set to 0.1.

$$p(l^z|l^x) = \Big(\prod_{l_n^x \in l^x \wedge l_n^z \in l^z} p(l_n^z|l_n^x) \Big)^{\frac{1}{N}}, \tag{4.7}$$

$$p(l_n^z|l_n^x) = GGD(l_n^x - l_n^z; \alpha = C_l), \tag{4.8}$$

where l_n^x and l_n^z are the individual location descriptor values, and N is the number of location descriptors. Different individual location descriptors refer to different robot frames. C_l is a constant also set to 0.1.

$$p(g^z|g^x) = \Big(\prod_{g_n^x \in g^x \wedge g_n^z \in g^z} p(g_n^z|g_n^x) \Big)^{\frac{1}{N}}, \tag{4.9}$$

$$p(g_n^z|g_n^x) = GGD(g_n^x - g_n^z; \alpha = C_g), \tag{4.10}$$

where g_n^x and g_n^z are the individual force descriptor values, and N is the number of force descriptors. Different individual force descriptors refer to force projections on different axis for different load cells. C_g is a constant set to 100.

4.2.2 Experiments on Pose Estimation

The goal of the proposed aspect representation is to identify an observation in memory that is similar to the current observation and, thus, affords similar actions. The proposed representation is tested on instance pose recognition on the Washington RGB-D Objects dataset [49] under the assumption that an object's pose and affordance are strongly correlated; the same object usually supports the same set of actions when placed in similar orientations. This approach is shown to achieve state of the art results in accuracy.

4.2.2.1 Dataset

The Washington RGB-D dataset contains RGB images, depth images, point clouds, and masks for 300 objects. Each object is placed on a turntable and approximately 250 frames are captured for each elevation angle $(30°, 45°, 60°)$. Every 5th frame is labeled with the approximated turntable angle. The 30° and 60° frames are used for training and the 45° frames are used for testing. The goal of the instance pose estimation task is to identify the turntable angles of frames taken at 45° elevation angle. Experiments on category pose estimation is not tested since the task is to identify the aspect of a specific object instance in this work. For this experiment, the depth images are preprocessed to fill in empty values with the values of the closest pixels. Point clouds are also generated for each data based on the processed depth images.

4.2.2.2 Settings

During testing, the frames in the training set are treated as aspects x and the test frame as observations z. The prerecorded frame in memory that has the closest

turntable angle to the current observation should also afford the most similar set of actions. Hence, the angle of the frame in the training set that has the maximum posterior probability $p(x|z)$ given the test frame is chosen as the estimated angle. For each frame that is labeled with the turntable angle, the hierarchical CNN features is extracted and used to generate the appearance and pose descriptor. The number of extracted hierarchical CNN features is set to $N^5 = 30$ and $N^4 = 5$ in the conv-5 and conv-4 layer. conv-3 layer features is not included to reduce the test time. In this dataset, there are no force information from interacting with the object and there is no need to identify the object location with respect to the robot, therefore the force descriptor and location descriptor are not used in this experiment. The likelihood is modeled as $p(z|x) = p(r^z|r^x) \cdot p(q^z|q^x)$ instead. The force descriptor and location descriptor are tested in experiments described in the next chapter.

4.2.2.3 Results

Since the distribution of angle differences are skewed across objects, both the average error and the median error are used for evaluation. This framework that uses the proposed aspect representation is compared against three other reported approaches, (a) object pose tree with kernel descriptors [50], (b) hierarchical matching pursuit [7], and (c) pre-trained CNN with RGB and depth image [80]. This approach achieves a 38.1° average pose error and a 16.3° median pose error; both of these numbers achieve state of the art results as shown in Table 4.4. Note that while this experiment evaluates every test frame in the test set, other works only evaluate on frames that are correctly classified as the correct instance. Many of the errors are due to objects in the dataset that have similar appearance across multiple viewpoints. I argue that these objects often support the same set of actions when they are oriented at visually similar poses and only need to be represented by one aspect. For example,

the orientation of an orange is not important to the robot as long as the robot learned to manipulate it from one similar observation.

Table 4.4. Median and average instance pose estimation error on Washington RGB-D Objects dataset.

Work	Angular Error (°)	
	MedPose(I)	AvgPose(I)
OPTree, Lai *et al.* [50]	30.2	57.1
HMP, Bo *et al.* [7]	18.0	44.8
CNN: RGB-D, Schwarz *et al.* [80]	18.7	42.8
Hierarchical CNN feature (proposed approach)	**16.3**	**38.1**

4.3 Conclusions

This chapter presents an aspect representation that captures the essential affordances and supports manipulation in a hierarchical fashion. I first introduce the hierarchical CNN feature that captures the hierarchical relationship between filters in a convolutional neural network. These features are used to tackle the problem of preshaping a human-like robot hand for grasping based on visual input. The proposed approach first identifies hierarchical CNN features that are active consistently among the same type of grasps and localizes them by backpropagating the response along a single path to the point cloud. Robot controllers of different kinematic subchains are then associated with features in different convolutional neural network layers based on their corresponding hierarchy. The proposed approach is evaluated on the collected dataset and show significant improvement over approaches that do not associate filters in different layers in cluttered scenarios. This solution is further tested in a grasping experiment on Robonaut-2 where a total of 100 grasp trials on novel objects are performed and is shown to have a much higher success rate compared to a point cloud based approach. An aspect representation that supports manipulation and allows generalization to similar observations based on these hierarchical CNN features

is then introduced. This representation is evaluated on the instance pose estimation task in the Washington RGB-D Objects dataset and is shown to perform better then state of the art approaches.

CHAPTER 5

LEARNING FROM DEMONSTRATION

The aspect transition graph (ATG) memory model introduced in Chapter 3 is created through exploration in Section 3.2 and is manually designed in Section 3.5. These approaches for generating ATGs may work in lower dimensional spaces but do not scale well to higher dimensions. In this chapter, how ATG models combined with the hierarchical aspect representation introduced in Chapter 4 can be learned from demonstrations efficiently is introduced.

Learning from demonstration (LfD) is an attractive approach to programming robots because it resembles how humans transfer skills to each other. However, most work on LfD has focused on learning the demonstrated motion, action constraints, and/or trajectory segments with respect to the object pose. The assumption that an accurate object pose can be obtained may be accomplished in an industrial setting, but does not hold, in general, for the uncertainty and variability common in unstructured environments. This work deviates from this standard and defines actions based on features. An integrated approach that treats identifying informative features as part of the learning process is taken. This gives robots the capacity to manipulate objects without explicit pose information and to learn actions focused on salient parts of the object. With this approach, the robot can still interact with an object even if 1) the object does not have a global notion of pose, such as an articulated object, or 2) when the object's global pose is ambiguous but it's affordance can be identified.

In Section 5.1, a method for classifying demonstrations into three different types based on the interaction between visual features and robot end effectors is introduced.

In Section 5.2, how this categorization can help build ATG models from demonstrations is described. Section 5.3 explains how multiple demonstrations can be distilled to create more robust ATGs. Experiments performed on a ratchet task is then described in Section 5.4. With a small amount of demonstrations, the robot can perform tasks that require high accuracy. In Section 5.5, a drill grasping task that requires extending the robot's reach with both arms is demonstrated.

5.1 Demonstration Types

An action in an ATG is represented using a controller in the control basis framework [36] and is written in the form $\phi|_{\tau}^{\sigma}$, where ϕ is a potential function that describes the error between the current and target robot configuration, σ represents sensory resources allocated, and τ represents the motor resources allocated. The potential functions are formulated as $\phi_V = \sum_{v \in V}(v - g_v)^2$, where v and g_v are visual features and goal locations for these features ($v, g_v \in \mathbb{R}^3$) and $\phi_R = \sum_{r \in R}(r - g_r)^2$, where r and g_r are robot frames and goals for these frames ($r, g_r \in SE(3)$).

Demonstrated actions are classified into three types:

1. *robot-visual actions* $a_{RV} = \phi_{\mathrm{R}}|_{\tau}^{\sigma_{\mathrm{V}}}$
2. *robot-proprioceptive actions* $a_{RP} = \phi_{\mathrm{R}}|_{\tau}^{\sigma_{\mathrm{P}}}$
3. *visual-visual actions* $a_{VV} = \phi_{\mathrm{V}}|_{\tau}^{\sigma_{\mathrm{V}'}}$

Parameters σ_V and σ_P are the sensory resources containing a set of observed visual features V and a set of robot frames P based on proprioceptive feedback, respectively. Potential functions ϕ_R and ϕ_V have a minimum when a set of robot frames R and a set of visual features V that are controllable by the robot matches a set of corresponding goals G calculated based on offsets to σ. In this work, hierarchical CNN features introduced in Section 4.1 are used as visual features. Examples of these three types of demonstrations are in the following.

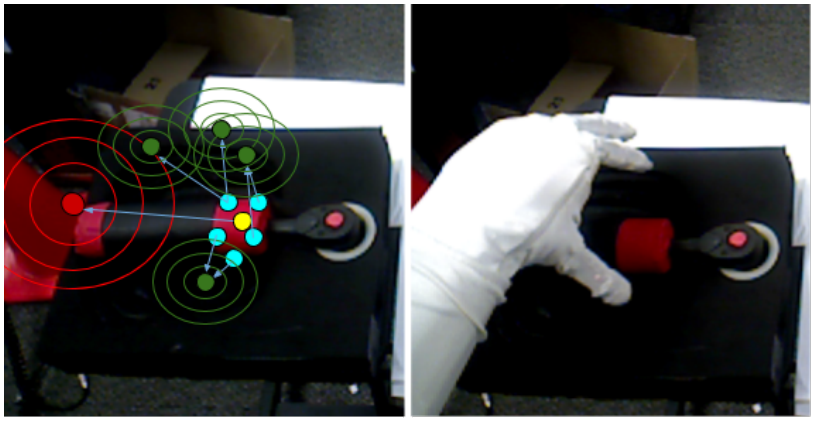

Figure 5.1. Example of a *robot-visual action* (a_{RV}) that reaches the ratchet pre-grasp pose.

a) The *robot-visual action* (a_{RV}) specifies the target pose of a set of robot frames with respect to a set of visual feature locations in 3-D. The left and right image in Figure 5.1 shows the result of executing an a_{RV} action where the goal is to reach the ratchet pre-grasp pose. The yellow and cyan dots are visual features represented by the 5th and 4th layer hierarchical CNN features and the red and green circles represent the minima of potential functions for the hand and fingers. The arrows represent the offset from features used as references to construct potential functions and the red and green ellipses represent the contour lines of the potential functions for the hand and index finger.

Figure 5.2. Example of a *robot-proprioceptive action* (a_{RP}) that extracts the ratchet.

b) The *robot-proprioceptive action* (a_{RP}) specifies the target pose of a set of robot frames with respect to a set of current robot frames based on proprioceptive feedback. The left and right image in Figure 5.2 shows the result of executing an a_{RP} action where the goal is to move the hand relative to the current hand frame so that the grasped ratchet is extracted from the tool holder. The yellow ellipse is the current hand pose and the arrow indicates the reference offset derived from demonstration. The red ellipses represent the contour lines of the potential functions for the hand.

