Computer Science and Artificial Intelligence Laboratory

Technical Report

MIT-CSAIL-TR-2007-004 January 16, 2007

Robot Manipulation in Human Environments

Aaron Edsinger

massachusetts institute of technology, cambridge, ma 02139 usa — www.csail.mit.edu

Robot Manipulation in Human Environments

by

Aaron Ladd Edsinger

B.S., Stanford University (1994)
S.M., Massachusetts Institute of Technology(2001)
Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy in Electrical Engineering and Computer Science

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

January 2007

Author .....
Department of Electrical Engineering and Computer Science
January 16, 2007

Certified by .....
Rodney A. Brooks
Professor of Computer Science and Engineering
Thesis Supervisor

Accepted by .....
Arthur C. Smith
Chairman, Department Committee on Graduate Students

Robot Manipulation in Human Environments

by

Aaron Ladd Edsinger

Submitted to the Department of Electrical Engineering and Computer Science
on January 16, 2007, in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy in Electrical Engineering and Computer Science

Abstract

Human environments present special challenges for robot manipulation. They are often dynamic, difficult to predict, and beyond the control of a robot engineer. Fortunately, many characteristics of these settings can be used to a robot's advantage. Human environments are typically populated by people, and a robot can rely on the guidance and assistance of a human collaborator. Everyday objects exhibit common, task-relevant features that reduce the cognitive load required for the object's use. Many tasks can be achieved through the detection and control of these sparse perceptual features. And finally, a robot is more than a passive observer of the world. It can use its body to reduce its perceptual uncertainty about the world.

In this thesis we present advances in robot manipulation that address the unique challenges of human environments. We describe the design of a humanoid robot named Domo, develop methods that allow Domo to assist a person in everyday tasks, and discuss general strategies for building robots that work alongside people in their homes and workplaces.

Thesis Supervisor: Rodney A. Brooks
Title: Professor of Computer Science and Engineering

Acknowledgments

Foremost, I want to thank Rod Brooks for his support over the years. His creativity, openness to far out ideas, and belief in his students has been a continual source of inspiration. Rod gave me the opportunity to build an unexpectedly complex and expensive robot, and for that I am very appreciative. Especially because I had already started building it before asking his permission...Also, thank you to the other members of my committee, Rod Grupen and Daniela Rus, for their thoughtful guidance and encouragement on this thesis.

It has been seven years of three a.m. bicycle rides home from lab, curry tofu at the food trucks, beers at River Gods, art openings with the Collision Collective, BBQs with Busy People, and a million other little details that have defined this time in Cambridge. MIT and CSAIL are remarkable places, certainly unlike any other in the world. If only they would open a San Francisco office...I am grateful for all the staff, professors, and students that make it, well, MIT. The sentiment especially extends to Ron Wiken, who's good humor and incredible stock room are very much appreciated...Many thanks to Charlie Kemp for his invaluable discussions and significant contributions to this work, your constant curiosity and consumption of Coke and Saltines never fails to amaze me. As does Adam Kumpf's great enthusiasm for building most anything. Arm bashes to Dan Paluska, who not only soldered all of Domo's amplifiers, but has also been a good friend and the genesis of many good things around MIT. Pia Lindman and her double, and Jonathan Bachrach, the 300lb gorilla-artist in the room, get much respect as well...Also, thank you to all in the Living Breathing Robots group, Lijin Aryananda, Una-May O'Reilly, Eduardo Torres-Jara, Juan Velasquez, Varun Aggarwal, and Ann Whittaker. Word to Frank Gehry for his beautiful building that I have called home. This work was sponsored by the NASA Systems Mission Directorate, Technical Development Program under contract 012461-001, and by Toyota Motor Corporation under the contract for Partner Robots in the Home. I am very appreciated of their support, and in particular, the guidance of Yuichiro Nakajima and Takemitsu Mori.

To my friends who have inspired, cared, and shared many a drink over the years, thank you! This means you, Amy and Parul. Jessica Banks, though not given to sharing drinks, has made up for it with years of supportive friendship, through the many ups and downs on the path to this dissertation. It goes without saying that Jeff Weber, a brilliant robot designer and Domo's illegitimate father, has also been a great friend. Not only would Domo not exist without his talents, but I likely would be living in a van in Mexico if not for him. Damn you, Jeff...

Kelly McDermott gets her very own paragraph in gratitude for her selfless friendship, hand delivered baked goods, many never-a-dull-moments, giant orange dog, and for not really caring what Domo does or does not do.

I'm still not sure how I went from Enumclaw farm kid to MIT farm kid, but I'm pretty sure my family had everything to do with it. My bros and sis, Eric, Craig, and Carrie are all amazingly eccentric, brilliant, and creative...I must thank Grandpa Edsinger, though we never got to meet, because the family drive to create, build, and invent must originate, in part, with him. And to my parents, thank you for your constant love and support, it means everything.

To my parents,
Richard and Yolanda.

Contents

1 Introduction **14**

1.1 Human Environments 15

1.2 Robots Doing Useful Work 16

1.2.1 The *Sense-Model-Plan-Act* Approach 17

1.2.2 The Behavior-Based Approach 18

1.3 The Robot Domo 19

1.3.1 Partner Robots 21

1.3.2 Creature Robots 21

1.4 Manipulation in Human Environments: Our Approach 22

1.5 Contributions 23

1.6 Roadmap 24

2 Three Themes for Design **27**

2.1 Let the Body do the Thinking 27

2.1.1 Human Form 27

2.1.2 Designing for Uncertainty 28

2.1.3 Taking Action 29

2.2 Cooperative Manipulation 31

2.2.1 Assistive Robotics 32

2.2.2 Collaborative Cues 34

2.3 Task Relevant Features . . . 36
2.3.1 Background . . . 37
2.3.2 Perceptual Robustness . . . 38
2.3.3 Generalization . . . 40

3 Recent Work **41**
3.1 Connell: Behavior-Based Manipulation . . . 42
3.2 Metta and Fitzpatrick: Poking and Affordances . . . 43
3.3 Robonaut: Collaborative Tool Use . . . 44
3.4 Dexter: Manipulation Gaits and Grasp Features . . . 44
3.5 STAIR: Learning of Grasp Points . . . 46

4 Building Bodies **47**
4.1 Notation . . . 48
4.1.1 Pinhole Camera Model . . . 50
4.2 Compliant Actuators . . . 51
4.2.1 Design . . . 51
4.2.2 Control . . . 55
4.3 Arms . . . 55
4.3.1 Arm Design . . . 55
4.3.2 Arm Control . . . 58
4.3.3 Inverse Kinematic Control . . . 59
4.4 Hands . . . 60
4.4.1 Hand Design . . . 63
4.5 Head . . . 63
4.5.1 Head Design . . . 63
4.5.2 Head Control . . . 65
4.6 Manipulator Safety . . . 66
4.6.1 Measuring the Head Injury Criterion . . . 67
4.7 Discussion . . . 70

5 Building Brains **71**
5.1 Computational Organization . . . 71

5.2 SLATE: A Behavior-Based Architecture . . . 74
5.2.1 SLATE Components . . . 75
5.2.2 Example Program . . . 78
5.2.3 SLATE on Domo . . . 79
5.3 Designing Tasks in SLATE . . . 79
5.3.1 Coordinating Modules . . . 80
5.3.2 Decomposing Manipulation Tasks . . . 81
5.3.3 An Algorithm for Manual Skills . . . 84
5.4 Discussion . . . 86

6 Visual Attention System **87**
6.1 Visual Motion . . . 89
6.1.1 Visual Motion Model . . . 89
6.1.2 Visual Motion Prediction . . . 91
6.2 *InterestRegions* . . . 94
6.3 The Sensory EgoSphere . . . 96
6.3.1 Features in the SES . . . 99
6.3.2 Spatial Distributions in the SES . . . 100

7 Let the Body do the Thinking **102**
7.1 *StiffnessAdapt* . . . 102
7.2 *ContactDetect* . . . 104
7.2.1 Contact Motion . . . 104
7.2.2 Contact Forces . . . 106
7.3 *GraspAperture* and *GraspDetect* . . . 109
7.4 *SurfaceTest* . . . 111
7.4.1 Results . . . 113
7.5 *SurfacePlace* . . . 116
7.5.1 Results . . . 117
7.6 Discussion . . . 120

8 Cooperative Manipulation **121**
8.1 Perception of People . . . 121

8.1.1 Detecting Hands, Fingers, and Faces . . . 122
8.1.2 *PersonDetect* . . . 125
8.1.3 *PersonSeek* . . . 128
8.1.4 *VocalRequest* . . . 129
8.2 *AssistedGrasp* and *AssistedGive* . . . 129
8.3 Testing Cooperative Manipulation . . . 132
8.3.1 The Give and Take Experiment . . . 133
8.4 Discussion . . . 135

9 Task Relevant Features **139**
9.1 The Task Relevant Tool Tip . . . 140
9.1.1 Related Work in Robot Tool Use . . . 141
9.1.2 Tool Tip Detection . . . 142
9.1.3 Probabilistic Estimation of the 3D Tool Tip Position . . . 142
9.1.4 Tool Tip Estimation Results . . . 145
9.1.5 Discussion . . . 149
9.2 Control of the Tip . . . 150
9.2.1 *TipPose* . . . 150
9.2.2 *TipServo* . . . 152
9.2.3 Results . . . 154
9.3 Moving from Tool Tips to Everyday Objects . . . 154
9.3.1 Results . . . 156
9.4 Learning the Task Specific Prior . . . 158
9.4.1 Density Estimation . . . 159
9.4.2 Results . . . 161
9.5 Working with Tips . . . 164
9.5.1 *TipPriors* . . . 165
9.5.2 *TipEstimate* . . . 165
9.5.3 *WristWiggle* . . . 165
9.5.4 *TipUse* . . . 166
9.6 The Task Relevant Hand . . . 166
9.6.1 *PalmServo* . . . 168

9.7 Discussion . . . 169

10 Putting It All Together **170**

10.1 *SwitchHands* . . . 170

10.1.1 Results . . . 173

10.2 *BimanualFixture* . . . 175

10.2.1 *BimanualServo* . . . 177

10.2.2 Results . . . 179

10.3 *ContainerInsert* . . . 183

10.3.1 *ContainerPlace* . . . 184

10.3.2 Results . . . 185

10.4 *PuttingStuffAway* . . . 188

10.4.1 Results . . . 190

10.4.2 Discussion . . . 192

10.5 *HelpWithChores* . . . 193

10.6 Discussion . . . 195

11 Conclusions and Future Directions **199**

11.1 Contributions . . . 200

11.2 How Far Does This Extend? . . . 201

11.3 Lessons Learned . . . 203

11.4 Going Forward . . . 205

11.4.1 A Suite of Task Relevant Features . . . 205

11.4.2 More Dynamics, More Adaptation . . . 205

11.4.3 Learning from Demonstration . . . 207

11.5 Final Remarks . . . 207

A Module Summary **208**

List of Figures

1-1 Idealized versus actual human environments 15
1-2 The robot Domo 20
1-3 Organizational roadmap 26

2-1 *Coffeepot for Masochists* 36
2-2 Domo's office environment 39

4-1 Robot coordinate frames 49
4-2 Pinhole camera model 50
4-3 Series Elastic Actuator designs 52
4-4 Control topology for the SEA 54
4-5 Mechanical design of the arm 56
4-6 Cable-drive layout of the arm 57
4-7 Manipulator inverse kinematics 59
4-8 Domo's hand. 61
4-9 The mechanical design of Domo's hands 62
4-10 Passive compliance in the hand 63
4-11 Mechanical design of the head 64
4-12 Manipulator mass distribution 67
4-13 The Head Injury Criterion 69

5-1 Computational architecture 72

5-2 SLATE control organization 75
5-3 The flow of control within a SLATE module 81
5-4 Task decomposition. 82
5-5 Control algorithm for manual skills. 85

6-1 Visual attention system 88
6-2 Kinematic prediction in the visual motion model 91
6-3 Voting by the interest point operator 95
6-4 Execution of the *InterestRegions* module 97
6-5 The Sensory EgoSphere. 98
6-6 View of the Sensory EgoSphere. 98

7-1 Manipulator stiffness versus controller response 103
7-2 Detecting external contact 105
7-3 Torque prediction during reaching. 107
7-4 Torque prediction error histograms 108
7-5 Prediction of the grasp aperture 110
7-6 The *SurfaceTest* module 112
7-7 Reaching trajectories during *SurfaceTest* 114
7-8 Execution of the *SurfaceTest* module 114
7-9 The *SurfacePlace* module 115
7-10 Testing *SurfacePlace* robustness 117
7-11 Objects tested with *SurfacePlace* 119
7-12 Stability estimates of *SurfacePlace* 119

8-1 Detection of hands, fingers, and faces 122
8-2 Sample detections by *InterestRegions* 123
8-3 Feature category statistics for human-robot interaction 124
8-4 The *PersonDetect* module 124
8-5 Output from the face tracker and skin detector. 126
8-6 Spatial distribution of faces in the SES 126
8-7 Detection of hand waving 127
8-8 The *AssistedGrasp* module 130

8-9 Execution of the *AssistedGrasp* module . . . 131
8-10 Setup of the Give and Take experiment . . . 133
8-11 Sample grasp alignment errors . . . 136
8-12 Results for grasp alignment . . . 137

9-1 View of the tip estimation process . . . 140
9-2 Geometry for 3D tip estimation . . . 143
9-3 The tools used for tip estimation . . . 145
9-4 Sample tip detections . . . 146
9-5 Mean prediction error for each tool . . . 147
9-6 Tip prediction for each tool . . . 148
9-7 Tip detection error histogram . . . 149
9-8 Control of the tool tip . . . 151
9-9 Results from *TipPose* and *TipServo* . . . 155
9-10 Estimation of the generalized tool tip . . . 157
9-11 Average tip appearance model . . . 158
9-12 The task specific prior . . . 159
9-13 Distributions for the task specific priors . . . 162
9-14 Learning the task specific priors . . . 163
9-15 The *TipUse* module . . . 167
9-16 View of *Palm*Servo execution . . . 168

10-1 The *SwitchHands* module . . . 171
10-2 Execution of the *SwitchHands* module . . . 172
10-3 Experiment results for *SwitchHands* . . . 173
10-4 Visual realignment during *SwitchHands* . . . 174
10-5 The *BimanualFixture* module . . . 176
10-6 Illustration of bimanual fixturing . . . 177
10-7 Bimanual grasp of three boxes . . . 179
10-8 Disturbance response during bimanual fixturing . . . 180
10-9 Execution of the *BimanualFixture* module . . . 181
10-10 The *ContainerInsert* module . . . 182
10-11 The *ContainerPlace* module . . . 184

10-12 Execution of the *ContainerPlace* module 185
10-13 Execution of the *ContainerInsert* module 186
10-14 Results for *ContainerInsert* 187
10-15 *ContainerInsert* using a flexible hose 187
10-16 The *PuttingStuffAway* module 189
10-17 Execution of the *PuttingStuffAway* module 191
10-18 Three bottles tested with *PuttingStuffAway* 192
10-19 Experiment results for *PuttingStuffAway* 192
10-20 Execution of *PuttingStuffAway* on a deformable object 193
10-21 The *HelpWithChores* module 194
10-22 Execution of HelpWithChores, part one 196
10-23 Execution of HelpWithChores, part two 197

11-1 Task learning from demonstration 206

CHAPTER 1

Introduction

> I have seen the future and it doesn't work.
>
> –Robert Fullford

Robots have long been imagined as mechanical workers, helping us with the chores of daily life. Robots that can work alongside us in our homes and workplaces could extend the time an elderly person can live at home, provide physical assistance to a worker on an assembly line, or help with household chores. Already, robots can be found working in factories around the world, performing manipulation tasks with remarkable speed and precision. Why is it, then, that a robot can pick up and place a massive engine block with submillimeter precision but can't yet put away the groceries in a typical kitchen?