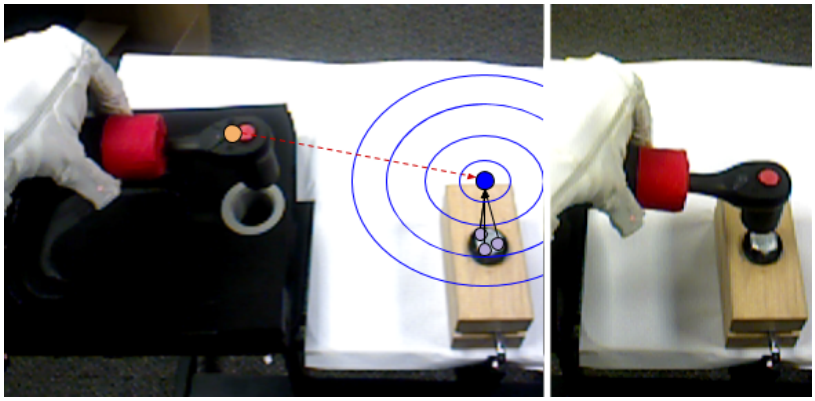

Figure 5.3. Example of a *visual-visual action* (a_{VV}) that places the ratchet on top of the bolt.

c) The *visual-visual action* (a_{VV}) specifies the goal position of a set of controllable visual features relative to another set of visual features on a different object in 3-D. The left and right image in Figure 5.3 shows the result of executing an a_{VV} action where the goal is to place the socket on top of the bolt. The purple dots are features on the bolt used as references for constructing the potential function and the orange dot is the feature on the socket controlled by the potential function. The blue dots are goal positions generated based on relative positions to features indicated by the black arrows. The red dotted arrow shows a path for the feature to reach the minimum of the potential function represented by the blue ellipse contours.

After a single visual-visual action, visual features on the grasped object may fail to reach the goal location due to movement error, change in object in-hand pose, or

imperfect camera calibration. To tackle this problem, the same action is executed multiple times with updated visual feature locations on the grasped object until convergence. Unlike *robot-visual actions*, modeling spatial relations between visual features achieves the same intended outcome even when the in-hand ratchet poses are different.

The detected locations of visual features and robot frames are inevitably influenced by noise in the system that may be caused by imperfect sensors or changes in the environment. This makes tasks that require high precision challenging. To accommodate this problem, the references for motor resources τ is assumed to be generated by adding zero mean noise $N(0, \Sigma)$ to the original reference. By sampling from this distribution during execution, the controller superimposes an additive zero mean search to the motion. Such structural search movement increases the tolerance to uncertainty of tasks such as insertion.

5.2 Building Models From Demonstrations

The hierarchical aspect representation introduced in Chapter 4 can be combined with ATG models based on the demonstration types introduced in the previous section. Action edges in an ATG can be represented by one of the three demonstration types based on interactions with hierarchical CNN features and proprioceptive feedback in the hierarchical aspect representation stored in the aspect nodes. Figure 5.4 shows this sensorimotor architecture that drives transitions in the ATG model. The perceptual feedback is used to represent aspect nodes and actions are executed based on these sensory resources defined in action edges.

Each demonstration coupled with information provided by the operator is used to create an ATG model. During execution, this set of ATG models is used to determine the next action based on the current observation and the given target. This section

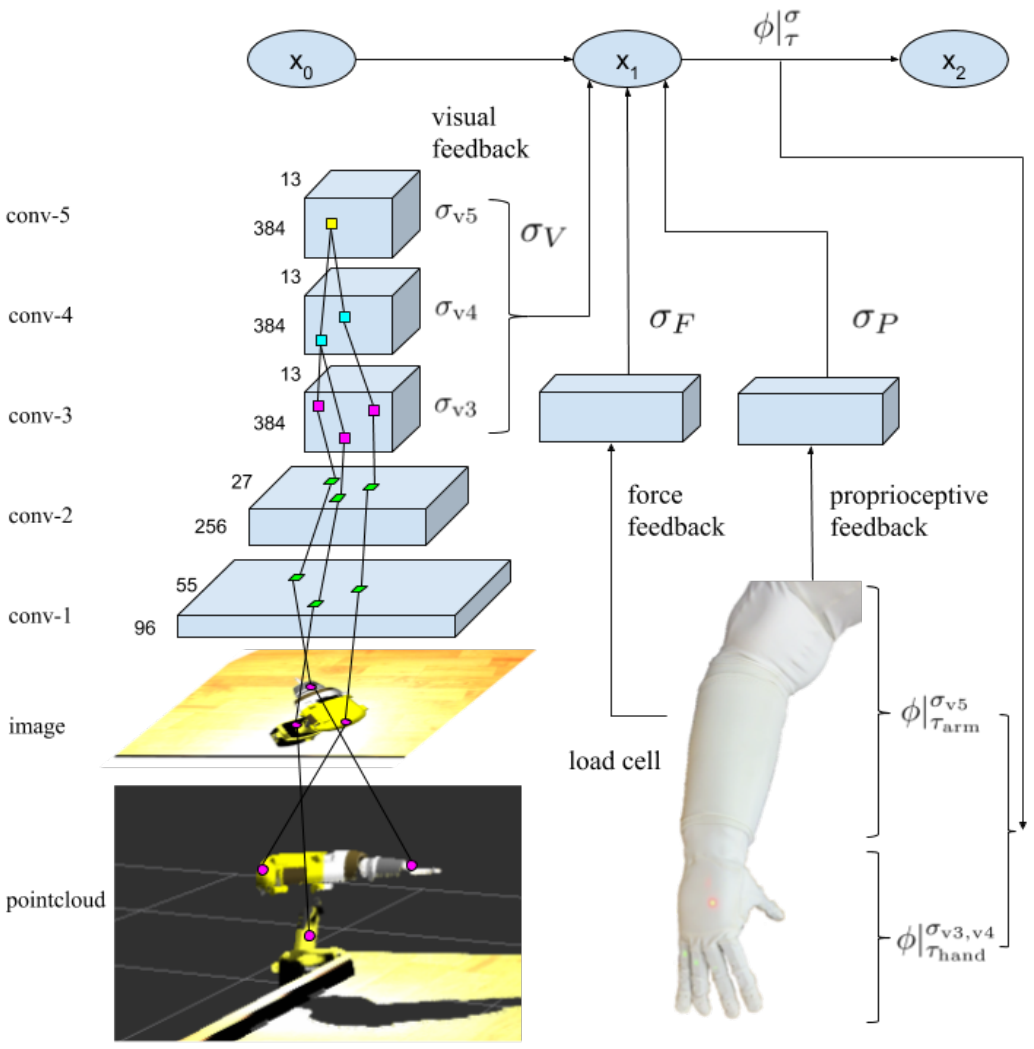

Figure 5.4. The sensorimotor architecture driving transitions in the ATG framework. The sensory resources σ_{F} that represent a set of features based on visual and force feedback and σ_{P} that represents a set of robot frames based on proprioceptive feedback are used to parameterize actions $\phi|_{\tau}^{\sigma}$. Here ϕ is a potential function that describes the error between the current and target robot configuration and τ represents the motor resources allocated. In this example, the 5th layer hierarchical CNN features σ_{v5} are used to control the arm motors τ_{arm} and the 3rd and 4th layer hierarchical CNN features $\sigma_{\mathrm{v3,v4}}$ are used to control the hand motors τ_{hand}.

describes the user interface for learning from demonstration and how ATG models based on hierarchical aspect representation can be learned.

5.2.1 User Interface

Demonstrated tasks are performed using a teleoperator implemented in the MoveIt! platform [88], in which the user can drag interactive markers in a graphical interface to move the robot end effector or change the robot hand configuration. Similar to the keyframe demonstration approach [1] [66], users indicate intermediate steps for each demonstration required for creating the ATG model. After the robot reaches an intermediate step, the interface asks the user to provide the type of demonstration listed in Section 5.1 that the user performed.

For the *robot-proprioceptive action* $a_{RP} = \phi_{\mathrm{R}}|_{\tau}^{\sigma_{\mathrm{P}}}$, the user can select either the end effector frame or the body frame as the proprioceptive sensor resource σ_P. The user also has the option to add a structural search movement described in Section 5.1 to the demonstrated action.

5.2.2 Creating ATG models

During the demonstration, an aspect node is created for each observed feature cluster at each intermediate step. A feature cluster can be a single object or multiple objects in contact based on the Euclidean cluster extraction algorithm in the point cloud library [77]. Based on the demonstration type selected by the user, the system connects new aspect node x_t to aspect node x_{t-1} created at the previous time step with action edge a_{t-1} that models the demonstrated action.

For the *robot-visual action* $a_{RV} = \phi_{\mathrm{R}}|_{\tau}^{\sigma_{\mathrm{V}}}$, the relative poses between a set of visual features V and a set of robot frames R are recorded in the action edge. This set of visual features is selected based on the feature's proximity to the robot end effector after action execution. In this work, hierarchical CNN features mentioned in Section 4.1 are used as visual features. Five 5th layer hierarchical CNN features that are closest to each hand frame and eight 4th layer hierarchical CNN features that are closest to each finger tip frame are selected. For example, the action that moves

the robot hand to a pre-grasp pose for grasping the ratchet will use features such as the corner of the handle or the neck of the ratchet that are close to the fingers as references for placing the fingers relative to the ratchet. The aspect nodes connected by this action is identified based on their proximity to the active robot end effector.

For the *robot-proprioceptive action* $a_{RP} = \phi_{\mathrm{R}}|_{\tau}^{\sigma_{\mathrm{P}}}$, the relative poses between the set of robot frames R and the set of reference robot frame P are recorded in the action edge. For example, the action that lifts the ratchet up after grasping it is modeled by moving the hand frame relative to the current hand frame. The aspect nodes connected by this action is identified based on their visual similarity to the last connected aspect node.

For the *visual-visual action* $a_{VV} = \phi_{\mathrm{V}}|_{\tau}^{\sigma_{\mathrm{V}'}}$, the relative poses between a set of visual features V on the tool grasped by the robot and a set of visual features V' on the target object interacting with the tool is recorded in the action edge. The set of visual features on the tool is selected based on the feature's stability with respect to movement under the assumption that the grasped object is rigid. This is determined by the position differences of the features in the hand frame before and after the action. The set of visual features on the target object is then selected based on the feature's distance to the selected features on the tool after the action. This *visual-visual action* is represented by two action edges that connect the aspect nodes that represent the tool and the target object to the aspect node that represents the interaction. These aspect nodes can be identified base on their relative distance and proximity to the active end effector.

This sequence of aspect nodes connected by action edges become the ATG model that represents the demonstration. Aspect nodes that are created from other feature clusters that are not chosen are grouped into a background ATG that is used to recognize feature clusters that are not targets for manipulation. For example, a demonstration that only grasps the ratchet does not care about the bolt platform. A

background ATG that represents the bolt is therefore used to match to the feature cluster of the bolt during execution.

5.3 Distilling Multiple Demonstrations

With a single demonstration, there remain ambiguities regarding the goal. For example, in the action that puts the socket on top of the bolt, it is ambiguous whether the demonstration intends to convey a spatial relationship between the socket and the bolt or some other part of the ratchet and the bolt. With multiple demonstrations, this ambiguity may be resolved by observing consistent relations between features. In this section, how to take multiple demonstrations of the same task and create more robust ATG models is described. These ATGs created from multiple demonstrations are called *distilled* ATGs.

5.3.1 Identifying Common Features

A set of features are stored in the aspect node to represent the observation of an aspect. Correctly associating the current observations with a memorized aspect node is crucial for implementing transitions to goal status. However, not all features provide the same amount of information. Moreover, some features are more sensitive to lighting changes and some may belong to parts of the visual cluster that may change appearance across examples. With a single demonstration, these kinds of features may be indistinguishable. With multiple demonstrations, common features can be identified by estimating the feature variance across demonstrations.

Given demonstrations of the same task with the same sequence of intermediate steps, the proposed approach looks for features that are consistent across multiple demonstrations. For the observations at each intermediate step, the N most consistent features are chosen. The consistency score is defined as $S_c = n_f / std(f)$, where n_f is the number of times feature f appears among the matched intermediate steps

and $std(f)$ is the standard deviation of the value of feature f. Visual features, proprioceptive features, and force features are scored together with weights of $1, 1, 0.001$, respectively.

5.3.2 Recognizing Consistent Actions

For action edges that represent a *robot-visual action* a_{RV} or a *visual-visual action* a_{VV} in an ATG model, the action reference is specified in terms of a subset of features stored in the aspect node. As result of a single demonstration, features are chosen based on their proximity to robot frames or features controllable by the robot. With multiple demonstrations, a more robust set of features can be identified and used to define the aspect.