Everyday, human environments present special challenges for robot manipulation. They are often dynamic, difficult to predict, and beyond the control of a robot engineer. Fortunately, many characteristics of these settings can be used to a robot's advantage. Human environments are typically populated by people, and a robot can rely on the guidance and assistance of a human collaborator. Everyday objects exhibit common, task-relevant features that reduce the cognitive load required for the object's use. Many tasks can be achieved through the detection and control of these sparse perceptual features. And finally, a robot is more than a passive observer of the world. It can use its body to reduce its perceptual uncertainty about the world.

Figure 1-1: Traditional, model-based approaches to robot manipulation are well suited to idealized environments such as this kitchen(left). However, actual, everyday settings (right) can exhibit difficult to model effects such as variability in lighting, clutter, and unknown objects. In this thesis, we consider manipulation within this class of environments.

In this thesis we present advances in robot manipulation that address the unique challenges of human environments. We describe the design of a humanoid robot named Domo, develop methods that allow Domo to assist a person in everyday tasks, and discuss general strategies for building robots that work alongside people in their homes and workplaces.

1.1 Human Environments

As we see in Figure 1-1, everyday human environments have a number of defining characteristics that make life difficult for today's robots. Foremost, they are variable, dynamic, and difficult to model. The location of a couch or a lamp can change without the robot acting. There can be wide variation within related objects. Consider the variety of coffee cups found in the average home. Every home and workplace is different, and any attempt to enumerate their objects and their locations is sure to become incomplete and out of date. Human environments are also perceptually noisy. The appearance of a sunlit table changes throughout the day. Even the tidiest office or home is cluttered, and objects obstruct each other in unpredictable ways. Draping a coat over a chair can suddenly render the chair unrecognizable to a robot. Objects are often worn or dirty, making prior descriptions of their appearance inaccurate. Also, many of the items that people manipulate on a daily basis aren't rigid, violating a common assumption in robotics. Fitting a mathematical model to a crumpled towel, an extension cord, a newspaper, or a handful of dish soap seems ill-advised at best.

People are also present in human environments and they present their own unique challenges. A person may put away a tool in one of several locations in one of several orientations. While a roboticist will often knowingly stereotype their behavior for a robot, an untrained person will act in unpredictable ways. In human environments, people can no longer be fenced off from the robot's workspace. Robots that work with people must be safe for physical interaction. People also expect a robot to be responsive, operating in real time and adapting to their actions.

Finally, human environments have been built to accommodate the size and shape of human bodies. The size of a chair, the handle on a cabinet, the amount space between the coffee table and the couch are all designed around people. A robot working in a human environment will need to work with human sized tools, navigate human sized corridors, and perceive the world from a similar physical perspective as a person.

1.2 Robots Doing Useful Work

It seems that robots are always on the verge of moving into daily life, transforming the way we live and work. Commercially available robot toys and vacuum cleaners are already in our homes, entertaining and cleaning. The number of robots in human environments is expected to increase dramatically in coming years. However, before robots can realize their full potential in everyday life, they will need the ability to manipulate the world around them, as it is.

Imagine a newly purchased robot helper. You remove it from its box, power it on, and bring it into your kitchen. It has never seen your particular kitchen before, or the dishes in the dishwasher, but after some brief instruction, perhaps by demonstration, it begins putting away the dishes. Perhaps you remove the dishes from the dishwasher, hand them to your helper, and it puts them away in the cabinet. After finishing the dishes, it follows you into the kid's bedroom. Again, you demonstrate how to pick up and put away the clutter of toys on the floor, and the robot sets about its task.

Robots have a ways to go before they are doing this type of useful work in our homes. Currently, robots can perform impressive manipulation acts when the state of the world and the robot are known. These tasks are typically achieved by first sensing the world, updating an internal model to match the sensory state, updating a task plan, and then pro-

ducing a planned motor action. This approach is often characterized as *sense-model-plan-act* (Brooks, 1991a). Unfortunately, human environments are generally not amenable to internal models and the *sense-model-plan-act* approach. A popular alternative to *sense-model-plan-act* is the *behavior-based approach.* Behavior-based robotics has been demonstrably successful in robot navigation, but not yet robot manipulation within human environments. We now consider these two approaches in further detail.

1.2.1 The *Sense-Model-Plan-Act* Approach

Robot manipulation emerged from the industrial workplace around 40 years ago and quickly found success with the *sense-model-plan-act* approach. This was because the robot's body and workplace could be engineered to match a well defined internal model. On an automotive assembly line, the geometry of a car part is known precisely and any uncertainty about its pose in the world can be limited through alignment jigs. Of course, this requires careful design of the assembly line, and also very stiff, precise control of the robot. As a consequence, robot assembly lines are very costly to design and industrial manipulators are very dangerous around people.

In the early days of artificial intelligence, researchers concerned with manipulation followed the industrial paradigm and constructed static, homogeneous sets for their robots, complete with controlled lighting and polyhedral objects. This allowed researchers to continue within the *sense-model-plan-act* framework. As Brooks (1991a) points out, these carefully engineered environments allowed the robot to transform its sensory world into a model world. The robot could then plan its manipulator trajectory within this model, and finally output the computed plan to its actuators.

However, work within manipulation has long been concerned with uncertainty. Early on, researchers realized that, according to Lozano-Perez et al. (1984), "Knowing the object shapes and positions to sufficient accuracy is not enough." However, uncertainty was generally an issue regarding the positioning accuracy of the manipulator relative to the assembly tolerances of a part. Lozano-Perez et al. go on to describe compliant motion strategies for the canonical peg-insertion task given uncertainty about the peg's position. Later work by Mason (1985) and Erdmann (1988) takes this work one step further, proposing a sensorless manipulation approach, where a part in a completely unknown initial pose can be aligned to a known final pose.

Even when uncertainty is considered, the underlying assumption in this work was that a detailed model can sufficiently describe the interaction of the robot with the world. As a consequence, there was often little need for an expensive and failure-prone robot. Because the interaction of a high-impedance manipulator with a known object could be fully simulated or modelled analytically, much of the ensuing work in robot manipulation was done either virtually, theoretically, or with stiff, well-characterized manipulators such as the Puma arm. Relevant, recent work by Taylor (2004) brings the *sense-model-plan-act* approach to manipulation in human environments. Taylor's robot uses a laser scanner to reconstruct a dense, 3D model of household objects, and then grasps an object using this model.

1.2.2 The Behavior-Based Approach

In the late 80's an alternative to *sense-model-plan-act* emerged. Researchers such as Brooks (1986), Agre and Chapman (1987), and Arkin (1989) developed what is now commonly referred to as the behavior-based approach to robotics. In this approach, a robot continuously refers to its sensors instead of a model of the world. The robot's behavior is not necessarily predictable, but is an emergent property of its interaction with a dynamic and changing world. The robot's controller is composed of many simple behaviors operating locally over an incomplete sensory representation of the environment, and the embodiment of the robot in the actual world is an essential components of the controller design. In this approach, the physical robot is no longer irrelevant to the problem, but central. Everyday environments are embraced instead of avoided.

Using behavior-based methods, robots began to do useful work in human environments. The robot Herbert (Connell, 1989) could find and retrieve empty soda cans within an office building. The robot Polly (Horswill, 1993) lead scores of guided tours of the MIT Artificial Intelligence Laboratory. The success of this approach in robot navigation later led to the algorithms controlling the NASA Mars Soujourner and the iRobot Roomba Vacuum robots, among many others.

However, there has been a scarcity of behavior-based approaches to robot manipulation. For now, and the foreseeable future, sensor and actuation technologies will force a robot to perform tasks using uncertain, piecemeal views of its body and the world. Likewise, robot manipulation in human environments will require an assemblage of algorithms and

approaches to deal with this uncertainty. A behavior-based approach is now emerging as an essential part of the puzzle.

1.3 The Robot Domo

Robots are complex machines, involving sensors, actuators, computation, power electronics, control systems, and perceptual algorithms. These components often have interrelated effects on each other. The placement of touch sensors on a hand can determine the type of contact that can be detected. This, in turn, can reduce, or increase, the need for visual detection of contact. The mechanical stiffness of an arm can limit the bandwidth of the control system, dictating the real-time computational horsepower required. As a consequence of the coupling among its many components, a robot is ideally designed as an integrated whole and not as an assemblage of existing modules.

In addition, a robot's design often suggests a particular research path. As an example, a manipulator with high stiffness lends itself to the well travelled agenda of position control in external, Cartesian coordinates. This has the effect of transforming tasks into problems of geometric positioning relative to a 3D visual scene reconstruction. At this point, a *sense-model-plan-act* approach to manipulation is not very far off.

As an alternative to this path, over the last three years we have developed the upper-torso humanoid robot, Domo, shown in Figure 1-2. Domo has 29 active degrees of freedom (DOF), with 9 DOF in the head, 6 DOF in each arm, and 4 DOF in each hand. It has two CCD cameras, a 3 axis gyroscope in the head, and speech synthesis and recognition. The 22 DOF from the neck down use safe, compliant, force controlled actuators. The hands are also covered in a compliant rubber skin. A Linux cluster of 15 Pentium computers implements a behavior-based control architecture, including visual attention and manipulator coordination. Although the robot is not mobile, it can be manually rolled to different locations in order to vary the robot's workspace.

The design of Domo is covered in further detail in Chapter 4. We would like Domo to exhibit the qualities of both a *partner robot* and a *creature robot*. In the next sections we expand on these qualities.

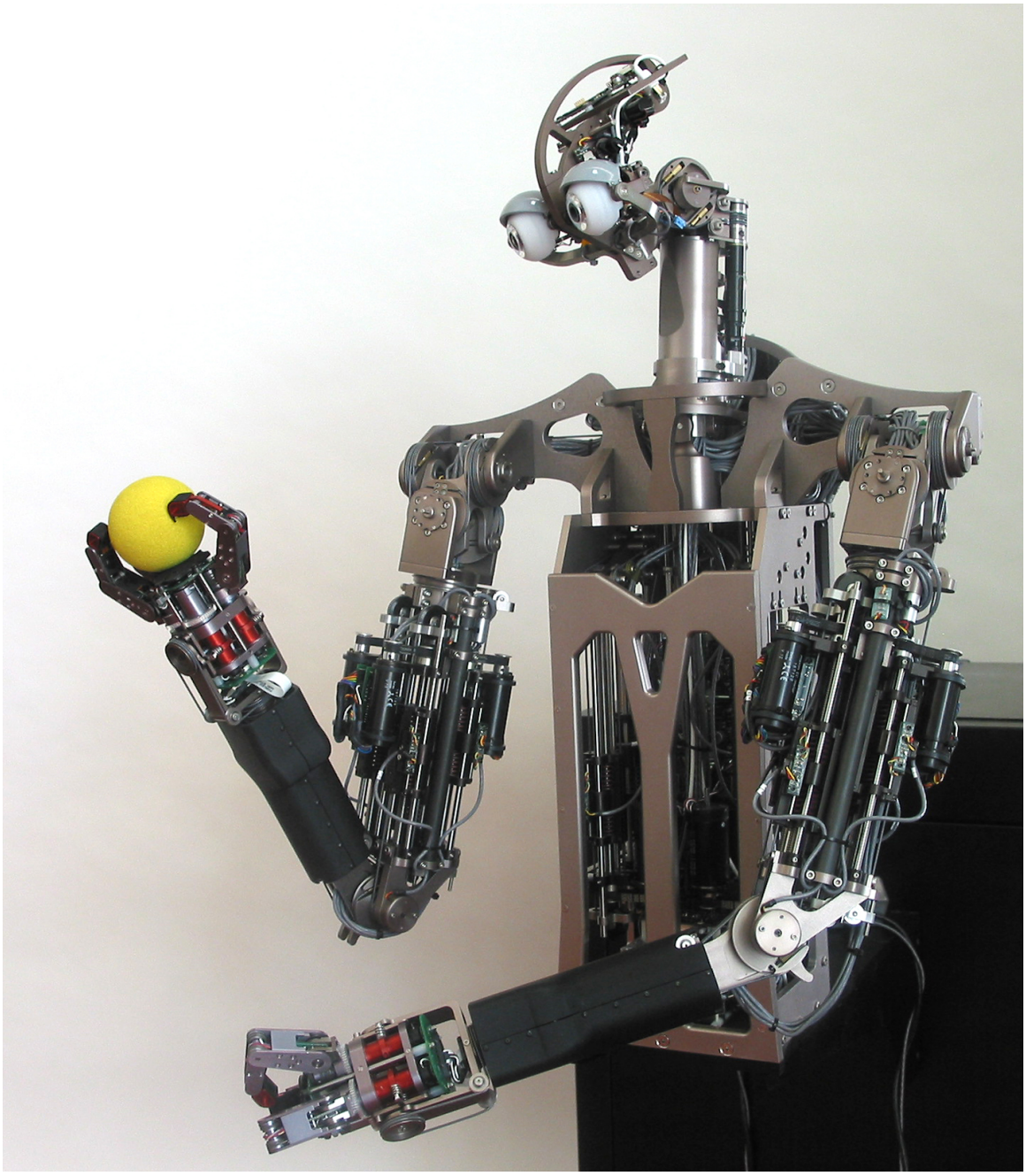


Figure 1-2: Domo, the humanoid robot developed by the author for this work.

1.3.1 Partner Robots

By our definition, a partner robot is a machine of roughly human form designed to work with and assist people. A partner robot is intrinsically safe for human interaction, allowing people to communicate with it using physical cues. For example, if it is performing a task incorrectly, a person could grab its arm and correct its action. The robot's body posture, eye-gaze, and gestures are intuitively understood by people because they immediately map to human social cues. An infant learns early on that pointing combined with crying can cause a caregiver to respond by handing an out of reach object (Vygotsky, 1962). Likewise, if the robot looks at and reaches for an out of reach object, a person would intuitively understand that it wishes to be handed the object. Finally, because a partner robot has a roughly human form, it is able to navigate and work in human environments. It can reach a shelf in a closet, fit through a doorway, grasp a screwdriver, and even safely shake hands with a person.