For the *robot-visual action* $a_{RV} = \phi_{\mathrm{R}}|_{\tau}^{\sigma_{\mathrm{V}}}$, the top N pairs of robot frames $r \in R$ and visual features $v \in V$ that have the lowest variances in XYZ position offsets are chosen to represent the action. For example, when learning from multiple demonstrations of the action that grasps the ratchet, this approach concludes that features on the ratchet are more reliable than features on the tool holder since the ratchet may be placed at different positions in the tool holder across demonstrations.

For the *visual-visual action* $a_{VV} = \phi_{\mathrm{V}}|_{\tau}^{\sigma_{\mathrm{V}'}}$, the top N pairs of visual features in the tool aspect node $v \in V$ and the target object aspect node $v' \in V'$ that have the lowest variance $var(v, v')$ is selected. $var(v, v')$ is the variance of the XYZ position offsets between feature v and feature v' after the action across demonstrations. For example, the action that places the socket of the ratchet on top of the bolt determines that a consistent spatial relation exists between the features on the socket and those on the bolt after executing the action. Figure 5.5 shows the top feature pairs identified for constructing a *visual-visual action* from demonstrations. The robot is able to comprehend that the head of the ratchet should be aligned with the bolt autonomously.

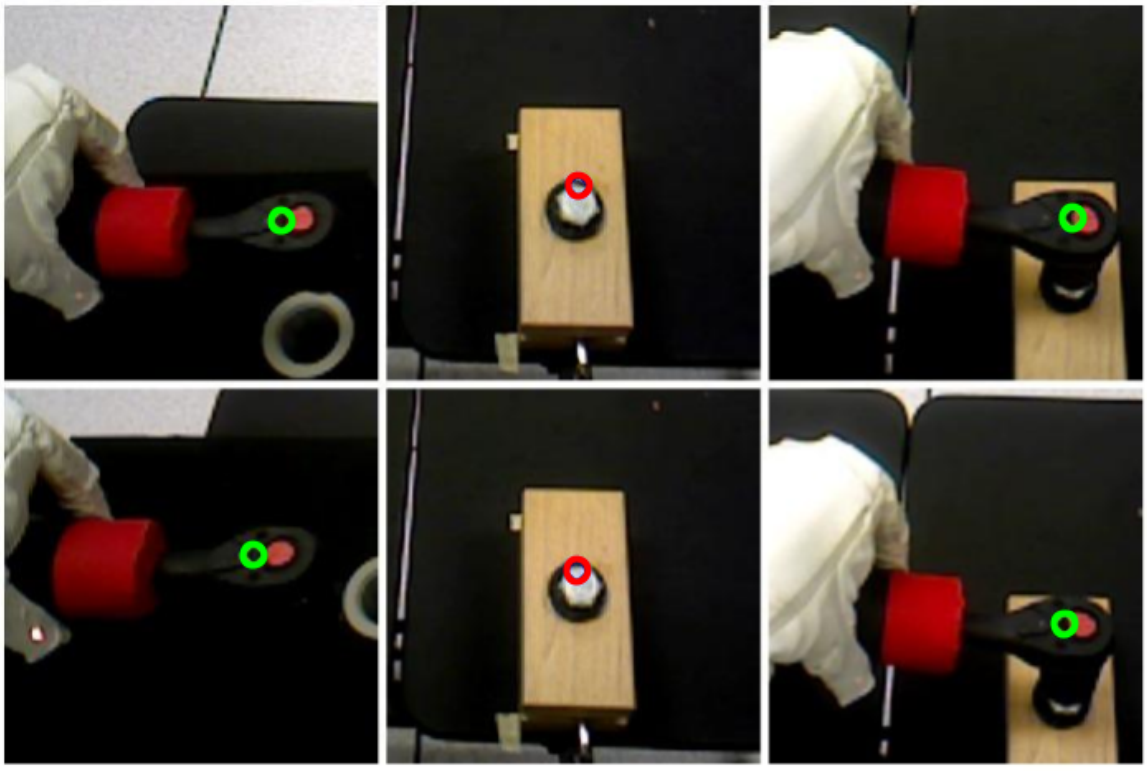

Figure 5.5. Identifying informative features from multiple demonstrations. The two rows represent two demonstrations that place the socket of the ratchet on top of the bolt. The columns from left to right show the aspect nodes representing the tool, the target object, and the interaction for this *visual-visual action* $a_{VV} = \phi_{\mathrm{V}}|_{\tau}^{\sigma_{\mathrm{V}'}}$. The green circles in the tool and interaction aspect nodes represent the top visual feature $v \in V$ used to reach the minimum of the potential function ϕ_V while the red circles in the target object aspect node represent corresponding features $v' \in V'$ that are used as references.

To confirm that the selected visual features represent meaningful parts of an object, the feature identified on the ratchet head is visualized in Figure 5.5 using the visualization tool introduced by Yosinski et al. [101]. Figure 5.6 shows the top 9 images that the filters f_{23}^5, f_{60}^4, and f_{184}^3 have the highest response on among the ImageNet dataset [76]. The feature tuple $(f_{23}^5, f_{60}^4, f_{184}^3)$ can be interpreted as a red region surrounded by black regions. Although there are no ratchet class trained on the neural network, it reuses similar visual patterns learned among other classes such as bird species. The left image in Figure 5.7 shows a visualization on what pixels

contribute to the feature using guided backpropagation [86]. The background grey corresponds to zero contribution and the red dot represents the mean location of all the responses weighted by its contribution. Notice that the feature is only contributed by the head of the ratchet and represents meaningful parts for modeling the action. The right image is the corresponding input image.

Figure 5.6. Visualization of the hierarchical CNN feature $(f_{23}^5, f_{60}^4, f_{184}^3)$ that is identified on the ratchet head by showing the top 9 images that have the highest response among ImageNet for filter f_{23}^5, f_{60}^4, and f_{184}^3.

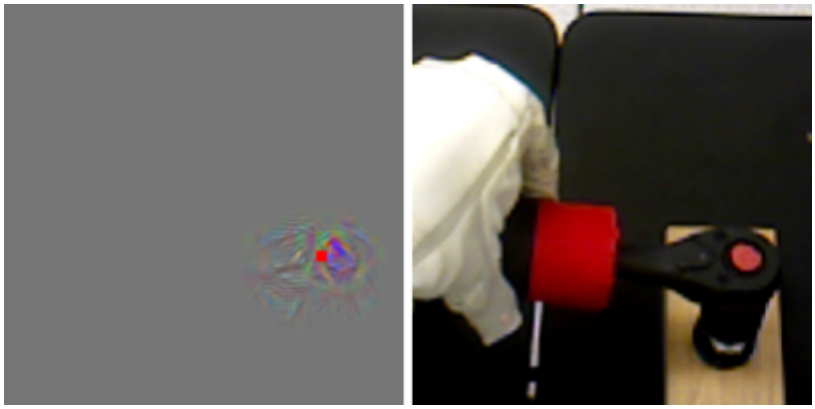

Figure 5.7. Visualization on what pixels contribute to the hierarchical CNN feature $(f_{23}^5, f_{60}^4, f_{184}^3)$ using guided backpropagation.

5.4 Experiments on the Ratchet Task

This section shows that with a small set of demonstrations, Robonaut-2 is capable of performing a ratchet task that involves grasping the ratchet, tightening a bolt, and putting the ratchet back into a tool holder. The success rate of mating the socket to the bolt as a function of the number of demonstrations is analyzed. To evaluate the accuracy of the position of the socket with respect to the bolt, experiments in the Robonaut-2 simulator [15] using up to five demonstrations are conducted. The success rate of mating the socket to the bolt as a function of the number of demonstrations and the size of the feature space is further compared on the real robot. The demonstration collection process, the planner, the experimental setting, and the result of the comparison is described in the following.

5.4.1 Demonstrations

Instead of demonstrating the entire ratchet task in one session, the task is segmented into shorter sequences of sub-tasks that are easier to demonstrate. The ratchet task is segmented into five different subtasks, *a)* grasping the ratchet, *b)* mating socket to the bolt, *c)* tightening the bolt, *d)* removing the socket from the bolt, and *e)* putting the ratchet back into the tool holder. For subtasks *a)*, two demonstrations are provided. For subtask *b)* and *e)* four demonstrations are combined to create the distilled ATG model as described in Section 5.3. For subtasks *c)* and *d)*, only one demonstration is performed since the features that support these actions are unambiguous.

Figure 5.8 shows the ATGs created from these five sub-tasks from top to bottom. The type of demonstrations classified for each action are listed next to the action edges. For example, the ATG created for subtask *a)* (grasping the ratchet) has four different relations between the hand, the ratchet, and the tool holder: the ratchet in the tool holder (no hand), pre-grasp, grasped within and without the tool holder.

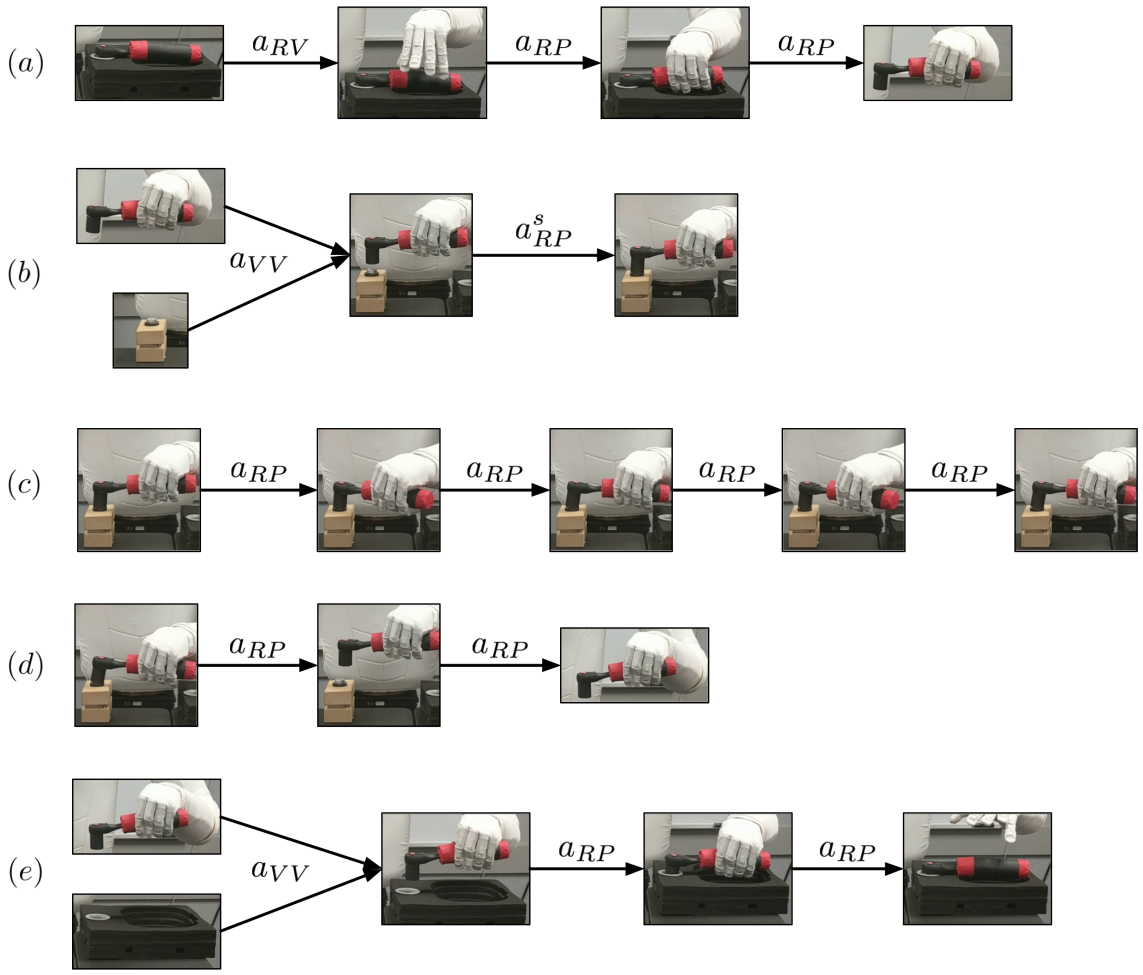

Figure 5.8. The visualization of the set of ATGs created from demonstrations for the ratchet task. Each connected ATG represents a sub-task. The images represents aspect nodes and the edges indicate the type of actions used to model transitions.