Partner robots offer several advantages. They do not require specialized tools or fixturing but can use the objects typically found in human environments. People find greater ease in working with a robot that can understand and respond to instruction in a way that is intuitive for them (Breazeal et al., 2004). The robot, in turn, can leverage the advanced perceptual, motor, and planning abilities of a person for assistance in portions of a task that are beyond its ability. The physical and social interface of the partner robot brings a person in to the loop, allowing the robot and person to collaborate in achieving a common goal.

1.3.2 Creature Robots

Many research robots are limited in their autonomy and integration. They often require human intervention, teleoperation during and between tasks, or only run long enough to collect a desired experimental result. Cheng et al. (2001) summarizes the architectures of most robots as either being organized around specific tasks, switching between distinct modes, a compendium of sub-systems, or richly integrated. The richly integrated perspective describes what Brooks (1990) calls the creature approach. A creature robot is an autonomous robot that operates over a long period of time in a dynamic world that has not been specifically engineered for the robot. It utilizes a behavior-based control system that is tightly

coupled to the structure found within its environment. A creature robot, by definition, exhibits coherent observable behavior, responds to salient stimuli, and adequately achieves some long-term task specification.

For manipulation in human environments, a creature robot should be able to smoothly transition between tasks, handle contact disturbances gracefully, respond to human cues, recover from task failures, and do work that is ultimately useful to people. As we will show, Domo's richly integrated control architecture allows it to exhibit many of the creature robot qualities.

1.4 Manipulation in Human Environments: Our Approach

Our goal is to enable robots to accomplish useful tasks through the manipulation of everyday objects. Because of the characteristics of human environments, we avoid methods that rely on precise knowledge about the state of the world. Instead, we are guided by the ideas of embodied artificial intelligence and behavior-based robotics. We have adopted the following tenets:

1. The robot should exhibit the creature robot qualities of coherent behavior, responsiveness to the environment, and adequacy in task execution.
2. The robot should exhibit the partner robot qualities of intrinsic safety for physical interaction, humanoid morphology, and social cueing.
3. The robot should be able to work in the world without significant modification.
4. The robot should be able to work with the everyday objects found in human environments.
5. The robot should accomplish useful tasks.

As much as possible, these tenets have served to guide our design choices and research goals. Out of this exploration, we have arrived at three themes which characterize our approach to manipulation in human environments.

1. The first theme, *let the body do the thinking*, encompasses several ways in which a robot can use its body to simplify manipulation tasks.

2. The second theme, *cooperative manipulation*, refers to the advantages that can be gained by having the robot work with a person to accomplish a task as a team.

3. The third theme, *task relevant features*, emphasizes the benefits of carefully selecting the aspects of the world that are to be perceived and acted upon during a manipulation task.

These themes address many of the unique challenges of human environments. We elaborate on them in the next chapter and describe their application in the remainder of this thesis.

1.5 Contributions

This thesis advances our understanding of how to build robots that can work with people to accomplish useful work. The significant contributions include:

Real robot Our work has been implemented and tested on real hardware. We have designed and built Domo, a full upper-torso humanoid robot using compliant, force controlled actuators. Domo's safe design allows it to exhibit many of the desired qualities of a partner robot. In addition, we have created a behavior-based system called SLATE that allows Domo to exhibit creature robot qualities such as coherent behavior and responsiveness to the environment.

Real tasks Domo can assist a person in useful tasks using unmodeled, everyday objects. The tasks are accomplished in a naturally lit office environment containing all of the uncertainty, dynamics and clutter that one would expect. These abilities are integrated into a single, real-time behavior system, allowing for responsive interaction with a person during task execution. We demonstrate the capabilities of our approach in an unscripted scenario where the robot:

1. Assists in putting items way by:
   (a) taking objects from a person
   (b) passing the objects between its hands
   (c) placing the objects on a shelf
2. Assists in preparing a drink by:

(a) taking a cup and bottle from a person,

(b) inserting the bottle into the cup

(c) taking a spoon from a person and stirring the cup

3. Assists in cleaning up by:

(a) grasping a box between two hands

(b) positioning the box near a person as they place items in it

Three themes for design We present a general strategy for manipulation in human environments. It is summarized by the three themes: *let the body do the thinking, cooperative manipulation*, and *task relevant features*. We show how useful tasks can be accomplished without explicit models of the world or strong environmental constraints. This strategy is an important piece of a bigger puzzle that will undoubtedly involve planning, statistical models of uncertainty, and task learning, among other things.

1.6 Roadmap

This thesis is organized around a generic algorithm for manual skills which integrates the aforementioned design themes. This algorithm is presented in Section 5.3.3 and illustrated in Figure 5-5. We demonstrate the algorithm, and each theme, using modules implemented in Domo's behavior-based control architecture. The organization of these modules is shown in Figure1-3, and they are are summarized in Appendix A. The remainder of the document is outlined as follows:

- Chapter 2: Overview of our three themes for design.
- Chapter 3: Recent work in robot manipulation in human environments.
- Chapter 4: The design of the robot's body.
- Chapter 5: The robot's control architecture.
- Chapter 6: The robot's visual attention system.
- Chapter 7: Modules and experiments illustrating the theme of *let the body do the thinking.*

- Chapter 8: Modules and experiments illustrating the theme of *cooperative manipulation*.
- Chapter 9: Modules and experiments illustrating the theme of *task relevant features*.
- Chapter 10: Integration of the modules into broader tasks.
- Chapter 11 Discussion and future work.

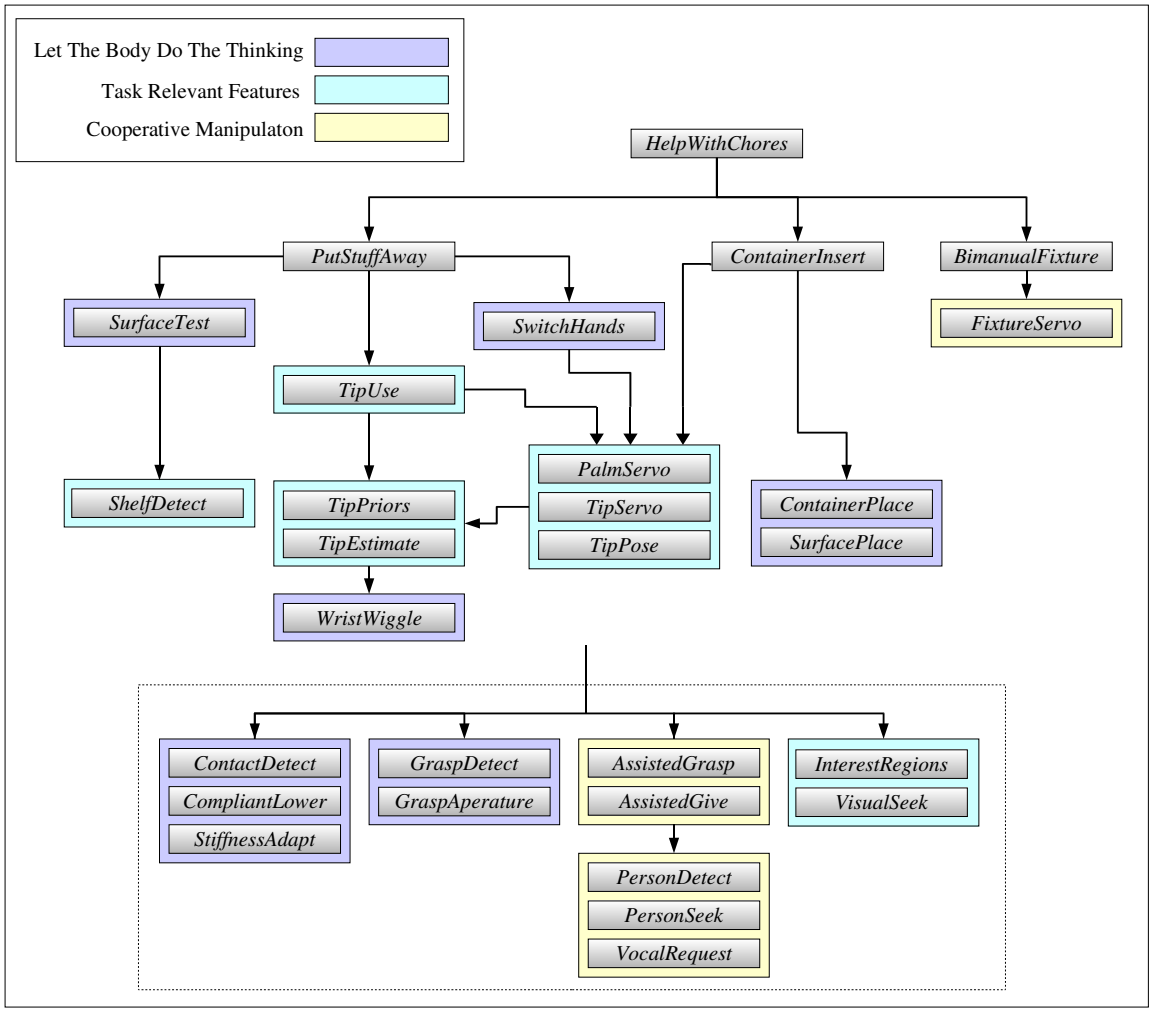


Figure 1-3: The thesis is organized around our three design themes. Each theme is demonstrated using modules (*italics*) from Domo's behavior-based control architecture. For clarity, related modules are grouped and colored according to the strategy they best illustrate. Modules can be composed from simpler modules, and arrows show dependence. The modules in the dotted box provide basic motor and perceptual processing to all of the higher-level modules. Combined, the modules allow Domo to accomplish the *HelpWithChores* task. Appendix A provides a brief summary of all of the modules.

CHAPTER 2

Three Themes for Design

In this chapter we expand on the three themes which characterize our approach to manipulation in human environments.

2.1 Let the Body do the Thinking

This theme bundles together a number of design strategies that make use of the robot's body to simplify manipulation.

2.1.1 Human Form

As we have discussed, human environments are well matched to the human body and human behavior. The width of a door, the height of a cabinet, and the handle on a refrigerator are all designed with people in mind. Manipulation tasks can often be simplified by taking advantage of these built-for-human characteristics. A robot with a camera at eye-level can look down on tables, counter tops, and shelves where many domestic items are placed. A gripper with a human range of grasp sizes can work with many of the same tools and objects that people do. If the robot's footprint is similar to a person's, it can navigate the tight passages of a cluttered room. And if the workspace of a robot's manipulators is comparable to a human's, then the robot can work on an assembly line or in putting away dishes while

keeping its lower body stationary, as people do.

Of course, a robot doesn't require a humanoid appearance to work in human environments. For example, the Cardea mobile manipulator matched the physical properties deemed important to human environments, but did not have a humanoid form. This robot, for which we designed the manipulator, could navigate office environments and push open doors (Brooks et al., 2004). However, humanoids such as Domo have advantages over robots such as Cardea. Domo's form allows it to intuitively cue the person with whom it is working. When reaching for an out-of-reach object, Domo's eye gaze, arm gesture, and open hand will have a similar appearance to a human requesting an object. This more effectively communicates Domo's request than if it had a wholly alien body.

2.1.2 Designing for Uncertainty

Within the domain of robot navigation, researchers have addressed perceptual uncertainty by explicitly modeling it using statistical methods (Thrun et al., 2005), typically with respect to state representations in the form of 3D maps and the robot's pose. This has allowed robots to localize their pose within unmodeled environments and traverse unstructured terrains. Related statistical methods will almost certainly play a role in addressing the challenges of manipulation in human environments, but they are only one piece to the puzzle.

Often, uncertainty in a robot's environment can be reduced, or ignored, through clever design of the body. Research on robot locomotion by Raibert (1986), Cham et al. (2002), and Tedrake (2004), among others, has convincingly demonstrated the benefits of exploiting compliance and natural dynamics for robot control when the robot is in contact with an unmodeled world. Moreover, the work of Williamson (1998a) on exploiting the natural dynamics of a task has demonstrated that similar strategies can be successfully applied to robot manipulation. A common feature among these robots is that the natural dynamics of the body are exploited instead of overridden, and mechanical compliance is embraced instead of avoided.

Compliance can allow a manipulator to adapt to positioning errors resulting from perceptual uncertainty, to limit restoration forces caused by these errors, and to improve contact stability. In manipulation, compliance is often achieved through active impedance control, where the controller maintains a desired force-velocity relationship at the end-effector via velocity or force sensors. The impedance control of manipulators and hands has been widely

studied (Salisbury, 1980; Cutkosky and Kao, 1989). Unfortunately, fundamental limitations on the bandwidth of existing actuator technologies prevent impedance controlled manipulators from effectively responding to unexpected contact with the world (Lawrence, 1989). An alternative, which we adopt with Domo, is to incorporate passive compliance into the body.

Passive compliance is exploited in the wrist of many of today's manufacturing robots. The Remote Center of Compliance (RCC) wrist incorporates elastic elements that allow for small amounts of position uncertainty during a contact task (Whitney and Nevins, 1979). Recent work by Dollar and Howe (2005) optimized the design parameters of a robot hand so that it could better grasp objects with uncertain positions. The hand is made entirely out of soft, urethane materials of varying stiffness. It has embedded tactile and position sensors, and is actuated by remote motors through tendons. The hand's compliance, combined with its optimized design, allows it to robustly form power grasps on a variety of objects. Remarkably, the hand is also robust to sustained impacts from a hammer.

When designing a robot's body with uncertainty in mind, we can trade off perceptual computation for physical computation. This tradeoff is central to Pfeifer's notion of morphological computation (Pfeifer and Iida, 2005). Morphological computation is characterized as performing a "task distribution" between the robot's controller, body, and environment. This distribution is designed through clever use of sensor placement, material properties, body kinematics, and matching of the robot's body to its environment. For example, Iida and Pfeifer (2006) present a planar biped that uses a simple feedforward sinusoidal controller to achieve locomotion. Elastic, polyarticulate tendons spanning across the hip and knee provide the robot with a human-like gait and compliant contact with the unsensed ground. The gait dynamics, in turn, are sensed to provide information about the world (e.g., friction).

2.1.3 Taking Action

A significant advantage of embodied creatures is that they can take actions in the world. Their actions can serve to test a perceptual hypothesis, gain a better view on an item of interest, or increase the salience of a sensory signal. For example, a person will tilt their head in order to better hear a speaker. When a robot moves its sensors in order to resolve ambiguity about a feature of interest, it is engaged in active perception.