ATG for *b)* shows that in order to execute the *visual-visual action*, both the ratchet-in-hand aspect and the bolt aspect have to exist. The second action edge that mates the socket to the bolt incorporates a structural search motion as well. ATG for subtask *c)* is created from demonstrations of two tightening turns. Each clockwise and counter-clockwise turn is categorized as a type a_2 demonstration that moves relative to the hand frame.

5.4.2 Planner

The set of ATG models created from demonstrations stores observations of each feature cluster in aspect nodes and predicts transitions caused by action edges (Figure 5.8). During execution, this set of ATG models is used to plan actions to reach a given goal state.

At time step t, the aspect node x_t in the set of ATGs that has the highest posterior probability $p(x_t|z_t)$ given the current observation z_t of the feature cluster is first identified for each feature cluster. The prior probability $p(x_0)$ of being in an aspect node x_0 at time 0 is set to be uniform over all aspect nodes in the set of ATGs unless modified by the user. The probability $p(x_t)$ of being in an aspect node x_t is updated by the Bayes filter algorithm, $p(x_t) = \sum_{x_{t-1}} p(x_t|a_{t-1}, x_{t-1}) \cdot p(x_{t-1})$, where a_{t-1} is the action taken at time step $t-1$. The transition probability $p(x_t|a_{t-1}, x_{t-1})$ is set proportional to $p(a_{t-1}|x_t, x_{t-1})$, which is modeled as a multivariate Gaussian distribution based on the value difference between the parameters of the executed action a_{t-1} and the action edge $\hat{a}$ that connects aspects x_{t-1} and x_t in the ATG model. The maximum a posteriori (MAP) aspect node for each feature cluster can therefore be determined by calculating $p(x_t|z_t) \propto p(z_t|x_t) \cdot p(x_t)$, where $p(z_t|x_t)$ is modeled with generalized Gaussian distributions as introduced in Section 4.2.

During execution, the user selects a goal aspect. Based on the MAP aspect node x_t of each feature cluster, the next action is chosen based on the first action edge on the shortest path from the MAP aspect node to the goal aspect node. If the chosen action edge is a *visual-visual action* type, the planner needs to confirm that both the tool aspect node and the target object aspect node is observed. There are two ways to transition between nodes, 1) follow edges learned from demonstrations, or 2) identify equivalent aspect nodes in ATGs and transition between them. If there is no valid path from the current aspect node x_t to the given goal aspect node, the planner guesses possible paths by merging similar aspect nodes from the current ATG

to other ATGs until a path exists. The similarity between two aspect node uses the same model as the likelihood function $p(z_t|x_t)$. These two ways of identifying paths in ATGs allow the robot to learn subtasks separately and repeat the full task during execution.

In this experiment, the aspect where the bolt is tightened is first submitted as a goal aspect to the robot. The planner identifies the current aspect node and finds a path to reach the goal aspect. Once the robot finishes tightening the bolt, the aspect where the ratchet is put back to the tool holder is set as the goal aspect. Figure 5.9 shows the sequence of the complete task. With this approach Robonaut-2 is capable of executing the complete ratchet task successfully even when there are small differences in the initial tool, bolt, and tool holder locations.

5.4.3 Evaluating Ratchet Task

In this experiment, the robustness of the framework is tested on the ratchet task based on the ATGs created from demonstrations. A total of 22 settings are tested. For each setting, the initial location of the tool holder or bolt platform is altered. For the first 16 settings, the bolt platform is moved away 5 cm from the demonstrated position and the tool holder is placed at 16 different locations on a four by four grid that are 1 cm apart. For the other 6 settings, different bolt platform positions and orientations and one randomly chosen tool holder position on the four by four grid are set. These initial poses are shown in Figure 5.10. For each different settings, if grasping fails, the robot retries grasping. If mating the socket with the bolt fails, the robot skips tightening and continue. The number of successes for each subtask are shown in table 5.1. Grasping failed twice when the ratchet got stuck and the robot lifted the whole tool holder. Mating socket with the bolt and placing the ratchet back have 86.3% and 81.8% success rate. Tightening failed once when the socket slipped away from the bolt while tightening. 14 out of 24 trials succeeded the complete task.

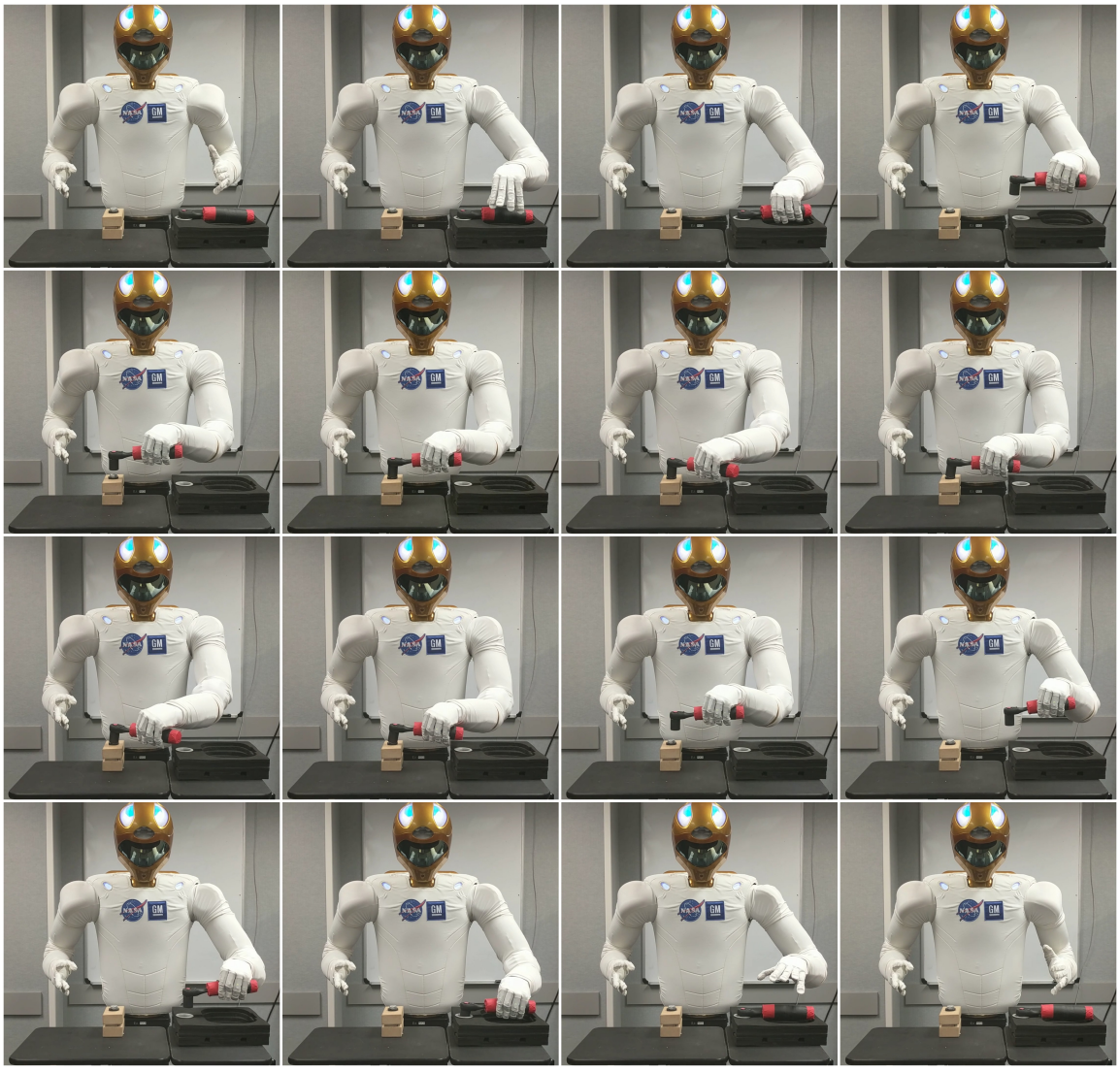

Figure 5.9. The ratchet task sequence performed by Robonaut-2. The images from left to right, then top to bottom, show a sequence of actions where Robonaut-2 grasps the ratchet, tightens a bolt on a platform, and puts the ratchet back into a tool holder.

Table 5.1. Number of successful trials on subtasks.

subtask	(a)	(b)	(c)	(d)	(e)
successful trials / total trials	22 / 24	19 / 22	18 / 19	22 / 22	18 / 22

13 corner case settings are further tested on mating the socket with the bolt. This set contains test cases with initial ratchet positions that are close to the sensor

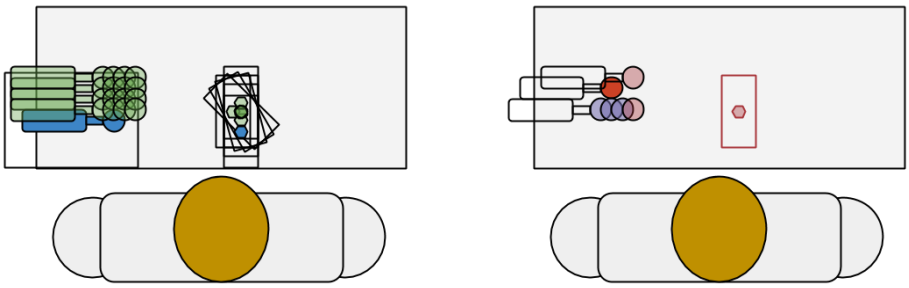

Figure 5.10. Top down views of initial poses and failed poses on the ratchet task. The green objects in the left image shows a set of initial poses tested and the blue objects are the initial poses for the demonstrations. The pink objects in the right image shows a set of initial poses that failed to mate the socket with the bolt and the purple objects are the initial poses that failed to place the ratchet back. The red ratchet pose failed in both subtasks in two different trials.

and joint limit, in hand ratchet positions that are at opposite ends, and cluttered scenarios. This approach achieved a similar success rate of 84.6%. Figure 5.11 shows some of the initial settings.

Figure 5.11. Corner case initial settings for mating the socket with the bolt. Note that in the 3rd and 4th image the in-hand ratchet positions are different.

5.4.4 Comparing Accuracy in Simulation

To understand how the number of demonstrations used affect the action accuracy, ATGs created with one to five demonstrations in the Robonaut-2 simulator are compared. For each of these five ATGs, 125 trials are tested on the placing socket on top of the bolt task with different in-hand ratchet poses and bolt platform locations. For each trial, a perturbation $P = (r_{xy}, r_\theta, b_{xy})$ is added to an initial configuration, where r_{xy} is the ratchet offset in the XY plane in hand, r_θ is the ratchet

angle difference on the Z axis in hand, and b_{xy} is the bolt platform offset in the XY plane. All combinations of the following set of parameters are tested, $r_{xy} = \{(0,0),(2,0),(0,2),(-2,0),(0,-2)\}$ in centimeters, $r_\theta = \{-0.2,-0.1,0,0.1,0.2\}$ in radians, and $b_{xy} = \{(0,0),(3,0),(0,3),(-3,0),(0,-3)\}$ in centimeters. For each trial, the distance between the final socket location and the ground truth socket location calculated based on the demonstration is recorded.