Research in active perception is generally tied to active vision and control of eye movement to assist vision processes. Eye movement is a particularly informative area because large amounts of human data are available, and the motor behaviors are simple enough to allow analysis of the full sensorimotor loop. In humans, involuntary eye-jitter has been shown to assist visual discrimination (Rucci and Desbordes, 2003). In robots, (Santini and Rucci, 2006) have shown that motion parallax cues, induced by eye motion, can be used to acquire visual depth information. A notable example involving manipulation is the work of Fitzpatrick et al. (2003), where a robot poked at objects with its manipulator in order to obtain a visual segmentation. We discuss this work further in the next chapter.

One form of active perception is a *compensatory action*, a term of our invention. Compensatory actions are actions that compensate for the robot's physical or perceptual limitations. They are designed to increases the robot's ability to sense and control important aspects of a task. For example, when a person writes with a pencil, they usually rest their hand on the table and keep the pencil angled to the side. This simple action stabilizes the hand from arm-induced disturbances, increasing the person's certainty about the pencil's pose and allowing for precision control. By holding it at an angle, the person increases the visibility of the tip for visual control, and increases their sensitivity to lateral forces felt at the pencil tip.

A robot can be designed with a suite of compensatory actions. These behaviors are not directly involved in solving the task, but critically precondition the robot's body and sensors such that task success is more likely. Some examples of human compensatory actions include:

- Using arms as feelers when walking down a dark hallway.
- Stiffening of the arm before inserting a key into a lock.
- Resting a coffee cup on a table before pouring coffee.
- Bracing a hand on a table while writing.
- Increasing the distance between the thumb and forefinger when grasping in the dark.
- Supporting an awkward grasp with a second hand.

People constantly take actions such as these. This suggests that robot tasks in human environments can be assembled piecemeal using many compensatory strategies. Compensatory

actions often involve what Erdmann and Mason (1988) have called "sensorless manipulation." In sensorless manipulation, a robot uses perceptually blind motion strategies to eliminate uncertainty. For example, the authors demonstrated that a binder clip resting on a tray can be positioned and oriented unambiguously through careful tilting of the tray. This also relates to a large body of work concerning peg insertion tasks under uncertainty, where peg orientation strategies can be used to achieve insertion (Inoue, 1979).

2.2 Cooperative Manipulation

Cooperative manipulation emphasizes the advantages that can be gained by having a robot work with a person to accomplish a task. For at least the near term, robots in human environments will be dependent on people. Fortunately, people also tend to be present within human environments. If we take cooperative manipulation as a design strategy, to what extent can we include the actions of a collaborator into the task design? Can we design a robot's behavior in a way that people will intuitively understand how to assist it? Are there common physical and social cues that a collaborator will generate in order to guide the robot?

Certainly, if the robot's usefulness outweighs the efforts required to help it, people will be motivated to assume the role of a collaborator. For example, the initial version of the Roomba vacuum cleaning robot relies on a person to occasionally prepare the environment, rescue it when it is stuck, and direct it to spots for cleaning and power. The robot and the person effectively vacuum the floor as a team, although the person's involvement is reduced to a few infrequent tasks that are beyond the capabilities of the robot. By treating tasks as a cooperative process, people and robots can perform tasks that neither one could perform as an individual.

As an example, imagine a mechanic working underneath a car. She reaches out for a wrench but can't quite get it. As she reaches about for the wrench, an assistant picks it up and hands it to her. At this moment, the utility of the mechanic's reaching action has been increased beyond her physical limitations. There are several subtle aspects to this cooperative effort that should be familiar to anyone who has ever worked on a car with a friend.

1. The assistant recognizes that the mechanic is physically and perceptually constrained

because she is lying under the car. Consequently, he adapts his actions in order to compensate for her limitations.

2. As the mechanic reaches for the wrench, the assistant naturally adapts the trajectory of his delivery to match hers such that their hands meet at the correct point. A coupled feedback loop is formed between his motor actions and her motor actions.

3. The assistant offers the mechanic the wrench such that her hand naturally grasps the handle in an appropriate manner, knowing that she will have difficulty reorienting it in her hand otherwise.

4. If her view is obstructed, the assistant may push the wrench into her hand with a deliberate force to signal that he is ready for her to grasp it.

In this example, cooperation occurs not just in the physical transport of the wrench. Secondary actions are employed that compensate for the mechanic's physical and perceptual constraints. This is just one example of many such cooperative mechanisms. People will guide someones arm if they are doing a task wrong, provide guiding disturbance forces to a commonly held object during transport or assembly, or hold an object steady while someone works on it.

Cooperative manipulation is as much about understanding how a person will respond to a robot as it is about programming the robot. Consequently, work in human-robot interaction (Burke et al., 2004), socially interactive robotics (Fong et al., 2002; Breazeal, 2004), and assistive robotics (Pineau et al., 2003) is particularly relevant. Although these overlapping areas of research do not typically involve manual tasks, they do emphasize putting people "in the loop" in a way that is intuitive and comfortable for the person. In the next sections we discuss work in assistive robotics and the use of collaborative cues within social robotics.

2.2.1 Assistive Robotics

Assistive robots aim to provide physical or instructional assistance to people in need, such as the elderly, hospital patients, and people with physical disabilities. This is a rapidly expanding area of robotics research due largely to worldwide increases in the elderly population combined with shortages of qualified nurses. For example, within the next five years,

the percentage of the Japanese population over the age of 65 will increase to 25% while the number of available nurses will decline (Ogawa, 2003). This is a worldwide trend, and as workforces age, industrialized countries will face significant labors shortages for which robotics will be one part of the solution.

The field can be partitioned into two categories: hands-on and hands-off (Kyong et al., 2005). The hands-off approach emphasizes social interaction (Tapus and Mataric, 2006; Pollack et al., 2002) for therapeutic aid while the hands-on approach emphasizes physical interaction. The hands-on approach is most relevant for our work. A recent survey on assistive robotics by Haigh and Yanco (2002) shows that manipulation assistance is but one topic in a broad field involving human factors, autonomous wheelchair navigation, cognitive assistance devices, physical therapy, and smart homes. In a survey of 200 potential users of manipulation assistance technology (Stanger et al., 1994), respondents replied that most useful tasks include preparing food, operating electronics devices, picking items up from the floor, and placing items on a shelf. Interestingly, hobby activities such as gardening were also rated highly. These activities all require mobile assistive manipulation. Consequently, work being done with wheelchair mounted manipulators is particularly important.

Wheelchair manipulators provide an incremental path from human teleoperation to autonomous manipulation in human environments. Currently, these manipulators are controlled by a person using a joystick or keypad. For example, the Manus arm is a popular, commercially available manipulator that can pour a drink, put a dish in a microwave, feed food to a person, and open a doorknob (Rosier et al., 1991; ExactDynamics, 2006). Human factors and manipulator design are generally emphasized in this research area (Hillman et al., 1999). However, teleoperation of these manipulators places a significant cognitive load on their users. Although autonomy is an active area of research, there hasn't yet been a strong emphasize on perception in human environments. For example, the KARES robot incorporates force sensing and vision to achieve autonomous tasks such as picking up a pen or a cup (Song et al., 1998). However, color markers and templates are required to sense these objects while the dynamics of everyday environments are not considered.

Contact between a robot and a person is common during physical rehabilitation and workplace assistance tasks. Safe, intelligent control of the interaction forces is therefore critical. Contact can be lethal if the large inertial mass of a typical manipulator is moving with sufficient velocity. We consider the design of safe manipulators in Section 4.6. One

notable example in this area is the cobot. Cobots (Collaborative-Robots) are passive robot devices intended for direct physical interaction with people (Worsnopp et al., 2006; Pan et al., 2005). A cobot and a person collaborate to produce control of the end-effector. The person supplies motive forces and moments, while the cobot constrains the motion. Constraint surfaces can be programmed to assist a human in manipulating a load from one configuration to another, or to assist in physical rehabilitation of stroke patients. Safety is a critical concern for hands-on assistive robotics.

Recently, a third category of assistive robots has been emerging: autonomous partner robots. As described previously, partner robots provide both social companionship and physical assistance. The ARMAR robot at the University of Karlsruhe is a notable example (Zöllner et al., 2004). ARMAR operates in an everyday kitchen environment specially constructed for the robot. It has demonstrated the bimanual unscrewing of a jar lid based on human demonstration and the ability to recognize and respond to human gestures during non-manual collaboration in the kitchen (Stiefelhagen et al., 2004). The highly integrated Hermes robot can navigate dynamic environments and perform scripted manipulation tasks such as fetching a glass of water (Bischoff and Graefe, 2005). At AIST, Sian et al. (2006) have pursued a combination of teleoperation and behavioral autonomy for the bipedal HRP-2 humanoid. The robot manages low-level control and perception under a user's guidance, allowing the robot to execute complex domestic tasks such as fetching a carton from a refrigerator .

2.2.2 Collaborative Cues

During a manipulation task, a robot can use social and physical cues to direct its collaborator. Likewise, the collaborator can use these same cues to supervise the robot. As noted by Breazeal et al. (2004), a true collaboration requires that the robot and person have a mutual understanding of each others' beliefs, intentions, and desires. This shared understanding can be established in part through contextually appropriate cues. If the robot looks to the collaborator, this can cue them to take their turn at a task (Kikuchi et al., 1998). If the collaborator points to an object, this can cue the robot to pick it up (Littman et al., 1996). However, we are still a long ways from robots that can form a nuanced understanding of a person's internal state. Breazeal maintains that without this understanding, the robot acts as a tool and not a partner during a collaborative task. However, even without this rich

understanding, a robot can use rudimentary cues to induce a person to assist it.

One of the earliest examples of this is the demonstration of turn-taking and social amplification with the robot Kismet (Breazeal et al., 2001; Breazeal, 2000). Kismet would engage in dialogue based turn-taking with a naive subject by vocalizing a phrase and then pausing until it had detected the subject's reply. The subject intuitively understood the pause as a cue that the robot was relinquishing its turn in the interaction. Turn-taking simplified the robot's auditory perception because it restrained the person from talking over the robot. The term *social amplification* describes the use of a social cue to amplify the effect of a motor act. With Kismet, if a subject was too far away for the visual attention system to perform well, the robot would crane its head forward. This had the effect of bringing the camera closer in. It also has a greater, secondary effect of beckoning the person to come closer. With this simple act, Kismet could regulate the subject's distance from the camera, improving its ability to detect and track faces. Social cues such as these, as well as affective displays, have been widely investigated within the social robotics community (Fong et al., 2002).

In addition, perception of deictic, e.g. pointing and gazing, cues have been used in several robots to allow a person to direct the attention of the robot. Calinon demonstrated a system where a robot and person used eye gaze, pointing, and head nodding cues to switch context during a collaborative game (Calinon and Billard, 2006). The Infanoid robot of Kozima and Yano (2001) captures the direction of a caregiver's face and gaze in order create a shared attentional focus. Nagai (2005) has shown that subtle differences between the motion of reaching and pointing gestures can be learned by a robot, allowing it to differentiate the intent of such actions. And Breazeal et al. (2004) demonstrate a collaborative task with the Leonardo robot that uses an overhead stereo system to recognize deictic gestures. However, the task is the simple, coordinated pressing of buttons. Particularly relevant to cooperative manipulation is work done at NASA on Robonaut. Robonaut used human pointing cues to determine where to apply a power drill during a lug-nut tightening task (Bluethmann et al., 2004). Related work on the Dexter humanoid accomplishes a similar task where the intention of a teleoperator is inferred from the motion of the manipulator (Rosenstein et al., 2005). Unfortunately, the perceptual environments of these robots are typically constrained and not representative of realistic human environments.

Physical interaction cues have been explored to a lesser extent for cooperative tasks.

Figure 2-1: The *Coffeepot for Masochists* by artist Jacques Carelman (Norman, 1990). Many objects in human environments have been designed to match our physical and cognitive abilities. The design of the coffeepot, for example, has evolved such that inertia of the pot and coffee is well controlled from the handle, the handle is matched to a human-scale power grasp, and the spout is positioned to accommodate visual observation during the pour. (Personal collection of D. A. Norman. Photograph by Norman. Reproduced with permission).

One commonly explored example is the collaborative carrying of an object. On the HRP-2 platform, the robot and a person carry a sheet of plywood together. The person can apply forces to the plywood, which are sensed in the robot's impedance controlled arms, in order to direct the robot as it walks (Yokohama et al., 2003). Other work (Yigit et al., 2003b,a; Waarsing et al., 2001) has also demonstrated cooperative holding of an object but with lesser degrees of integration.

2.3 Task Relevant Features

Human environments have been designed to match peoples' physical and mental abilities, reducing the cognitive load required to function within them. People design in common structural features, or *task relevant features*, so that they can manage in a world of thousands of objects they have never encountered before. The theme of *task relevant features* emphasizes the benefits of carefully selecting the aspects of the world that are to be perceived and acted upon during a manipulation task.

Rather than pursue the problem of general perception within unstructured environments,

we use specialized perceptual detectors for common features that are important to a task. Our expectation is that these detectors will be more robust because of their specialization, and more general because they detect features that are representative of the task, not the object.

Donald Norman's book *The Design of Everyday Things* (Norman, 1990) exemplifies this approach. Norman describes how common features among everyday objects are matched to peoples' bodies and cognitive abilities. A person can pick up and pour coffee from a pot even though they've never seen that particular pot. The physical appearance of the handle and spout may be unfamiliar, but functionally these features are identical to other coffeepots that the person has used. Also, the handle is shaped to match the shape of human hands. Objects that violate expectations of design and function can be exceedingly frustrating for people to use. For example, the *Coffeepot for Masochists*, shown in Figure 2-1 with its spout pointing in the wrong direction, seems absurd because it is poorly matched to a human body.

By specializing perception and control for task relevant features, we can simplify robot manipulation in human environments. In the next sections we discuss how this is possible.

2.3.1 Background

The idea of task relevant perception and control has its roots in the early work of Agre and Chapman (1987) with the Pengi video game agent. Pengi exhibited seemingly complex behavior through a set of simple actions taken in a complex world. Pengi's representation of the world was specified using *indexical functional aspects*. *Aspect* indicates that only a partial representation of the world is required–knowledge of the global state is not assumed. *Indexical* indicates that the world is defined relative to the agent and its motivations. Particular objects are interchangeable and their representation dependent on the agent's context. For example, "the screwdriver nearest me" defines a relative relationship, not a particular object or location in the world. *Functional* emphasizes that things in the world are defined by how they are used rather than descriptively. In the subsequent work of Brooks (1999), perception is directly coupled to action in the form of modular behaviors that eschew complex intermediate representations. Rather than attempting to reconstruct the world in its entirety, these behaviors focus the robot's sensory resources on elements of the world that are important to the current task.