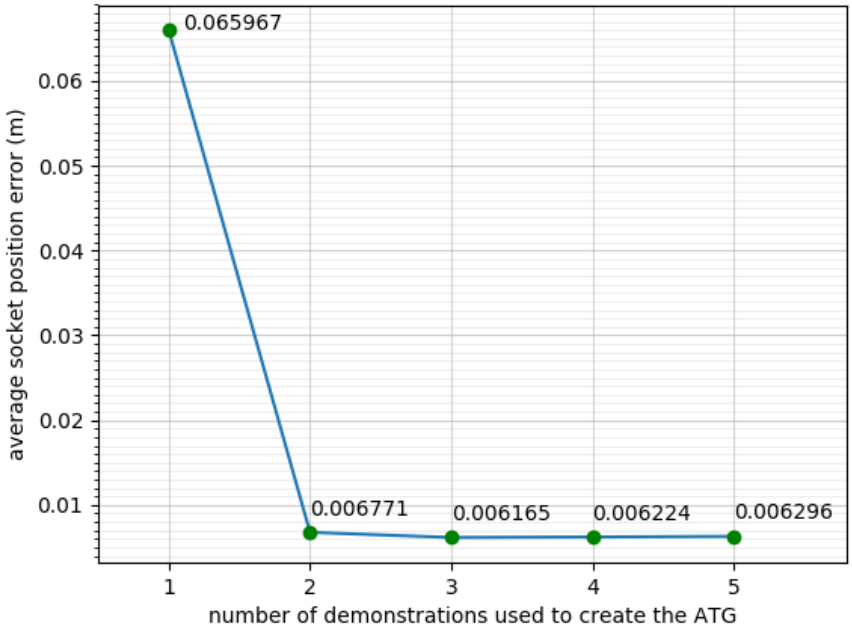

Figure 5.12. The accuracy of the placing socket on top of the bolt task versus the number of demonstrations used to create the ATG.

The results are shown in Figure 5.12. With two demonstrations a distilled ATG can lower the socket position error significantly. Adding more demonstrations did not improve the accuracy much on this task. This may be because that with two demonstrations the visual features identified are already the best among the detected set. Figure 5.13 shows the informative features, represented by green dots, identified on the ratchet for ATGs created with one, two, and five demonstrations. The feature selected in the ATG created from a single demonstration is further away from the socket than the features selected by ATGs created from multiple demonstrations.

This offset may result in less accurate actions when the ratchet is held in the hand with a different angle. In this case, the features identified among ATGs created from two to five demonstrations are similar, and therefore result in similar accuracy. This result is consistent to findings in the Robonaut-2 experiment. With a small set of demonstrations, the distilled ATG identifies more informative features and lowers the action error significantly.

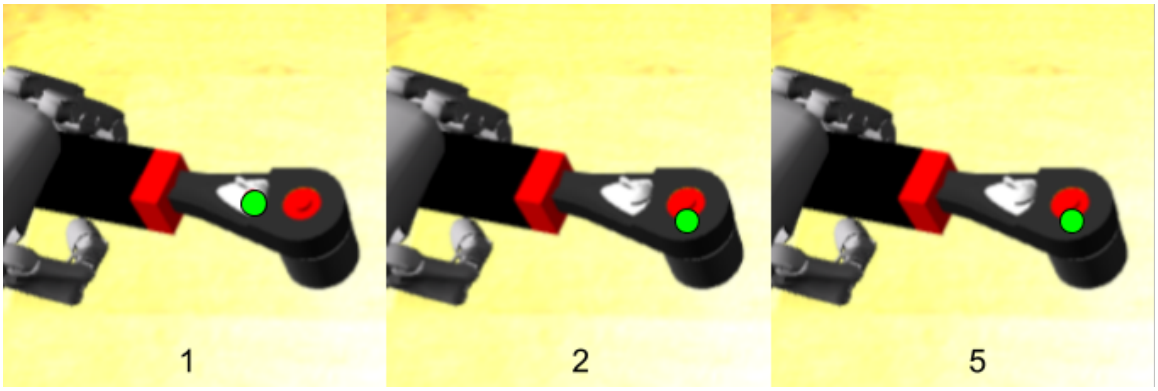

Figure 5.13. Informative features identified in experiments in simulation. The images from left to right corresponds to tool aspect nodes for the putting socket on top of the bolt task using ATGs created from one, two, and five demonstrations. The green dots represent the visual features selected to represent the action. The feature selected in the ATG created from a single demonstration is further away from the socket and may result in less accurate actions.

5.4.5 Effects of Multiple Demonstrations and Feature Complexity

To further evaluate how the number of demonstrations and the size of the visual feature space affect the learned action on the real robot, the success rates of mating the socket to the bolt under different configurations are compared. In this experiment, the robustness of ATGs created from one to four demonstrations and with hierarchical CNN features in the 3rd and 4th layer is compared. Hierarchical CNN features in the 3rd layer $H^3 = (f_i^5, f_j^4, f_k^3)$, represents a feature with an additional filter f_k^3 and have a feature space $|f^3| = 384$ times larger compared to features in the 4th layer $H^4 = (f_i^5, f_j^4)$, where $|f^3|$ is the number of filters in the conv-3 layer. The assumption

is that more complex features will require more demonstrations to learn, but may result in more robust actions. For each trial, the robot starts with the grasped ratchet and the bolt placed on the right side of the robot. The trial succeeds if the robot mates the socket to the bolt. 22 trials are performed for each ATG. The results are shown in Figure 5.14.

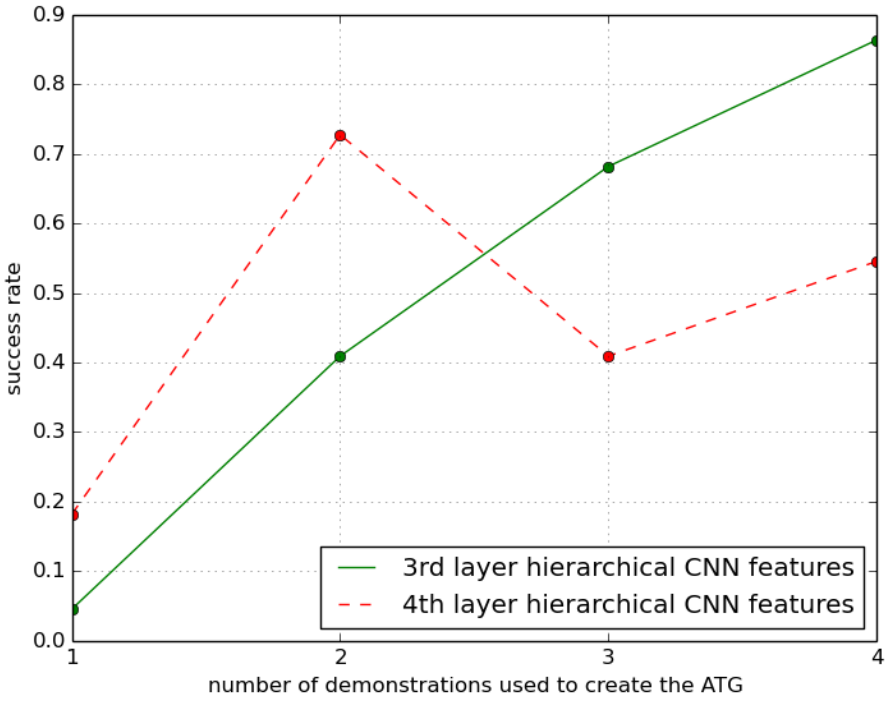

Figure 5.14. Success rate versus number of demonstrations and size of feature space

Consistent with the assumption, the success rate of using H^3 features increases with more demonstrations and performs better than H^4 features when more demonstrations are used. The results for using H^4 features however fluctuates with more than two demonstrations. I suspect that this is because H^4 features have a smaller feature space and good features can be found with fewer demonstrations. The up and down in success rate with more demonstrations may be due to imperfect demonstrations and H^4 features that are less precise in location.

5.5 Experiments on Drill Grasping

In this section, experiments on a drill grasping task on Robonaut-2 using the proposed frame work is described. It is demonstrated that with the hierarchical aspect representation, the current observation can be associated with the aspect that supports the same set of actions. Through a few demonstrated manipulations on a drill, the robot is able to grasp the drill in a position that is normally out of reach by combining learned actions in sequence.

5.5.1 Settings

The goal of the task is to grasp the drill handle correctly with the left robot hand based on 8 demonstrated manipulation action sequences on a drill on Robonaut-2 collected through teleoperation. Three of the demonstrated action sequences are grasp action sequences, where the drill is placed at three different orientations and the user teleoperates the robot to hold the drill handle with its left hand. Four of the demonstrated actions are drag action sequences, where the drill is placed at the right side of the robot and the user teleoperates the right hand to drag the drill from the right side to the center. The other demonstrated action sequence is a turn action, where the drill is placed such that the tip of the drill is facing toward the right side of the robot and the user teleoperates the right hand to turn the drill such that the tip of the drill is facing away from the robot.

For each test trial, the drill is placed on the table in front of the robot. The robot can manipulate the drill until it successfully grasps it on the handle. For example, if the drill is placed on the right side of the table and not reachable with the left hand, the robot can drag the drill closer with its right hand and grasp it with its left hand. If the drill ends up in a pose that is no longer graspable or if the robot tries to grasp and fails, the trial is recorded as failed. The experiment is evaluated based on the

number of successful grasps where the robot fingers surround the handle such that the tip of the drill is facing outward from the wrist.

5.5.2 Demonstrations

An ATG is generated for each of the 8 demonstrations based on intermediate steps and demonstration types indicated by the user during teleoperation as described in Section 5.2; an aspect node that stores the proposed aspect representation is created between each action. For example, one demonstrated turn example is a three action sequence of moving the hand to a pre-turn pose, pushing the drill tip to turn the drill, and moving the hand back. These demonstrations are classified into robot-visual actions a_{RV} and robot-proprioceptive actions a_{RP} described in Section 5.1. In this experiment, the top three filters with the highest responses in the conv-5, conv-4, and conv-3 layers ($N^5 = N^4 = N^3 = 3$) is used for extracting the hierarchical CNN features. Each aspect representation is composed of an appearance descriptor with 39 hierarchical CNN feature response values, a pose descriptor with 741 XYZ differences between these 39 features, and a location descriptor with 4 distance values from the centroid of these features to the robot's palms and shoulders. An action edge that connects from aspect x_t to x_{t+1} in an ATG stores an action that is configured by the position offset between the robot end effectors and the set of corresponding hierarchical CNN features in aspect x_t. The top K features in x_t that the robot end effectors are closest to after executing the action is chosen. K is set to 5 in this test. In this experiment, the arm controllers are associated with the conv-4 layer hierarchical CNN features while the hand controllers are associated with the conv-3 layer hierarchical CNN features as described in Section 4.1.

The 8 ATGs created from demonstration are combined into one ATG that represents all the manipulations the robot memorized for different aspects of the drill. Both the drag and turn action sequences conclude in a state where the drill is on the

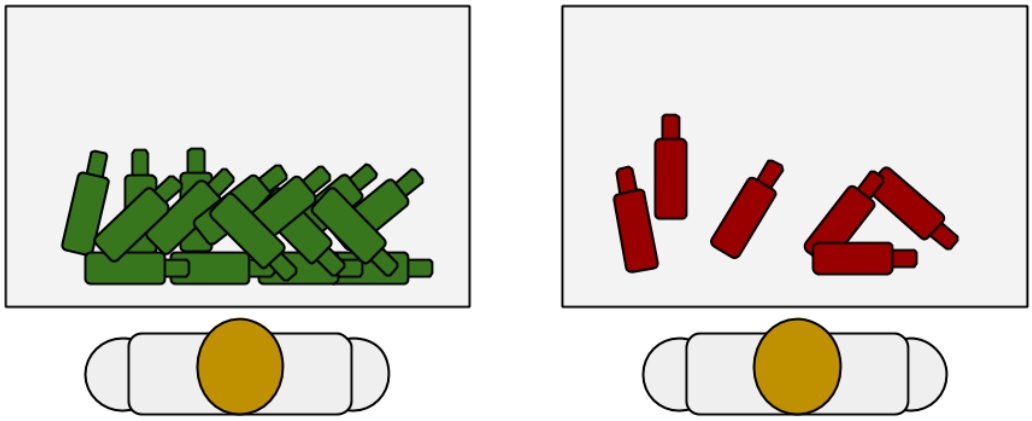

Figure 5.15. Initial drill poses that the robot succeeded and failed in grasping with its left hand during testing. The green drill poses in the left figure shows the succeeded poses and the red drill poses in the right figure shows the failed poses. This approach allows the robot to grasp drills located at position that is normally out of reach.

table with no contact with the robot; since the orientation and location of the drill is uncertain after these actions, the last aspect node of the ATGs corresponding to these demonstrations is connected to an intermediate node that is connected to the first aspect of all of the 8 ATGs created. The three grasp action sequences end up in a state where the drill is grasped correctly in the robot hand; since the task is to reach such state, the last aspect nodes of the ATGs corresponding to these demonstrations is connected to an aspect that indicates that the drill is grasped. The final ATG contains 31 aspect nodes and 32 action edges.

5.5.3 Approach

For each trial, the aspect node x in the combined ATG that has the highest posterior probability $p(x|z)$ given the current observation z is first identified. The prior probability $p(x)$ is set to be uniform among aspect nodes that are the first aspect of each demonstration. As in the ratchet experiment described in the previous section, the maximum a posteriori (MAP) aspect node $p(x|z)$ can be determined by calculating $p(z|x) \cdot p(x)$, where $p(z|x)$ is described in Section 4.2. The next action is

then chosen based on the first action edge on the shortest path from the MAP aspect node to the goal aspect node.

For each action, the target positions for the palms are determined by the mean position of a set of conv-4 layer features plus the corresponding offsets. The same approach is used to determine target positions for the fingers and thumbs from a set of conv-3 layer features plus their respective offsets.

Each action is executed in two steps. First, the arm controllers move the arms such that the distance from the palms to their corresponding targets are minimized. Once the arm controllers converge, the hand controllers move the wrists and fingers to minimize the sum of distance from the index finger tip, middle finger tip, and thumb tip to their corresponding target. The posterior probability $p(x|z)$ is recursively updated based on the Bayesian filtering algorithm [91] after each action and observation as described in Subsection 3.2.1. The next action is always chosen based on the MAP aspect node after the update.

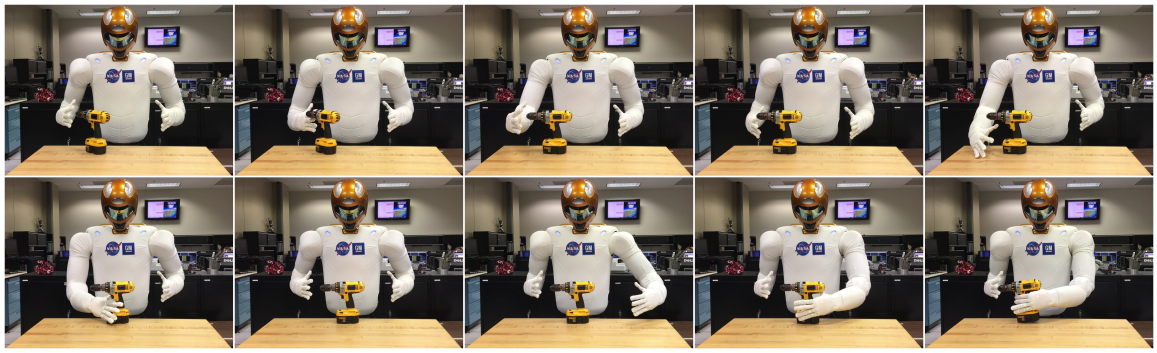

Figure 5.16. Sequence of actions in one grasping test trial. The images are ordered from left to right then top to bottom. The initial pose of the drill is at an angle that is not graspable and is located too far right for the left hand to reach. Therefore the robot turns the drill then drags it to the center before grasping with its left hand.

5.5.4 Results

22 grasping trials are performed in this experiment and the proposed approach successfully grasped the drill on the handle 16 times. Among 11 of the successful trials,

the robot turned or dragged the drill with its right hand before grasping it with its left hand. Figure 5.16 shows one of the trials that the robot executed both turning and dragging before grasping the drill. The initial poses of the drill that the robot succeeded or failed in grasping are shown in Figure 5.15. Three of the failed trials are due to the robot trying to grasp the drill while the drill is placed at a pose almost within reach. Adding more demonstrations or the ability to recover from error may improve the performance on this task. Calculating the aspect representation takes about 3 seconds on a desktop computer with the NVIDIA GTX 780 graphics card. Matching the current aspect to a memorized aspect has a complexity of $O(n)$, where n is the number of memorized aspects; this matching process takes less then a second in this experiment.

5.6 Conclusions

This chapter describes how ATG models combined with the proposed hierarchical aspect representation can be learned from demonstrations efficiently. I first introduce a categorization that classifies demonstrations into three different types depending on what frames or features are used as references and what is used to calculate the error to the target. Having the user provide additional information about the type of the demonstration allows the system to create action edges in ATGs by modeling the spatial relations between hierarchical CNN features automatically. It is then shown that through multiple demonstrations, informative visual features and relative poses may be identified and used to model actions that are more accurate than models of single demonstrations. This effect is clearly observed in the improvement in success rate and accuracy over single demonstration models on the task on mating the socket to the bolt on Robonaut-2. I show that with this proposed approach, Robonaut-2 is capable of grasping the ratchet, tightening a bolt, and putting the ratchet back into a tool holder with a small set of demonstrations. The capability of such framework

is further demonstrated in a drill grasping task that requires the robot to use both hands to extend its reach. The goal is to grasp the drill placed in multiple poses based on a small set of grasp, drag, and turn actions demonstrated to the robot. It is shown that by combining learned subtasks, the robot can extend the situations it is capable of solving beyond the ones it was demonstrated.

CHAPTER 6

CONCLUSIONS AND FUTURE DIRECTIONS

The goal of this dissertation is to present a framework that allows robots to solve tasks in an unstructured environment through predicting perceptual action consequences based on memory and observation. I propose a framework where hierarchical aspect representations are used to construct aspect transition graph (ATG) models that memorize how actions change observations. I then show that this integrated memory model of perception and action can be learned efficiently from demonstrations and is capable of solving tasks beyond the given examples.

Chapter 3 introduces the aspect transition graph memory model that memorizes action consequences through a directed multigraph composed of aspect nodes and action edges. By predicting action outcomes with this memory model, I show that the robot can perform actions that help distinguish objects, reach a goal reliably with a sequence of open-loop and closed-loop actions, grasp a drill without explicit pose estimation, and detect errors early.

Chapter 4 presents the hierarchical CNN feature that is capable of representing local parts that belong to a high level structure. These features can be localized in 3D and are associated with controllers that belong to different kinematic subchains to support grasping. Based on these hierarchical CNN features and other sensory feedback, a hierarchical aspect representation that captures object affordance is introduced. This approach achieves state of the art result on instance pose recognition and outperforms a baseline approach on grasping novel objects.

Chapter 5 describes how ATG models combined with the proposed hierarchical aspect representation can be learned from demonstrations efficiently. Demonstrations are classified into three different types based on the reference frames. Through multiple demonstrations, informative visual features and consistent spatial relationships can be identified and used to model actions with higher accuracy. I show that this approach is capable of accomplishing a challenging bolt tightening task where the in hand ratchet poses vary. This approach is also tested on drill grasping task where the robot has to extend its reach by combining a sequence of learned actions.

6.1 Conclusions and Discussions

In the introduction, I presented a modified conceptual diagram of the neocortex (Figure 1.1) taken from the book "On Intelligence" [33]. In this figure, memory regions that connect sensory neurons and motor neurons of the same layer is added to the original diagram. The modules and connections of this diagram that are implemented in this dissertation are highlighted in Figure 6.1 and discussed in the following.

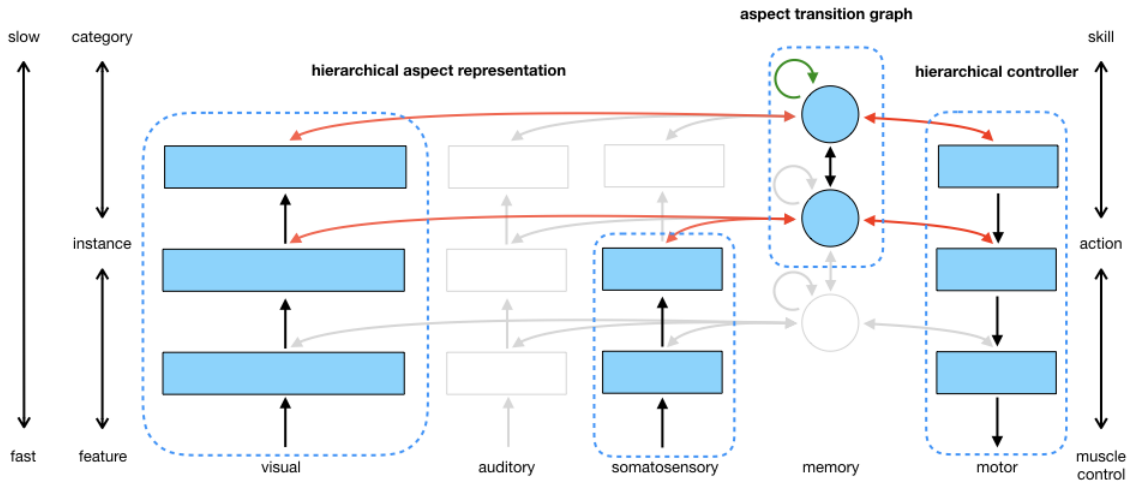

Figure 6.1. The proposed conceptual diagram of the neocortex with modules and connections implemented in this dissertation highlighted. The colored blocks and connections are the parts that are tested in robotics experiments.

- The memory regions in the diagram correspond to the memory model introduced in Chapter 3. The connection loops within the memory regions resemble the action edges that can be used to predict action consequences in an ATG.

- The visual neuron layers in the diagram are represented by hierarchical CNN features introduced in Section 4.1 and are combined with the somatosensory neurons to form the hierarchical aspect representation described in Chapter 4.

- The highlighted layer-wise connections between visual neuron layers and motor neuron layers correspond to the hierarchical controller introduced in the Robonaut-2 grasping experiment in Subsection 4.1.4.

- The connections between the aspect transition graph, hierarchical aspect representation, and the hierarchical controller can be learned through demonstration as introduced in Chapter 5.

Throughout the robotic experiments conducted in this dissertation, I show the advantages of the proposed memory model and the merit of having hierarchical associations between controllers, perceptual features, and memory models. This framework is tested on multiple robotic tasks including 1) recognizing and modeling partially observed objects through interaction, 2) grasping tools with visual servoing, 3) handling error in a stochastic environment, 4) grasping novel objects, 5) tightening bolts with learning from demonstrations, and 6) grasping drills with both arms to extend reach. This framework has shown to be effective in many aspects and capable of accomplishing challenging tasks that few current approaches can achieve under similar conditions. These results can be seen as support to the conjectured connections between sensory neurons, motor neurons, and memory regions in the neocortex.

6.2 Future Directions

This dissertation can be seen as a proof of concept on building intelligent robots that interact with the environment through predicting action consequences based on memory. There are many directions for extending this proposed framework to achieve a reliable system capable of solving more complicated tasks. As shown in Figure 6.1, this dissertation only covers parts of the proposed conceptual diagram of the neocortex and there are many additional experiments that can provide insights on how to build more robust robots and reveal a more complete picture of the neocortex. I discuss a few possible future directions in the following.

6.2.1 Haptic

In this dissertations, force information is incorporated into the aspect representation but is not used in controllers directly. In addition, only simple force features are considered. In future work, I would like to investigate in more complex hierarchical haptic features that matches the proposed visual features. A higher layer may represent more abstract notions such as force closure and insertion. These haptic features should also provide information for the hierarchical controllers to execute tasks that have to rely on haptic feedback.