Many researchers treat robot manipulation as a planning problem performed with respect to the global state of the world (Simeon et al., 2004; Saha and Isto, 2006; Taylor, 2004). In contrast, a task relevant approach is aligned with the work of researchers who make use of carefully chosen aspects of the world's state. Jagersand and Nelson (1995) have demonstrated that many tasks can be visually planned and executed using sparse fiducial markers placed on an object's task relevant control points. Piater and Grupen (2002) have shown that task relevant visual features can be learned to assist with grasp preshaping. The work was conducted largely in simulation using planar objects, such as a square and triangle. Platt et al. (2004) have shown that complex dexterous manipulation behaviors can be composed in terms of simple, hierarchical wrench-closure features. Pollard and Hodgins (2002) have used visual estimates of an object's center of mass and point of contact with a table as task relevant features for object tumbling. While these features allowed a robot to generalize learning across objects, the perception of these features required complex fiducial markers. Natale et al. (2004) have demonstrated that a robot can learn to identify objects using only sparse tactile and proprioceptive features.

Task relevant features also have a strong relation to Gibson's notion of an object affordance (Gibson, 1977, 1979). An affordance is a relationship between an object and its environment, defined by how the object could possibly be used. For example, a chair affords sitting while a table affords support. Affordances have been explored within robotics to develop action-oriented perception (Rome et al., 2006).

Human use of task relevant features has been explored experimentally and theoretically. In particular, Todorov and Jordan (2002) suggest that motor coordination may be optimized in terms of the task objectives rather than detailed motor trajectories.

2.3.2 Perceptual Robustness

Robust perception is one of the most significant challenges for robots in human environments. For example, consider the images in Figure 2-2 that show typical views from Domo's camera as it completes a task. In Domo's office environment, the background is cluttered with books and papers, making objects grasped in the robot's hand difficult to distinguish. A person interacting with the robot generates unexpected dynamics. In addition, the lighting changes drastically throughout the day. Humans exhibit incredible adaptivity to this type of variability. In fact, a person teleoperating NASA's Robonaut can accomplish human-level

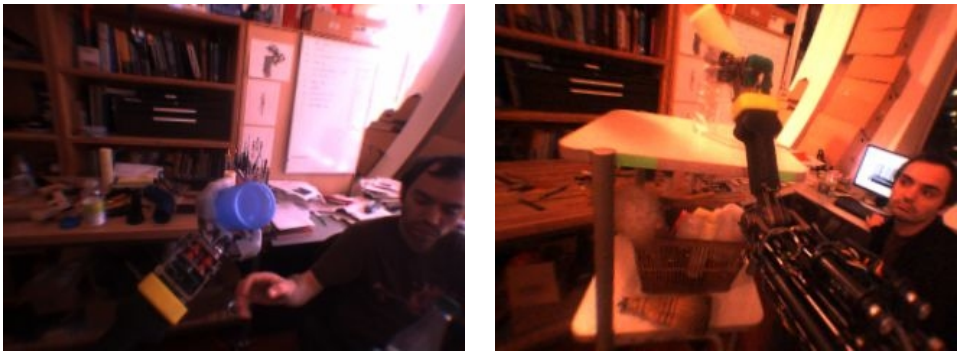


Figure 2-2: Typical views from Domo's camera of its office environment. Everyday environments can be perceptually very difficult for robots. The background is cluttered with books and papers, making objects grasped in the robot's hand difficult to distinguish. A person interacting with the robot, generates unexpected dynamics in the environment. Also, the lighting varies widely throughout the day. (Left) During the day, the natural light in the top corner saturates the scene. (Right) At night, the color hue and contrast shift substantially.

manipulation tasks if they act as the robot's perceptual system (Glassmire et al., 2004). Outside of teleoperation, robot manipulation is typically limited to controllable laboratory and factory environments, or specialized skills. Fitzpatrick (2003) has described two general approaches to perception in real, everyday environments:

1. **Robustness from perspective**: Adopting the right frame of reference can simplify a problem.

2. **Robustness from experience**: The robot's experience in the world can be used to improve perceptual models.

The *robustness from perspective* approach includes the careful selection of perceptual features that are relevant to the task and environment. This has been successful in a number of robotics projects. For example, the robot Polly could navigate the dynamic, cluttered hallways of the MIT AI Lab in real-time using relatively little computation (Horswill, 1993). Polly's perception and control was highly specialized for office environments, where obstacle avoidance could be reduced to carpet detection, an obstacle was defined simply as a non-carpet region, and an obstacle's depth was a easily found as a function of its height in the image. Fitzpatrick and Metta (2003) specialized perceptual detectors for the motion

generated at a robot's hand while coming into contact with an object, allowing the robot to obtain clean visual segmentation of objects of interest by poking them. Fitzpatrick also demonstrated robust, uncalibrated reaching over a surface by visually servoing both the end-effector and its cast shadow (Fitzpatrick and Torres-Jara, 2004). Reaching over a surface is a common activity within human environments, and a cast shadow is potentially less affected by variability in an object's appearance.

The success of work in face detection (e.g, Viola and Jones (2004)) has demonstrated the power of developing perceptual algorithms within specialized domains. Our approach of *task relevant features* looks to leverage this robustness in the context of perceptual features that are common to everyday tasks.

2.3.3 Generalization

We can specify a manipulation task in terms of the perception and control of sparse features that represent the task, not the object. Focusing the robot's resources on only those aspects of the world that are important to the task should also allow the same controller to extend to multiple objects. As noted earlier, a coffeepot handle and spout define its use, and perception and control of these sparse features allow us to use unfamiliar coffeepots. However, mastery of the use of a coffeepot handle can also generalize, roughly, to the use of a briefcase handle. Another example is the tip of a tool. The tip of a screwdriver and the tip of a knife are often used identically to pry something open. As we will show in Chapter 9, manipulation of a large set of human tools can be specified in terms of the tool's tip.

In addition, a task relevant approach allows for motor-equivalent control strategies, where a high-level goal can have multiple solutions at the control level. For example, we typically use a pencil by grasping it between our fingers. However, we could still write by grasping the pencil between our forearm and bicep, or even by wrapping our toes around it. This indicates that the planning and control of many tasks very likely occurs in a frame of reference that is invariant to the joint postures and forces. In fact, findings in neural motor control provide evidence that humans plan and execute tasks based in the coordinate frame of the intended motion of an object, not the desired joint trajectory (Cisek, 2005). Also, the motor representation used to plan an action appear to be independent of the effector used. A pincher grasp formed with our fingers has remarkably similar velocity and grasp aperture characteristics to a pincher grasp formed with a hand tool (Gentilucci et al., 2004).

CHAPTER 3

Recent Work

Robot manipulation is a broad research area, and in this chapter we discuss mostly recent work within the field. We refer the reader to excellent surveys and texts on model-based manipulation (Mason, 2001), visual servo control of manipulators (Kragic and Chrisensen, 2002), issues for general purpose manipulation (Grupen et al., 1989), calibration techniques for manipulators (Hollerbach and Wampler, 1996), design approaches for robot hands and graspers (Bicchi, 2000), approaches to model based grasping (Shimoga, 1996), and tactile sensing for manipulation (Tegin and Wikander, 2005).

In the last few years, there has been a resurgence of interest in robot manipulation, particularly for domestic applications, within Europe, Japan, and the US. In a recent whitepaper defining the nascent field of of Autonomous Mobile Manipulation (AMM) (Brock and Grupen, 2005), a large community of researchers outline the potential impact, challenges, and research directions for robots that can perform manual tasks in unstructured environments. The stated target is the development of robots with the visual recognition abilities of a two-year-old child, the manual dexterity of a six-year-old, and the ability to navigate human-scale environments. Much of this current generation of research aims to bring robots out of the labs and factories and into the world. It emphasizes the importance of human environments, the unique challenges that they pose, and the need for alternatives to the *sense-model-plan-act* approach.

In this chapter we discuss in more detail several research projects that illustrate alternatives to the *sense-model-plan-act* approach to manipulation.

3.1 Connell: Behavior-Based Manipulation

The Herbert robot of Connell (1989) represents one of the first autonomous mobile manipulators to use a behavior-based approach. Although not a recent result, this work still represents an important direction for the field. Connell developed a mobile robot with a planar two DOF arm able to locate and retrieve empty soda cans in an everyday, office environment. Remarkably, this was achieved with only simple sensing and a small network of behaviors fashioned after the early subsumption architecture of Brooks (1986). We attribute the success of this work primarily to two sources.

First, the robot's body and controllers were carefully designed to match its world of office buildings and soda cans. The gripper and arm were specialized for reaching to and grasping cylindrical objects on cluttered tables. The gripper was equipped with contact switches and IR break-beam sensors. It relied on a simple kinematics and sensors instead of complex 3D models and inverse kinematics. Connell notes that soda cans are all the same size, rotationally symmetric, and typically found in a vertical orientation on a table-height flat surface. By specializing the robot's reaching and grasping behaviors for these features, Connell leveraged the structure of human environments and the task.

Second, the robot controllers were constructed as simple behaviors that, in Connell's words, "communicate through the world". For example, the manipulator is left in the STOP state when a can is grasped. A RETRACT behavior pulls in the arm whenever it senses the STOP state. A third HOME behavior causes the robot to head back to its base whenever it senses that the arm is retracted. In this way, the state of the robot's body interacting with the world, not the state of an internal model, is used to coordinate behaviors. Connell maintains that directly referencing the state of the body and sensors improves the system's robustness.

A number of researchers have developed behavior-based controllers for manipulators subsequent to Connell's early work. However, few systems have demonstrated the extensive integration of this robot or the ability to perform more than a single, well defined task at a time. In the case of Waarsing et al. (2001), an industrial arm on a mobile base was used

to collaboratively carry an object with a person by sensing the forces in the arm through a load cell. Work by Wasik and Saffiotti (2003) with a small industrial arm demonstrated more complex behaviors for a visual pick-and place task. However, this was done in a very controlled environment using homogeneous blocks. Much of the work on the Dexter robot (Platt et al., 2003a) has behavior-based components, and this project is reviewed shortly.

3.2 Metta and Fitzpatrick: Poking and Affordances

Work on the Cog robot by Metta and Fitzpatrick (2003a) treats manipulation as a means to assist perception under uncertainty. Often, vision alone is not enough to reliably segment distinct objects from the background. Overlapping objects or similarity in appearance to the background can confuse most visual segmentation algorithms. Rather than defining an object in terms of its appearance or shape, the researchers defined an object in terms of coherent motion generated through contact.

Cog poked at objects on a table with its manipulator. It then used the tight correlation between its arm motion and the sensed optic flow to detect both its hand and the boundaries of the object. As a result, Cog could obtain clean visual segmentations of otherwise difficult to segment objects. This a clear demonstration of leveraging the robot's embodiment to deal with the type of uncertainty found in human environments. Although not demonstrated, a robot's perceptual system could employ this type of behavior to actively test perceptually ambiguous scenes.

Subsequent work (Fitzpatrick and Metta, 2003) used this poking behavior to learn about an object's rolling affordance. This a general affordance often explored by infants playing with bottles and toy cars, for example. A car rolls in the direction of its primary axis while a bottle does not. To learn this distinction, Cog poked at an object placed in front of it, using one of four directions. It then observed the motion of the segmented object relative to its primary axis. Subsequently, it could learn the correct direction to poke an object to cause it to roll. Although described as a "proof of concept," this work points to an affordance based representation of everyday objects, gained through manipulation, that avoids full blown 3D models.

3.3 Robonaut: Collaborative Tool Use

The Robonaut humanoid (Ambrose et al., 2000) developed at NASA JSC is perhaps the most mechanically sophisticated bimanual manipulation platform in use today. Its primary purpose is as a collaborative robot assistant that can work with astronauts during a space mission. Under teleoperation, the robot has achieved impressive cooperative tasks such as inserting a flexible cable into a receptacle (Glassmire et al., 2004). Recent work has pursued autonomous manipulation with marginal teleoperation. For example, Diftler et al. (2004) present an autonomous system where Robonaut visually detects a person, navigates to the person, takes a tool from the person, and then carries the tool while following them.

In a significant collaboration among many researchers, Bluethmann et al. (2004) describe a system whereby Robonaut reaches for and grasps a power drill, watches as a collaborator points to the lugnuts on a car wheel, determines the desired order for tightening, and inserts the drill on to and tightens each lugnut. This work demonstrates an impressive level of integration, using modules for learning under teleoperation, spatial reasoning and memory, gesture recognition, and tool pose recognition. The team reports that inserting the drill onto the nut is the most challenging aspect of the task as the grasp on the drill varies with each trial. However, the insertion is done open-loop in world coordinates, and not with visual feedback.

The Robonaut tasks are supervised by an operator and specified as a deterministic sequence of behaviors (Aldridge et al., 2000). A layered behavior architecture could potentially expand its autonomy, allowing it to respond to unexpected events during the task and perhaps better recover from failures. Robonaut's perceptual system uses template matching to known objects in order to recover the 3D pose of the object Huber and Baker (2004). This approach is reasonable for a space mission where the tools are presumably known, but not suitable for everyday, domestic environments.

3.4 Dexter: Manipulation Gaits and Grasp Features

Researchers at UMass Amherst are working with a bimanual robot, Dexter, to study cognitive development through manual interaction with the world. The group has taken a largely embodied approach, employing reactive, closed-loop controllers based on haptic feedback to acquire task knowledge, learn object properties, and accomplish bimanual tasks (Platt

et al., 2003a). Here we consider two such projects in further detail: composing tasks using manipulation gaits (Platt et al., 2004) and the learning of visual features to support grasping (Piater and Grupen, 2002).

A manipulation gait moves a grasped object into a desired pose while maintaining stable contact by essentially walking the fingers or hand around the object. This had previously been demonstrated on known, modelled objects, but Huber and Grupen (2002) introduced a reactive strategy for manipulation gaits that can be done without prior models. Their approach sequences a combination of concurrent, closed-loop haptic controllers. Each controller, drawn from a control basis (Platt et al., 2003c), achieves a simple force objective such as wrench closure. Manipulation behaviors are composed by nesting a controller objective, such as a posture, within the null-space of another controller objective, such as contact. In Huber and Grupen (2002), the approach demonstrated rolling of a cup between the robot's fingertips. The strategy was extended to bimanual tasks by treating each hand as a giant fingertip, and the fingertip controllers independently maintained the local contact objectives (Platt et al., 2003b,c).

Their treatment of dexterous manipulation is essentially behavior-based, as the robot continually monitors its haptic state in the world to update simple, reactive controllers. Although the work has been primarily concerned with rich haptic sensing on idealized objects, recent work by (Platt, 2006) demonstrated a grocery bagging task on everyday objects in a controlled environment.

When a person reaches for an object, they preshape their grasp in anticipation of contact with the object. Planning a grasp, prior to contact, is typically achieved through the selection of contact points on a 3D model (Shimoga, 1996). In contrast, an approach developed by Piater and Grupen (2002), as well as Coelho et al. (2000), learns the relevant visual features on an object that afford a stable grasp. Selection of these features allow the grasp to be preshaped, increasing the robustness of their control basis grasp controller. This work is a demonstration of task relevant features, as the object is sensed through sparse, visual features that afford grasping rather than a detailed 3D model. However, the approach was only demonstrated on simple, planar geometric objects. In subsequent work by (Platt, 2006), a 3D ellipsoid representation is used, and the robot learns to orient its grasp to the major axis of the ellipsoid.

3.5 STAIR: Learning of Grasp Points

A number of researchers at Stanford have begun an ambitious new project, named STAIR, aimed at autonomous mobile manipulation within human environments. Their stated goals include tidying up a living room, picking up and throwing away trash, loading the dishwasher, using multiple tools to assemble a bookshelf, as well as guiding guests around a museum or lab. Initial results from the project are encouraging.

Saxena et al. (2006) have developed a learning algorithm that predicts, based on a monocular image, the best position to form a pincer grasp on an object. The supervised learning was achieved offline using synthetic data, but the learned feature detector works on real images. When several views of the object are acquired, the most likely 3D location of the grasp feature is estimated, and the robot can reach to and grasp the object. The algorithm has successfully learned grasped points on a variety of everyday objects, including a pen, screwdriver, book, and duct tape. However, as of yet it hasn't been demonstrated using an everyday, cluttered background. This preliminary work is an excellent example of a specialized, task-relevant feature detector. Although the single-grasp-point representation does not encompass the variety of objects found in human environments, it is applicable to a wide range of objects, particularly objects with handles. The learning algorithm's choice of a single, sparse visual feature enables allows it to work with different, unmodeled objects.

CHAPTER 4

Building Bodies

This chapter describes the design of Domo[1]. As we will see, this is an integral part of our broader design theme to *let the body do the thinking*. Clever design of a robot's body can reduce the complexity of the perception and control required, especially in human environments. Actuators that exhibit passive compliance and force control can adapt to unforeseen disturbances in the world. Physical traits of a robot's body can replace sensing and computation. For example, the Roomba vacuum is a low-profile disc. This allows it to avoid getting snagged in tight spaces and trapped under beds, greatly reducing its need to sense its environment.

Often, a robot's design will influence the research questions that are pursued. Poor reliability and robustness can limit long time-scale experiments. High manipulator momentum can be unsafe for human experiments. Design limitations can also inspire new research approaches. For example, the inexpensive robot design of Horswill (1993) experienced drift in its mechanical components, precluding the possibility of precise camera calibration. This lead to the design of novel visual algorithms for robot navigation that were robust to this type of uncertainty. In other cases, such as with the Series Elastic Actuator (SEA) (Pratt and Williamson, 1995), the imprecision of inexpensive mechanical components can be can-

[1]The mechanical design is greatly indebted to the significant contributions by research engineer Jeff Weber.

celled by using control feedback.

Our robot Domo (shown in Figure 1-2) has 29 active degrees of freedom (DOF), with 9 DOF in the head, 6 DOF in each arm, and 4 DOF in each hand. It also has two CCD cameras, a 3 axis gyroscope in the head, speech synthesis, and speech recognition. The initial design included 24 FSR tactile sensors in the hands[2]. The hands are also covered in a compliant, rubber skin. All 29 DOF provide joint-angle sensing through potentiometers or encoders. The 22 DOF from the neck down use compliant, Series Elastic Actuators, providing the robot with the proprioceptive sense of joint torque. Each appendage has a dedicated embedded controller that handles sensor acquisition and motor control. These controllers are networked together using CANbus and are interfaced to a cluster of Linux PCs.

4.1 Notation

In this section, we briefly review the mathematical notation used to describe the robot . The kinematic structure and significant coordinate frames of Domo are shown in Figure 4-1. In general, a vector is written in lowercase boldface and a matrix in uppercase boldface. We add a superscript to a vector or matrix in order to refer it to a particular coordinate frame. The superscript may be dropped for clarity if it is evident from the context. For example, a vector in coordinate frame $\{A\}$ is specified as

$$\mathbf{p}^A = \begin{bmatrix} p_x \\ p_y \\ p_z \end{bmatrix}.$$

The world frame is often implicit: $\mathbf{p} = \mathbf{p}^W$. A homogeneous transform, describing a rotation and translation, from frame $\{A\}$ to $\{B\}$ is denoted as ${}^B\mathbf{T}^A$. Point $\mathbf{p}^A$ has coordinates $\mathbf{p}^B = {}^B\mathbf{T}^A\,\mathbf{p}^A$ in $\{B\}$, where

[2]It was found that the FSR sensors were not reliable for tactile sensing but increased the hand's complexity. These were eventually removed from the robot.

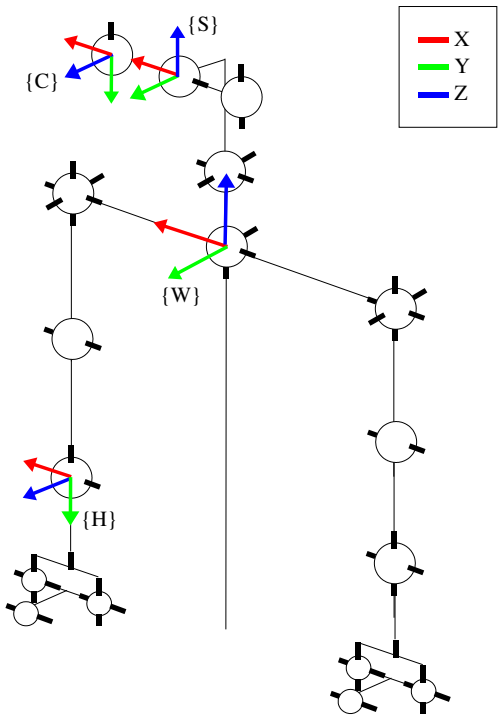


Figure 4-1: The kinematic structure and significant coordinate frames for the robot. $\{W\}$ is the gravity aligned world frame. $\{H\}$ is the hand coordinate frame at the end of the 6 DOF manipulator. $\{C\}$ is the camera frame with its origin at the focal point, and $\{S\}$ is the gravity aligned ego-sphere frame described in Chapter 6. A particular manipulator is delineated by a subscript such as $\{H_{right}\}$ if required.

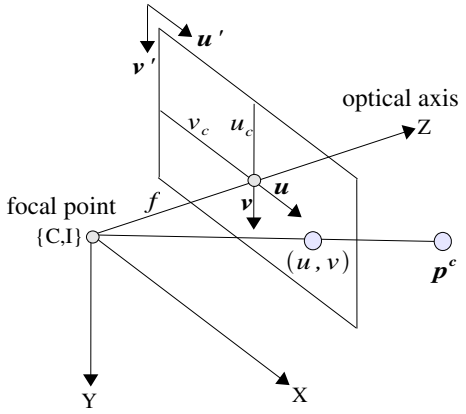


Figure 4-2: Coordinate frames for the calibrated pinhole camera.

$$ {}^{B}\mathbf{T}^{A} = [\,{}^{B}\mathbf{R}^{A} | \mathbf{t}^{B}_{Aorg}] = \begin{bmatrix} r_1 & r_2 & r_3 & t_1 \\ r_4 & r_5 & r_6 & t_2 \\ r_7 & r_8 & r_9 & t_3 \\ 0 & 0 & 0 & 1 \end{bmatrix}. $$

${}^{B}\mathbf{R}^{A}$ is a $3x3$ rotation matrix and $\mathbf{t}^{B}_{Aorg}$ is the origin of frame $\{A\}$ with respect to $\{B\}$. A position vector $\mathbf{p}^{A}$ is $4x1$, though a $3x1$ vector is used when considering directions only. For example,

$$ {}^{B}\mathbf{T}^{A}\mathbf{p}^{A} = {}^{B}\mathbf{T}^{A} \begin{bmatrix} p_x \\ p_y \\ p_z \\ 1 \end{bmatrix}, $$

but also

$$ {}^{B}\mathbf{R}^{A}\mathbf{p}^{A} = {}^{B}\mathbf{R}^{A} \begin{bmatrix} p_x \\ p_y \\ p_z \end{bmatrix}. $$

4.1.1 Pinhole Camera Model

We use a pinhole camera model for the robot's cameras. The model's intrinsic parameters are estimated with the Calibration Toolbox for Matlab and the lens distortion is corrected

in real-time using the undistort method in the Intel OpenCV computer vision library. We use the standard coordinate system for the pinhole camera model as shown in Figure 4-2. A 3D point in the world $\mathbf{p}^W$, with position $\mathbf{p}^C = {}^C\mathbf{T}^W\mathbf{p}^W = [x, y, z]^T$ in camera frame $\{C\}$, projects onto the image plane at

$$\mathbf{k} = \begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} \frac{fx}{z} \\ \frac{fy}{z} \end{bmatrix},$$

for a focal length f . For simplicity, the optical axis intersects the image plane at pixel $[0, 0]$. The true pixel coordinates of the point are relative the upper-left corner at

$$\mathbf{k}' = \begin{bmatrix} u' \\ v' \end{bmatrix} = \begin{bmatrix} u + u_c \\ v + v_c \end{bmatrix},$$

where u_c and v_c are dependent on the camera calibration.

4.2 Compliant Actuators

The 22 actuators in Domo's arms, hands, and neck incorporate linear springs to provide force sensing and passive compliance, as shown in Figure 4-3. These actuators are called Series Elastic Actuators (SEA). The SEA design originates with Pratt and Williamson (1995). The idea is simple. A linear elastic element is placed between the motor and the joint. The elastic deflection is measured using a displacement sensor such as a strain gauge or potentiometer. The force or torque at the actuator can then easily be computed using Hooke's law ($f = -K_s x$, where K_s is the spring constant and x is the spring displacement).

4.2.1 Design

We developed a cable-drive SEA for Domo's arms and a novel, compact configuration of the SEA for Domo's hands. According to Pratt and Williamson (1995), there are several advantages to these actuators:

1. The spring and potentiometer provide a mechanically simple method of force sensing.

2. Force control stability is improved when intermittent contact with hard surfaces is made. This is an important attribute for manipulation in unstructured environments.

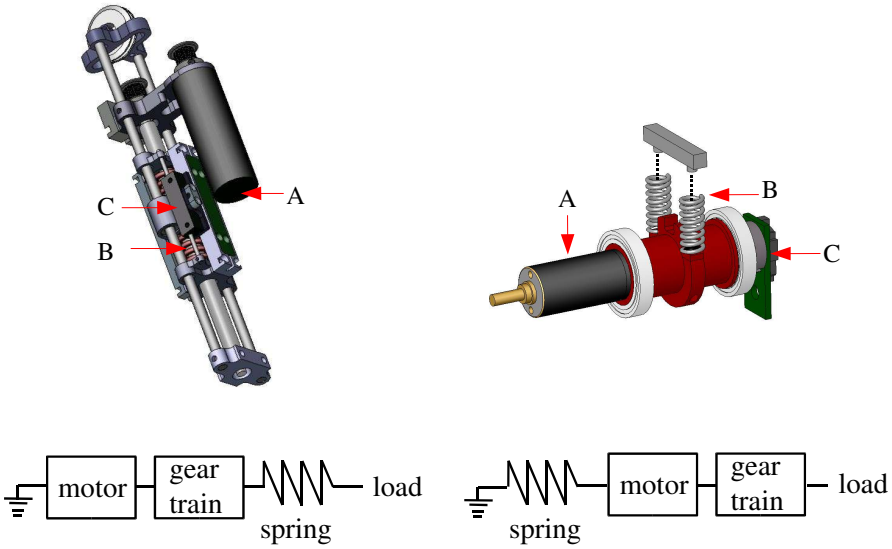


Figure 4-3: Left: the Series Elastic Actuator (SEA) used in the arms. Right: a novel SEA configuration developed for the hands. The traditional, arm SEA places a compliant element between the motor output and the load. The hand SEA places the compliant element between the motor housing and the chassis, allowing for a simpler and more compact design. The arm SEA drives a cable with motor (A) and measures the cable force through the spring deflection (B) with a linear potentiometer (C). For the hand SEA, the motor (A) is on bearings and held captive by the springs (B). The motor deflection is measured by a rotary potentiometer (C). The deflection is linearly related to the output torque for small deflections.

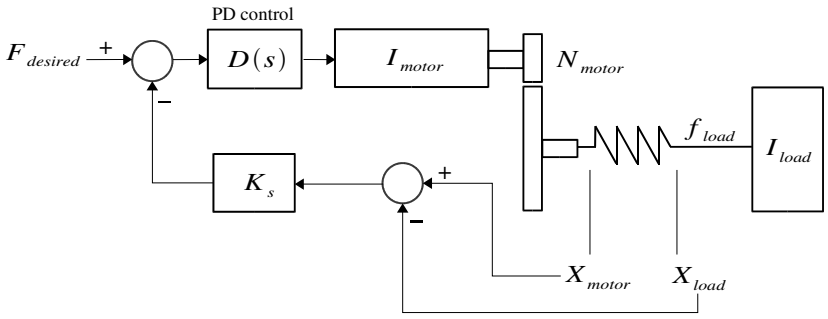
3. Shock tolerance is improved. The use of an N:1 geartrain increases the reflected inertia at the motor output by N^2, causing shock loads to generate high forces on the gear teeth. The passive spring serves as a mechanical filter of these high bandwidth forces, reducing the potential of damage to the gears.

4. The dynamic effects of the motor inertia and geartrain friction can be actively cancelled by closing a control loop around the sensed spring displacement. This allows a highly backdrivable actuator to be constructed out of low-grade components.

5. A traditional force controlled actuator exhibits a large impedance at high frequencies due to insufficient motor response. In contrast, the elastic elements in an SEA act as passive springs, reducing the effective impedance. This allows for intrinsic manipulator safety.

In contrast to traditional SEA designs, Domo's hand SEA places the elastic element between the motor and the chassis. While this method of sensing the actuator torque has recently been used for non-compliant sensing (Kim et al., 2005), this is the first demonstration using an elastic element. As shown in Figure 4-3, the motor is on bearings and held captive by springs. The motor's deflection is measured by a rotary potentiometer. This deflection is linearly related to the torque at the motor output for small deflections. The differences between these two SEA designs provide distinct advantages and disadvantages. The arm SEA uses a linear ballscrew and a cable transmission. The ballscrew provides greater efficiency and shock tolerance than a gearhead. However, this SEA is limited by the travel range of the ballscrew, creating packaging difficulties. The linear potentiometer must move with the motor output, precluding applications requiring continuous rotation. In contrast, the hand SEA can allow continuous rotation at the motor output as the sensor does not move with the motor output. This configuration is also much more compact than the arm SEA as it is mechanically simpler with fewer moving parts.

The springs in the arm actuators have a stiffness of $K_s = 91$kN/m while those in the hand have $K_s = 21$kN/m. For the arms, the precision ballscrews have a pitch of 1-1.5mm and the actuator stall force using Maxon EC20 brushless motors is approximately 800 N. For the hands, the brushed 13mm Maxon motors can output 5.4N at the fingertip.