6.2.2 Hierarchical Aspect Transitions

The current implementation of the aspect transition graph contains features of different visual layers and connects them with controllers within different robotic subchains. However, transitions of aspect nodes in different layers are not considered separately. In future work, I would like to consider a more flexible model where aspect nodes of different layers transition based on features and actions of corresponding layers and information from neighbor layer aspect nodes.

6.2.3 Cross Modality Top Down Inference

The current framework can infer missing visual features through a top down process based on memory. If a hierarchical CNN feature is not detected bottom up but is expected to appear based on memory, it can be inferred based on the features stored in aspect nodes in a top down fashion. This can also be done between different modalities. For example, if the robot's haptic sensor touches an object, the robot should expect to see the object in contact with the robot. If it does not see this object based on bottom up detection, it should be able to enhance the detection through a top down process based on previous joint observations.

6.2.4 Generalizing to Object Categories

The experiments conducted in this dissertation either work on object instances or simple categories such as cuboids and cylinders. For a general purpose robot to be useful, it will need to be able to interact with complex novel objects. Based on the recent success on deep learning approaches on object classification, storing categorical representations in an aspect reliably is possible. By learning the object class through a large amount of images and interaction with a small amount of demonstrations, meaningful hierarchical CNN features that can support actions and generalize across categories can be identified.

6.2.5 Planning Across Hierarchies

The planner used in this work can identify a path from the current aspect node to a goal aspect node on a different ATG model. This allows the robot to plan sequence of actions based on demonstrations of subtasks. With a hierarchical aspect transition graph, a planner that can plan across different hierarchies can solve novel tasks in a more creative way. A higher level aspect transition may correspond to categorical behaviors, while a middle level aspect transition may be related to instance manipulation. Being able to transition from instance level to categorical level planning

when a novel observation occurs may allow the robot to test out combinations of learned interactions in a creative way.

6.2.6 Learning Through Intrinsic Motivation

In this dissertation, memory models can be created through explorations, or learned from demonstrations. Learning action transitions through exploration allows the robot to acquire models of the world autonomously, but may require a significant amount of time to experience meaningful interactions in a high dimensional space. Intrinsic motivation can possibly be used to guide these explorations. By rewarding actions that may identify consistent causal relationships of action and observation, the created memory models would only consist of scenarios where actions have predictable consequences. These memories on actions that can achieve consistent outcomes may be sufficient to accomplish many tasks.

♫

The framework introduced in this dissertation is by no means comparable to the real intelligence all of us possess. The purpose of this dissertation is not to show a proof or solution but to ignite your imagination. By now, I hope some of you would be intrigued and join me on this lifelong journey on understanding the mystery of intelligence.

BIBLIOGRAPHY

[1] Akgun, Baris, Cakmak, Maya, Yoo, Jae Wook, and Thomaz, Andrea Lockerd. Trajectories and keyframes for kinesthetic teaching: A human-robot interaction perspective. In *Proceedings of the seventh annual ACM/IEEE international conference on Human-Robot Interaction* (2012), ACM, pp. 391–398.

[2] Alexandrova, Sonya, Cakmak, Maya, Hsiao, Kaijen, and Takayama, Leila. Robot programming by demonstration with interactive action visualizations. In *Robotics: science and systems* (2014).

[3] Amir, Eyal, and Chang, Allen. Learning partially observable deterministic action models. *Journal of Artificial Intelligence Research 33* (2008), 349–402.

[4] Baldi, Pierre. A computational theory of surprise. In *Information, Coding and Mathematics*. Springer, 2002, pp. 1–25.

[5] Bay, Herbert, Tuytelaars, Tinne, and Van Gool, Luc. SURF: Speeded up robust features. In *Computer vision–ECCV 2006*. Springer, 2006, pp. 404–417.

[6] Berlyne, Daniel E. Conflict, arousal, and curiosity.

[7] Bo, Liefeng, Ren, Xiaofeng, and Fox, Dieter. Unsupervised feature learning for rgb-d based object recognition. In *Experimental Robotics* (2013), Springer, pp. 387–402.

[8] Bülthoff, Heinrich H, and Edelman, Shimon. Psychophysical support for a two-dimensional view interpolation theory of object recognition. *Proceedings of the National Academy of Sciences 89*, 1 (1992), 60–64.

[9] Burridge, Robert R, Rizzi, Alfred A, and Koditschek, Daniel E. Sequential composition of dynamically dexterous robot behaviors. *The International Journal of Robotics Research 18*, 6 (1999), 534–555.

[10] Calinon, Sylvain, and Billard, Aude. A probabilistic programming by demonstration framework handling constraints in joint space and task space. In *Intelligent Robots and Systems, 2008. IROS 2008. IEEE/RSJ International Conference on* (2008), IEEE, pp. 367–372.

[11] Calinon, Sylvain, Guenter, Florent, and Billard, Aude. On learning, representing, and generalizing a task in a humanoid robot. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics) 37*, 2 (2007), 286–298.

[12] Casti, John L. *Complexification Explaining a Paradoxical World Through the Science of Surprise*. HarperCollinsPublishers, 1994.

[13] Dame, Amaury, and Marchand, Eric. Entropy-based visual servoing. In *Robotics and Automation, 2009. ICRA'09. IEEE International Conference on* (2009), IEEE, pp. 707–713.

[14] Diftler, Myron A, Mehling, JS, Abdallah, Muhammad E, Radford, Nicolaus A, Bridgwater, Lyndon B, Sanders, Adam M, Askew, Roger Scott, Linn, D Marty, Yamokoski, John D, Permenter, FA, et al. Robonaut 2-the first humanoid robot in space. In *Robotics and Automation (ICRA), 2011 IEEE International Conference on* (2011), IEEE, pp. 2178–2183.

[15] Dinh, Paul, and Hart, Stephen. NASA Robonaut 2 Simulator, 2013. [Online; accessed 7-July-2014].

[16] Donald, Bruce R. A geometric approach to error detection and recovery for robot motion planning with uncertainty. *Artificial Intelligence 37*, 1 (1988), 223–271.

[17] Edelman, Shimon, and Bülthoff, Heinrich H. Orientation dependence in the recognition of familiar and novel views of three-dimensional objects. *Vision research 32*, 12 (1992), 2385–2400.

[18] Faugeras, Olivier, Mundy, Joe, Ahuja, Narendra, Dyer, Charles, Pentland, Alex, Jain, Ramesh, Ikeuchi, Katsushi, and Bowyer, Kevin. Why aspect graphs are not (yet) practical for computer vision. *CVGIP: Image Understanding 55*, 2 (1992), 212–218.

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

[20] Finn, Chelsea, and Levine, Sergey. Deep visual foresight for planning robot motion. In *Robotics and Automation (ICRA), 2017 IEEE International Conference on* (2017), IEEE, pp. 2786–2793.

[21] Finn, Chelsea, Tan, Xin Yu, Duan, Yan, Darrell, Trevor, Levine, Sergey, and Abbeel, Pieter. Deep spatial autoencoders for visuomotor learning. *reconstruction 117*, 117 (2015), 240.

[22] Fischler, Martin A, and Bolles, Robert C. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM 24*, 6 (1981), 381–395.

[23] George, Dileep, and Hawkins, Jeff. Towards a mathematical theory of cortical micro-circuits. *PLoS Comput Biol 5*, 10 (2009), e1000532.

[24] Gibson, J. J. *The Ecological Approach to Visual Perception*. Houghton Mifflin, Boston, 1979.

[25] Gibson, James J. Perceiving, acting, and knowing: Toward an ecological psychology. *chap. The Theory of Affordance). Michigan: Lawrence Erlbaum Associates* (1977).

[26] Gigus, Ziv, and Malik, Jitendra. Computing the aspect graph for line drawings of polyhedral objects. *Pattern Analysis and Machine Intelligence, IEEE Transactions on 12*, 2 (1990), 113–122.

[27] Goldstein, E Bruce. The ecology of jj gibson's perception. *Leonardo* (1981), 191–195.

[28] Goodale, Melvyn, and Milner, David. *Sight unseen: An exploration of conscious and unconscious vision*. OUP Oxford, 2013.

[29] Grabner, Helmut, Gall, Juergen, and Van Gool, Luc. What makes a chair a chair? In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on* (2011), IEEE, pp. 1529–1536.

[30] Hariharan, Bharath, Arbeláez, Pablo, Girshick, Ross, and Malik, Jitendra. Hypercolumns for object segmentation and fine-grained localization. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2015), pp. 447–456.

[31] Harlow, Harry F. Learning and satiation of response in intrinsically motivated complex puzzle performance by monkeys. *Journal of Comparative and Physiological Psychology 43*, 4 (1950), 289.

[32] Hart, Stephen W. *The development of hierarchical knowledge in robot systems*. PhD thesis, University of Massachusetts Amherst, 2009.

[33] Hawkins, Jeff, and Blakeslee, Sandra. *On intelligence*. Macmillan, 2007.

[34] Herzog, Alexander, Pastor, Peter, Kalakrishnan, Mrinal, Righetti, Ludovic, Bohg, Jeannette, Asfour, Tamim, and Schaal, Stefan. Learning of grasp selection based on shape-templates. *Autonomous Robots 36*, 1-2 (2014), 51–65.

[35] Hoffmann, Frank, Nierobisch, Thomas, Seyffarth, Torsten, and Rudolph, Günter. Visual servoing with moments of sift features. In *Systems, Man and Cybernetics, 2006. SMC'06. IEEE International Conference on* (2006), vol. 5, IEEE, pp. 4262–4267.

[36] Huber, Manfred. *A hybrid architecture for adaptive robot control*. PhD thesis, University of Massachusetts Amherst, 2000.

[37] Hull, Clark Leonard. Principles of behavior: an introduction to behavior theory.

[38] Hutchinson, Seth, Hager, Gregory D, and Corke, Peter I. A tutorial on visual servo control. *Robotics and Automation, IEEE Transactions on 12*, 5 (1996), 651–670.

[39] Itti, Laurent, and Baldi, Pierre F. Bayesian surprise attracts human attention. In *Advances in neural information processing systems* (2005), pp. 547–554.

[40] Jägersand, Martin, and Nelson, Randal. On-line estimation of visual-motor models using active vision. *image 11* (1996), 1.

[41] Jia, Yangqing, Shelhamer, Evan, Donahue, Jeff, Karayev, Sergey, Long, Jonathan, Girshick, Ross, Guadarrama, Sergio, and Darrell, Trevor. Caffe: Convolutional architecture for fast feature embedding. *arXiv preprint arXiv:1408.5093* (2014).

[42] Kaelbling, Leslie Pack, Littman, Michael L, and Cassandra, Anthony R. Planning and acting in partially observable stochastic domains. *Artificial intelligence 101*, 1 (1998), 99–134.

[43] Kato, Hirokazu, and Billinghurst, Mark. Marker tracking and hmd calibration for a video-based augmented reality conferencing system. In *Augmented Reality, 1999.(IWAR'99) Proceedings. 2nd IEEE and ACM International Workshop on* (1999), IEEE, pp. 85–94.

[44] Katz, Dov, Venkatraman, Arun, Kazemi, Moslem, Bagnell, J Andrew, and Stentz, Anthony. Perceiving, learning, and exploiting object affordances for autonomous pile manipulation. *Autonomous Robots 37*, 4 (2014), 369–382.

[45] Koenderink, Jan J, and Van Doorn, Andrea J. The internal representation of solid shape with respect to vision. *Biological cybernetics 32*, 4 (1979), 211–216.

[46] Konidaris, George. Constructing abstraction hierarchies using a skill-symbol loop. *arXiv preprint arXiv:1509.07582* (2015).

[47] Kriegman, David J, and Ponce, Jean. Computing exact aspect graphs of curved objects: Solids of revolution. *International Journal of Computer Vision 5*, 2 (1990), 119–135.