$F_{desired}$	Commanded force
F_{load}	Force applied to load
X_{motor}	Position of motor
X_{load}	Position of load
K_s	Spring stiffness
I_{motor}	Inertia of motor
I_{load}	Inertia of load
N_{motor}	Motor gear ratio
$D(s)$	PD controller transfer function

Figure 4-4: Control topology for the SEA (Zinn et al., 2002).

4.2.2 Control

The control of an SEA has been well characterized (Robinson, 2000; Pratt and Williamson, 1995; J. Pratt and Pratt, 2001). We review the relevant details here. Controllers of Domo's two SEA configurations are identical, as the series ordering of the spring and motor does not affect the physical transfer function. Figure 4-4 depicts the control topology for the SEA. One advantage of the SEA is that it reduces the output impedance of the actuator across the entire frequency spectrum, making it inherently compliant, and therefore safe, when interacting with the world. In contrast, a stiff actuator will have high output impedance at high frequencies beyond its control bandwidth. It can be shown (Zinn et al., 2002) from Figure 4-4 that the output impedance of the SEA is

$$\frac{F_{load}(s)}{X_{load}(s)} = \frac{s^2 N_{motor}^2 I_{motor}}{\frac{s^2 N_{motor}^2 I_{motor}}{K_s} + 1 + N_{motor} D(s)}. \tag{4.1}$$

We see that for frequencies above the closed loop bandwidth (s is large), the output impedance reduces to the spring stiffness: $\frac{F_{load}(s)}{X_{load}(s)} = K_s$. In contrast for a stiff actuator ($K_s \approx \infty$), the output impedance reduces to $\frac{F_{load}(s)}{X_{load}(s)} = s^2 N_{motor}^2 I_{motor}$, which goes up with the square of the gear reduction.

The inherently low output impedance of an SEA makes it suitable for applications where human safety is important. However, the compliance also reduces actuator performance. The flexible modes of an SEA prevent control bandwidths greater than about one-third of its fundamental resonant frequency. The resonant frequency for cable-driven robots can be 10 Hz or less (Zinn et al., 2004). Fortunately, robot's that work cooperatively with people do not typically need to achieve high-performance ballistic motions during a task. In human environments, the lower control bandwidths of an SEA are a reasonable given the added benefits of safety and compliance. We consider safety issues further in Section 4.6.

4.3 Arms

4.3.1 Arm Design

Domo's arms were designed to be lightweight, safe, compact, efficient and mechanically robust. These are sometimes competing, but often complementary objectives. Lowering the manipulator weight reduces its effective inertia, improving the inherent safety. This also reduces the motor power requirements which can increase efficiency. Placing compliance

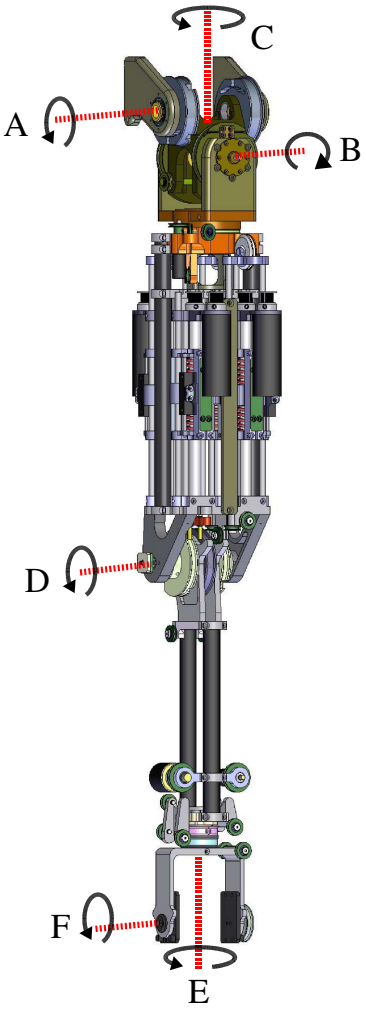


Figure 4-5: The mechanical structure of Domo's arms. A compact cable-drive differential at the shoulder provides pitch (A) and roll (B). These two DOF are driven by actuators placed in the robot torso. The bicep of the arm contains four other actuators: shoulder yaw (C), elbow pitch (D), and wrist roll (E) and pitch (F) driven by two cables routed through the elbow.

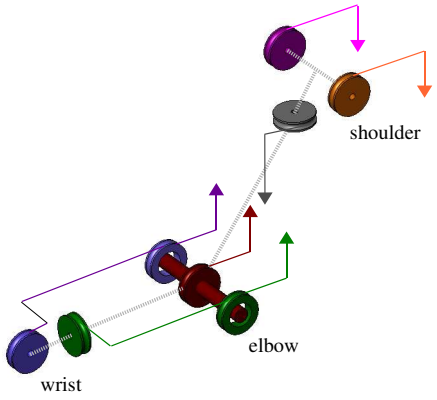


Figure 4-6: Layout of the arm cable-drive. Each SEA (located at the arrow) drives the joint through a pull-pull cable-driven hub. The cable-drive design allows the actuators for the differential shoulder to be placed in the body. The wrist actuators are placed in the forearm and their cable-drives pass through the elbow. This mass distribution lowers the arm inertia, though it also causes coupling between the elbow and wrist. This is compensated for in software.

in the drive train can improve safety and protect the geartrain. The primary innovation in Domo's arms is to combine the compliance provided by SEAs with the weight-reduction given by cable-drive WAM arms (Townsend and Salisbury, 1993). The resulting arm is lightweight (2.1kg), safe for human interaction, reasonably strong (5kg payload), human scale, and efficient (12W during static extension without load, 80W during peak ballistic motion).

The SEA arm design originated with our work on the Cardea robot (Brooks et al., 2004) as well as Williamson's SEA manipulator design for the Cog robot (Williamson, 1998b). As shown in Figure 4-5, a compact cable-drive differential at the shoulder provides pitch and roll. These two DOF are driven by actuators placed in the robot torso. The bicep of the arm contains another four actuators for shoulder yaw, elbow pitch, wrist roll, and wrist pitch. The cable-drive system, shown in Figure 4-11, allows for non-collocation of the actuators (though the joint angle is sensed at the joint). The drive-cables for the wrist actuators are routed through the center of the elbow joint. Placing the wrist actuators in the bicep lowers the arm inertia but causes mechanical coupling between the elbow and wrist.

This is compensated for in software. Unfortunately, a cable-drive manipulator can also be substantially more complex, and friction and elastic effects of the cables can introduce control complexity. Salisbury et al. (1988) consider design issues with cable-drive arms further.

4.3.2 Arm Control

Different methods are used to control the arms depending on the task. These include:

1. **Joint torque**. The DSP implements a PD controller of the form

$$F_{motor} = K_{fp}E_f + K_{fd}\dot{E}_f,$$

where the error E_f is measured with respect to the SEA spring as

$$E_f = F_{desired} - K_s \cdot (X_{motor} - X_{load}). \tag{4.2}$$

This controls the force output of each actuator. A desired joint torque $\tau_{desired}$ is readily converted to $F_{desired}$ using the known mechanics of the cable-drive system.

2. **Gravity compensated joint torque**. Given the manipulator joint angle vector $\mathbf{\Theta} = [\theta_1, \theta_2, \ldots, \theta_6]^T$, the joint torques resulting from gravity, $G(\mathbf{\Theta})$, are computed using the kinematic model and point mass estimates for the forearm and bicep (Gourdeau, 2005). The controller's desired joint torque is modified such that $\tau_{desired} = \tau_{gdesired} - G(\mathbf{\Theta})$, allowing for gravity independent control of the arm as it moves through its workspace.

3. **Joint angle**. The joint angle is controlled on the DSP by a PID loop around the gravity compensated torque controller. This controller is of the form

$$\tau_{desired} = K_{pp}E_p + K_{pd}\dot{E}_f + K_{pi}\int E_p dt - G(\mathbf{\Theta}), \tag{4.3}$$

where $E_p = \theta_{desired} - \theta$. In practice, we use an extended form of the controller without damping:

$$\tau_{desired} = K_{ps}(K_{pp} \cdot B(K_{bp}, E_p) + K_{pi} \cdot B(K_{bi}, \int E_p dt)) - G(\mathbf{\Theta}) + \tau_{bias} \tag{4.4}$$

The function $B(K, E)$ bounds the error E within $\pm K$ to safely limit the commanded torques. The stiffness of each joint can be set by K_{ps}, where $0 \leq K_{ps} \leq 1$ and $K_{ps} = 0$ puts the arm in a zero-gravity mode. In this mode, a controller can command a non-zero bias torque τ_{bias}.

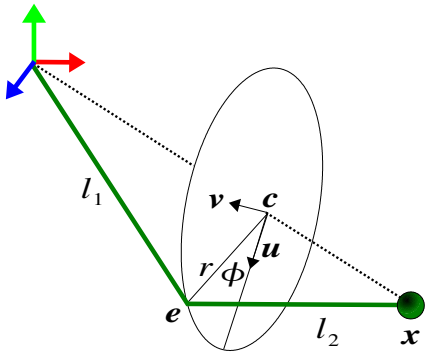


Figure 4-7: The inverse kinematics (IK) for Domo's manipulators are approximated using the analytic approach of Tolani et al. (2000). If the robot's wrist is held fixed at the target **x**, the elbow **e** is still free to swivel about a circular arc centered at **c** with radius r. The arm link lengths l_1 and l_2, along with **x**, uniquely define **c** and r , as well as the unit vectors **u** and **v**, which define a local coordinate system at **c**. The desired joint angles are uniquely defined by the hand location **x** and the swivel angle ϕ of elbow **e**. The desired swivel angle was learned through K-means clustering over arm postures collected while manually moving the arm about its workspace.

4.3.3 Inverse Kinematic Control

An inverse kinematic (IK) controller computes the joint angles that will bring the manipulator end-effector to a target pose in the world. Methods for solving manipulator inverse kinematics abound, and Buss (2004) provides an accessible introduction. In practice, these techniques require careful implementation, otherwise kinematic singularities and joint limits will cause stability and safety concerns. Anthropomorphic manipulators are also kinematically redundant and additional optimization criteria are often required to fully specify the manipulator pose. For these reasons, we use a straightforward, analytic approach for inverse-kinematics described by Tolani et al. (2000). This approach considers only the position of the end-effector and not its orientation. While there are numerous potential criteria for handling the kinematic redundancy, we would like to generate postures that appears natural while avoiding collision with the body.

We briefly describe Tolani's method as shown in Figure 4-7. If the robot's wrist is held

fixed at the target $\mathbf{x}$, the elbow $\mathbf{e}$ is still free to swivel about a circular arc centered at $\mathbf{c}$ with radius r. The arm link lengths l_1 and l_2 define $\mathbf{c}$ and r. We parameterize the arm pose for target $\mathbf{x}$ by the swivel angle ϕ which uniquely defines the elbow position. This is straightforward to derive as $\mathbf{e}(\phi) = \mathbf{c} + r(cos(\phi)\mathbf{u} + sin(\phi)\mathbf{v})$ if unit vectors $\mathbf{u}$ and $\mathbf{v}$ define a local coordinate system at $\mathbf{c}$. Therefore, given an desired ϕ and $\mathbf{x}$, the desired joint angles can be determined using the analytic form of Tolani et al. (2000).

The kinematic redundancy allows an infinite number of swivel angles for a target $\mathbf{x}$. We would like to select ϕ so that the arm appears qualitatively natural, avoids joint limits, and avoids contact with the body. To achieve this, we used unsupervised learning of canonical arm configurations based on captured data. The arm was placed in zero-gravity mode and manually moved through naturalistic arm poses within the safely reachable workspace. Approximately 4000 joint angle samples were recorded. K-means clustering (Duda et al., 2000) ($k = 100$) was then applied to find a set of canonical arm postures. Now, given a target $\mathbf{x}$, we can use a look-up table to find the canonical posture that brings the end-effector closest to $\mathbf{x}$. The swivel angle, ϕ, associated with this posture determine $\mathbf{e}(\phi)$ and consequently the desired arm pose. This method is fast, scaling linearly with k . For smooth, real-time tracking of 3D targets in the world, ϕ is temporally smoothed. Although our selection of ϕ is qualitative, a natural appearing posture is a common criteria for humanoid robots in the absence of optimization criteria.

4.4 Hands

Hands for humanoid robots are notoriously difficult to design. Humanoid arms often impose constraints on the size, weight, and packaging of the hand while demanding sufficient dexterity, strength, and speed. Consequently, these hands often lack force sensing and control. They often lack the mechanical robustness necessary for use in unknown environments where impacts and collisions are common. Unfortunately, incorporating these features can increase the complexity, weight, size, and cost of the hand.

Grasping of unknown objects can be assisted by incorporating passive compliance and force control into a hand. This allows the fingers to independently maintain contact with an object's surface without global knowledge of its shape. Many humanoid hands are position controlled and rely on tactile sensors or load cells at the fingertip to acquire force knowledge.

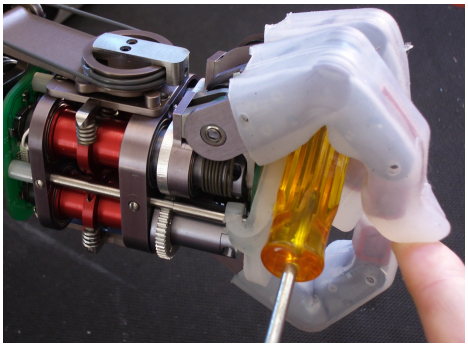


Figure 4-8: Domo's hand shown grasping a screwdriver (with a human finger for scale). The two 4 DOF hands that are approximately of human size and shape. Force sensing and control, combined with a compliant skin, allows them to maintain grasps on many everyday objects such as hand tools.

For example, the original NASA Robonaut hand utilized Force Sensing Resistors to sense the pressure at the fingers (Lovchik and Diftler, 1999). The Gifu Hand employs a combination of load cells and tactile sensors (Kawasaki et al., 2002) , and the Barrett hand used on the Dexter platform integrates a six axis load-cell in each fingertip (Platt et al., 2003c). These types of sensing typically require the robot to maintain fingertip contact in order to sense the grasp forces. In contrast, a hand with force controlled actuators can react to forces applied anywhere along its fingers. This allows for greater responsiveness to the environment and unexpected contact. Recently, humanoid hands that integrate force control with passive compliance have been emerging, notably the Keio hand (Maeno, 2005), the Obrero hand (Torres-Jara, 2005), as well as Domo's hand.

A compliant skin is often employed in hand designs to assist grasping. The skin can conform to an object's surface, expanding the contact surface area. However, the contact deformation of the skin is difficult to predict, and approaches that rely on geometric models for grasping typically avoid compliant skins. Sensing and modelling of the skin's deformation can help alleviate this issue, but these techniques are not yet realistic for real robot hands (Biagiotti et al., 2005) and not aligned with our model-free design approach. In contrast, recent work by Dollar and Howe (2005) has demonstrated blind, sensorless grasping using a completely compliant hand made of cast urethane. In addition, the compliance is also shown to protect the hand from severe impacts and collisions.