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

[49] Lai, Kevin, Bo, Liefeng, Ren, Xiaofeng, and Fox, Dieter. A large-scale hierarchical multi-view rgb-d object dataset. In *Robotics and Automation (ICRA), 2011 IEEE International Conference on* (2011), IEEE, pp. 1817–1824.

[50] Lai, Kevin, Bo, Liefeng, Ren, Xiaofeng, and Fox, Dieter. A scalable tree-based approach for joint object and pose recognition. In *AAAI* (2011).

[51] LeCun, Yann, and Bengio, Yoshua. Convolutional networks for images, speech, and time series. *The handbook of brain theory and neural networks 3361*, 10 (1995), 1995.

[52] Lee, Tai Sing, and Mumford, David. Hierarchical bayesian inference in the visual cortex. *JOSA A 20*, 7 (2003), 1434–1448.

[53] Lenz, Ian, Lee, Honglak, and Saxena, Ashutosh. Deep learning for detecting robotic grasps. *The International Journal of Robotics Research 34*, 4-5 (2015), 705–724.

[54] Leonard, John J, and Durrant-Whyte, Hugh F. Simultaneous map building and localization for an autonomous mobile robot. In *Intelligent Robots and Systems' 91.'Intelligence for Mechanical Systems, Proceedings IROS'91. IEEE/RSJ International Workshop on* (1991), Ieee, pp. 1442–1447.

[55] Leung, Cindy, Huang, Shoudong, Kwok, Ngai, and Dissanayake, Gamini. Planning under uncertainty using model predictive control for information gathering. *Robotics and Autonomous Systems 54*, 11 (2006), 898–910.

[56] Levine, Sergey, Finn, Chelsea, Darrell, Trevor, and Abbeel, Pieter. End-to-end training of deep visuomotor policies. *arXiv preprint arXiv:1504.00702* (2015).

[57] Logothetis, Nikos K, Pauls, Jon, and Poggio, Tomaso. Shape representation in the inferior temporal cortex of monkeys. *Current Biology 5*, 5 (1995), 552–563.

[58] Mnih, Volodymyr, Kavukcuoglu, Koray, Silver, David, Rusu, Andrei A, Veness, Joel, Bellemare, Marc G, Graves, Alex, Riedmiller, Martin, Fidjeland, Andreas K, Ostrovski, Georg, et al. Human-level control through deep reinforcement learning. *Nature 518*, 7540 (2015), 529.

[59] Montgomery, Kay C. The role of the exploratory drive in learning. *Journal of Comparative and Physiological Psychology 47*, 1 (1954), 60.

[60] More, Jorge J, and Trangenstein, John Arthur. On the global convergence of broydens method. *Mathematics of Computation 30*, 135 (1976), 523–540.

[61] Nakamura, Yoshihiko. Advanced robotics: redundancy and optimization, 1991.

[62] Palmeri, Thomas J, and Gauthier, Isabel. Visual object understanding. *Nature Reviews Neuroscience 5*, 4 (2004), 291–303.

[63] Pas, Andreas ten, and Platt, Robert. Using geometry to detect grasps in 3d point clouds. *arXiv preprint arXiv:1501.03100* (2015).

[64] Pastor, Peter, Hoffmann, Heiko, Asfour, Tamim, and Schaal, Stefan. Learning and generalization of motor skills by learning from demonstration. In *Robotics and Automation, 2009. ICRA'09. IEEE International Conference on* (2009), IEEE, pp. 763–768.

[65] Patil, Sachin, Duan, Yan, Schulman, John, Goldberg, Ken, and Abbeel, Pieter. Gaussian belief space planning with discontinuities in sensing domains. In *Int. Symp. on Robotics Research (ISRR)(in review)* (2013).

[66] Pérez-D'Arpino, Claudia, and Shah, Julie A. C-learn: Learning geometric constraints from demonstrations for multi-step manipulation in shared autonomy. In *IEEE International Conference on Robotics and Automation* (2017).

[67] Phillips, Mike, Hwang, Victor, Chitta, Sachin, and Likhachev, Maxim. Learning to plan for constrained manipulation from demonstrations. In *Robotics: Science and Systems* (2013), vol. 5.

[68] Pineau, Joelle, Gordon, Geoff, Thrun, Sebastian, et al. Point-based value iteration: An anytime algorithm for pomdps. In *IJCAI* (2003), vol. 3, pp. 1025–1032.

[69] Pinto, Lerrel, and Gupta, Abhinav. Supersizing self-supervision: Learning to grasp from 50k tries and 700 robot hours. *arXiv preprint arXiv:1509.06825* (2015).

[70] Platt, R., Tedrake, R., Kaelbling, L., and Lozano-Perez, T. Belief space planning assuming maximum likelihood observations. In *Proceedings of Robotics: Science and Systems* (Zaragoza, Spain, June 2010).

[71] Platt, Robert, Grupen, Roderic A, and Fagg, Andrew H. Re-using schematic grasping policies. In *Humanoid Robots, 2005 5th IEEE-RAS International Conference on* (2005), IEEE, pp. 141–147.

[72] Platt, Robert, Kaelbling, Leslie, Lozano-Perez, Tomas, and Tedrake, Russ. Efficient planning in non-gaussian belief spaces and its application to robot grasping. In *International Symposium on Robotics Research* (2011).

[73] Poggio, Tomaso, and Edelman, Shimon. A network that learns to recognize 3d objects. *Nature 343*, 6255 (1990), 263–266.

[74] Rodriguez, Alberto, Mason, Matthew T, Srinivasa, Siddhartha S, Bernstein, Matthew, and Zirbel, Alex. Abort and retry in grasping. In *Intelligent Robots and Systems (IROS), 2011 IEEE/RSJ International Conference on* (2011), IEEE, pp. 1804–1810.

[75] Ruiken, Dirk, Lanighan, Michael W, Grupen, Roderic, et al. Postural modes and control for dexterous mobile manipulation: the umass ubot concept. In *Humanoid Robots (Humanoids), 2013 13th IEEE-RAS International Conference on* (2013), IEEE, pp. 280–285.

[76] Russakovsky, Olga, Deng, Jia, Su, Hao, Krause, Jonathan, Satheesh, Sanjeev, Ma, Sean, Huang, Zhiheng, Karpathy, Andrej, Khosla, Aditya, Bernstein, Michael, Berg, Alexander C., and Fei-Fei, Li. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV) 115*, 3 (2015), 211–252.

[77] Rusu, Radu Bogdan, and Cousins, Steve. 3d is here: Point cloud library (pcl). In *IEEE International Conference on Robotics and Automation (ICRA)* (Shanghai, China, May 9-13 2011).

[78] Saxena, Ashutosh, Driemeyer, Justin, and Ng, Andrew Y. Robotic grasping of novel objects using vision. *The International Journal of Robotics Research 27*, 2 (2008), 157–173.

[79] Saxena, Ashutosh, Wong, Lawson LS, and Ng, Andrew Y. Learning grasp strategies with partial shape information. In *AAAI* (2008), vol. 3, pp. 1491–1494.

[80] Schwarz, Max, Schulz, Hannes, and Behnke, Sven. Rgb-d object recognition and pose estimation based on pre-trained convolutional neural network features. In *2015 IEEE International Conference on Robotics and Automation (ICRA)* (2015), IEEE, pp. 1329–1335.

[81] Sen, Shiraj. *Bridging the gap between autonomous skill learning and task-specific planning*. PhD thesis, University of Massachusetts Amherst, 2013.

[82] Shademan, Azad, and Janabi-Sharifi, Farrokh. Using scale-invariant feature points in visual servoing. In *Optics East* (2004), International Society for Optics and Photonics, pp. 63–70.

[83] Simonyan, Karen, Vedaldi, Andrea, and Zisserman, Andrew. Deep inside convolutional networks: Visualising image classification models and saliency maps. *arXiv preprint arXiv:1312.6034* (2013).

[84] Smallwood, Richard D, and Sondik, Edward J. The optimal control of partially observable markov processes over a finite horizon. *Operations Research 21*, 5 (1973), 1071–1088.

[85] Sondik, Edward Jay. The optimal control of partially observable markov processes. Tech. rep., DTIC Document, 1971.

[86] Springenberg, Jost Tobias, Dosovitskiy, Alexey, Brox, Thomas, and Riedmiller, Martin. Striving for simplicity: The all convolutional net. *arXiv preprint arXiv:1412.6806* (2014).

[87] Stoytchev, Alexander. Toward learning the binding affordances of objects: A behavior-grounded approach. In *Proceedings of AAAI Symposium on Developmental Robotics* (2005), pp. 17–22.

[88] Sucan, Ioan A., and Chitta, Sachin. Moveit!, 2013. [Online].

[89] Sutton, Richard S, Precup, Doina, and Singh, Satinder. Between mdps and semi-mdps: A framework for temporal abstraction in reinforcement learning. *Artificial intelligence 112*, 1 (1999), 181–211.

[90] Tarr, Michael J, and Bülthoff, Heinrich H. Image-based object recognition in man, monkey and machine. *Cognition 67*, 1 (1998), 1–20.

[91] Thrun, Sebastian, Burgard, Wolfram, and Fox, Dieter. *Probabilistic robotics*. MIT press, 2005.

[92] Ullman, Shimon, and Basri, Ronen. Recognition by linear combinations of models. *IEEE transactions on pattern analysis and machine intelligence 13*, 10 (1991), 992–1006.

[93] Van Den Bos, Esther, and Jeannerod, Marc. Sense of body and sense of action both contribute to self-recognition. *Cognition 85*, 2 (2002), 177–187.

[94] Varadarajan, Karthik Mahesh, and Vincze, Markus. Object part segmentation and classification in range images for grasping. In *Advanced Robotics (ICAR), 2011 15th International Conference on* (2011), IEEE, pp. 21–27.

[95] Varadarajan, Karthik Mahesh, and Vincze, Markus. Afrob: The affordance network ontology for robots. In *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on* (2012), IEEE, pp. 1343–1350.

[96] Wang, Xuemei. *Learning planning operators by observation and practice*. PhD thesis, Carnegie Mellon University, 1996.

[97] Wilkinson, Eric, and Takahashi, Takeshi. Efficient aspect object models using pre-trained convolutional neural networks. In *Humanoid Robots (Humanoids), 2015 IEEE-RAS 15th International Conference on* (2015), IEEE, pp. 284–289.

[98] Wise, Melonee, and Ciocarlie, Matei. ICRA Manipulation Demo, 2010. [Online; accessed 19-September-2015].

[99] Wörgötter, Florentin, Geib, Chris, Tamosiunaite, Minija, Aksoy, Eren Erdal, Piater, Justus, Xiong, Hanchen, Ude, Ales, Nemec, Bojan, Kraft, Dirk, Krüger, Norbert, et al. Structural bootstrappinga novel, generative mechanism for faster and more efficient acquisition of action-knowledge. *IEEE Transactions on Autonomous Mental Development 7*, 2 (2015), 140–154.

[100] Yang, Qiang, Wu, Kangheng, and Jiang, Yunfei. Learning action models from plan examples using weighted max-sat. *Artificial Intelligence 171*, 2 (2007), 107–143.

[101] Yosinski, Jason, Clune, Jeff, Nguyen, Anh, Fuchs, Thomas, and Lipson, Hod. Understanding neural networks through deep visualization. *arXiv preprint arXiv:1506.06579* (2015).

[102] Zeiler, Matthew D, and Fergus, Rob. Visualizing and understanding convolutional networks. In *Computer vision–ECCV 2014*. Springer, 2014, pp. 818–833.

[103] Zhang, Jianming, Lin, Zhe, Brandt, Jonathan, Shen, Xiaohui, and Sclaroff, Stan. Top-down neural attention by excitation backprop. In *European Conference on Computer Vision* (2016), Springer, pp. 543–559.

[104] Zhang, Li Emma, Ciocarlie, Matei, and Hsiao, Kaijen. Grasp evaluation with graspable feature matching. In *RSS Workshop on Mobile Manipulation: Learning to Manipulate* (2011).