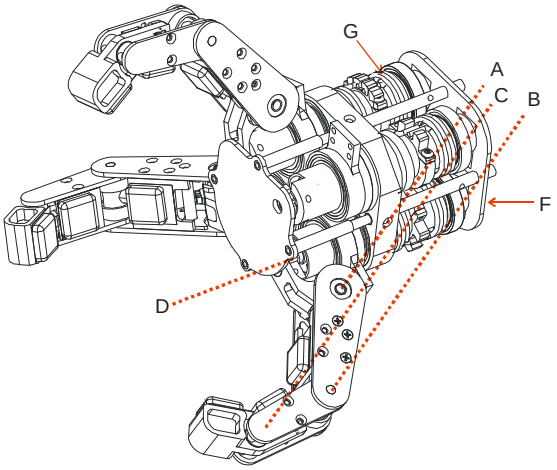


Figure 4-9: The mechanical design of Domo's hands. The four long cylinders are Series Elastic Actuators (G, only three visible). Each of the three fingers has three joints (A,B,C). Joint A is driven by an SEA through a cable-drive. Joint B is passively coupled to A through a rigid cable-drive. Joint C is passively linked by a compression spring to B. The spread between two of the fingers (about axis D) is also driven by an SEA. Electronics for sensor conditioning, force sensing, and the controller interface reside at the rear of the hand (F).

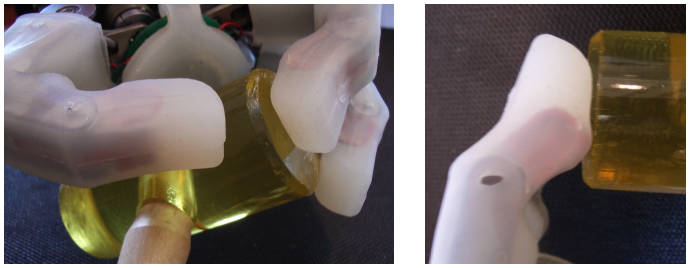


Figure 4-10: In Domo's hands, passive compliance of varying stiffness is integrated into the actuator, finger tip, and silicone skin. Although difficult to model, a compliant skin allows the robot to expand the contact surface during grasping, greatly improving grasp stability.

4.4.1 Hand Design

As show in Figure 4-9, Domo's hands contain four modular SEAs acting on three fingers. The SEA design is shown in Figure 4-3. One actuator controls the spread between two fingers. Three actuators independently control the top knuckle of each finger. The second knuckle is mechanically coupled to the top knuckle, and the fingertip is passively spring-loaded. Springs placed between the motor housing and the hand chassis provide compliance and force sensing for each finger. The three fingers are mechanically identical, however two of the fingers can rotate about an axis perpendicular to the palm. These axes of rotation are coupled through spur gears, constraining the spread between the two fingers to be symmetric.

Each hand weighs approximately 0.51kg , can lift 0.56kg at its fingertip, and is powered by a brushed 13mm Maxon motor. The power and control electronics are embedded in the manipulator forearm. A soft (Shore 00-30) silicone skin covers each finger and the palm. Figure 4-10 shows the compliance of the skin as it conforms to an object. We found that the addition of the skin greatly improved Domo's ability to maintain stable force controlled grasps on objects.

4.5 Head

4.5.1 Head Design

The design of Domo's active vision head is an evolution from previous designs used for Cog and Kismet (Brooks et al., 1999; Breazeal, 2000). It is a mechanical copy of the head

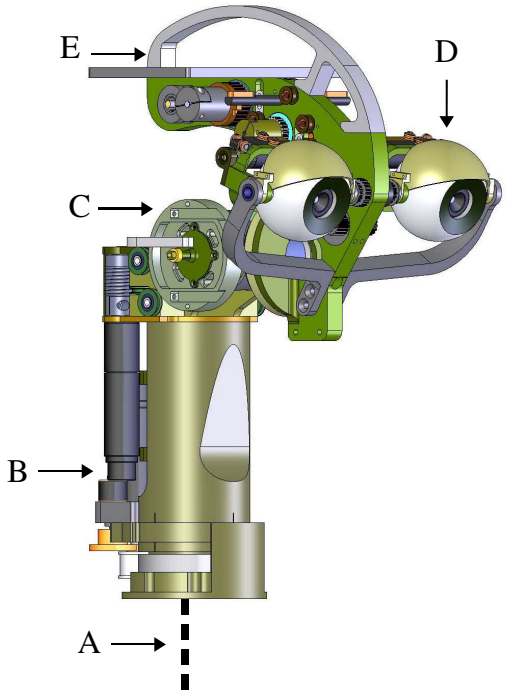


Figure 4-11: Mechanical layout of Domo's active vision head. An SEA driven universal joint (A, not pictured) combined with neck pan (B) provides a compact ball-and-socket like three DOF neck. The upper head provides roll and tilt through a cable-drive differential (C). Two FireWire CCD cameras (D) share a single tilt DOF and have two independent pan DOF. Expressive eyelids provide a final DOF. A 3 axis gyroscope (E, not pictured) provides an absolute reference with respect to gravity.

developed by Aryananda and Weber (2004) for the Mertz robot. The head has 7 DOF in the upper head, a 2 DOF force controlled neck, a stereo pair of synchronized Point Grey FireFly cameras, and a three axis InterSense gyroscope. The upper head provides roll and tilt through a compact cable-drive differential. The two cameras share a single tilt DOF but have independent control of the pan DOF. The head also includes 1 DOF expressive eyelids.

One design goal for the head was the ability to execute human-like eye movement. Human eye movements include saccades, smooth pursuit, vergence, vestibulo-ocular reflex, and the optokinetic response (Kandel et al., 2000). Domo's head is designed to accommodate all but the optokinetic response. Saccades require fast, ballistic movements of the eyes (900°/s) while smooth pursuit requires slow, controlled tracking movements of less than 100°/s. Vergence requires independent control of the eye pan to view objects of varying depth. The vestibulo-ocular reflex typically requires a head mounted gyroscope to counter-rotate the eyes as the head moves. Accommodating these features required particular attention to the eye drive system and motor selection. We use a small gearhead motor (Maxon 8mm 0.5W with 57:1 gearhead) and an efficient cable-drive system for the eye pan to ensure smooth response and sub-pixel servo errors.

4.5.2 Head Control

Humanoid heads are often kinematically redundant in order to allow for expressive head postures. Domo is no exception, having a 7 DOF serial kinematic chain between its torso and each camera. Fortunately, it is often reasonable to control a humanoid head in terms of a single visual target in the world. A camera's pan-tilt motors act to servo the target in the image, and the remaining joints move so as to keep the pan-tilt DOF centered in their joint range. This strategy has been previously demonstrated on the DB and Cog humanoids (Gaskett and Cheng, 2003; Brooks et al., 1999).

On Domo, the head controller takes as input the target $\mathbf{x}^S$ from the visual attention system. As we will describe in the next chapter, $\mathbf{x}^S$ is a continuous, smoothly changing target within the robot's Sensory EgoSphere. This corresponds to a pixel target $[u_r, v_r]^T$ and $[u_l, v_l]^T$ for each respective camera. The cameras independently servo this target to the center of their image. For a target $[u, v]^T$, the desired velocity of the pan DOF is

$$\dot{\theta}_{desired} = -K_p u.$$

This is simply a P velocity control loop with gain K_p. We limit the maximum controller error to 50 pixels within a 320×240 image so as to prevent excessive velocities. This controller is implemented for the left and right camera pan and a similar one for the eye tilt. Combined, these controllers keep $\mathbf{x}^S$ centered in the images. The vergence of the two cameras depends on the depth of $\mathbf{x}^S$. In order to ensure that the eye gaze appears natural, we limit the minimum depth to 200mm .

Next, the head's redundancy is used to keep the pan-tilt joints centered in their range. This assists the pan-tilt controllers when tracking a moving target by expanding the camera's effective field-of-view. For example, the neck pan can act to assist the eye pan. If the eye pan has joint angle θ_e and the neck pan θ_n, then controller for the neck pan has the form

$$\theta_{desired} = \theta_n + K_c\theta_e,$$

for gain K_c. On Domo, the neck-pitch DOF and upper-head-pitch DOF act to help the eye-tilt controller. The neck-pan DOF acts to help the eye-pan controllers. The neck-roll DOF and head-roll DOF are controlled separately to allow for expressive head postures. Careful tuning of these gains allows for the entire 8 DOF head to smoothly track moving targets in the world.

4.6 Manipulator Safety

Our design theme of *cooperative manipulation* requires that a person can safely work alongside a robot. Industrial manipulators are typically dangerous and unsuitable for human interaction. One approach to safety is to combine a lightweight arm with torque control at each joint. For example, the DLR arm can exhibit very low impedance for human interaction (Hirzinger et al., 2002). However, this low impedance is actively controlled and is not intrinsic to the manipulator. Above its control bandwidth, the manipulator cannot exhibit low impedance. The WAM arm is a direct-drive torque controlled manipulator that has been used with success in manipulation research (Guertin and Townsend, 1999). By eliminating the motor's gearhead, a direct-drive design can exhibit low effective inertia, improving its safety. Unfortunately, direct drive-designs are usually bulkier than their counterparts, offsetting these safety gains.

Recently, the field of physical human-robot interaction (pHRI) has emerged as a focused effort to design manipulators that are intrinsically safe for human interaction (Alami et al.,

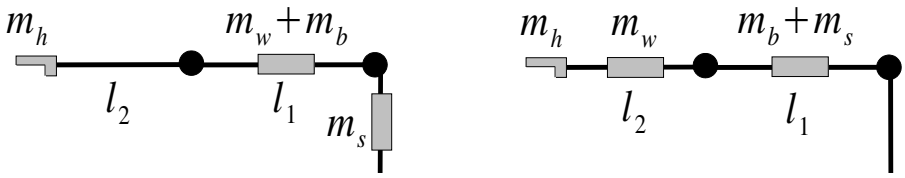


Figure 4-12: A cable-drive design allows for lower inertia and a safer manipulator. (Left) Domo's cable-drive design allows the shoulder mass m_s to be moved onto the body and the wrist mass m_w to be lumped with the bicep m_b. The hand mass is m_h. (Right) A traditional belt-drive design places the actuators near the joints. A point mass model estimates Domo's manipulator inertia as $I_{link} = 0.24\text{kg} \cdot \text{m}^2$ while the traditional design has inertia $I_{link} = 0.52\text{kg} \cdot \text{m}^2$ (using $m_h = 0.45\text{kg}$, $m_w = 1.2\text{kg}$, $m_s = 1.8\text{kg}$, and $m_b = 1.2\text{kg}$).

2006). One notable example is the Stanford DM^2 manipulator which uses parallel motors at each joint of a 4 DOF manipulator (Zinn et al., 2004; Khatib et al., 1999). A large, compliant SEA provides high torques at a low-bandwidth while a complimentary, smaller motor provides lower torques at higher control bandwidths. The combined actuator exhibits the benefits of SEAs but with improved performance. Also, Bicchi et al. (2003) have explored a promising approach utilizing variable stiffness actuators. These are arranged antagonistically, allowing the joint stiffness to be increased during ballistic motions. However, both of these approaches require significant mechanical complexity in order to achieve the improved performance. For many cooperative tasks, a manipulator shouldn't need to move faster than its collaborator. Consequently, an SEA manipulator is sufficient for our purposes.

4.6.1 Measuring the Head Injury Criterion

Robot safety encompasses a wide range of issues, and Ulrich et al. (1995) provide a comprehensive analysis of common robot safety criteria. One of the most significant means of injury is through unexpected physical contact. A robot can injure a person through static and impact contact forces, static and impact pinch forces, or crushing forces from its weight. Of these, impact contact forces are typically the most dangerous. A software bug, a glitch in the controller, or poor sensing of human proximity can cause a manipulator to make harmful contact. In the worst case, contact is made with the head. Consequently, the

Head Injury Criterion (HIC) is the most commonly used index to evaluate robot manipulator safety (Versace, 1971). The HIC, originally defined for the automotive industry, is a function of the impact acceleration of the head and the duration of impact. Typically it is solved numerically, though with some simplifications it can be written in a modified form that is relevant for compliant manipulators (Bicchi and Tonietti, 2004):

$$HIC = 2\left(\frac{2}{\pi}\right)^{\frac{3}{2}}\left(\frac{K_{cov}}{M_{oper}}\right)^{\frac{3}{4}}\left(\frac{M_{rob}}{M_{rob}+M_{oper}}\right)^{\frac{7}{4}} v_{max}^{\frac{5}{2}},$$

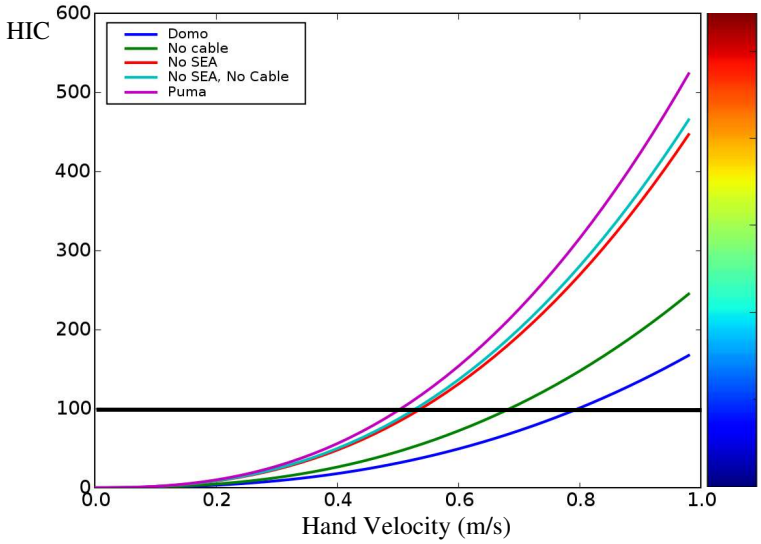
where the stiffness of the arm covering is K_{cov}, the mass of the operator's head is M_{oper}, and the maximum velocity of the end-effector is v_{max}. The effective robot mass M_{rob} is a function of the intrinsic stiffness of the manipulator as well as the rotor and link mass

$$M_{rob} = M_{link} + \frac{K_s}{K_s+\gamma}M_{rot},$$

where γ is the rigid joint stiffness. As noted previously, a low drivetrain stiffness K_s serves to decouple the rotor mass M_{rot} from the link mass M_{link}. If a large gearhead is used, M_{rot} can be significant.

An HIC value around 100 is generally safe for human contact while an HIC of 1000 is potentially fatal. A manipulator's HIC index can be improved in a number of ways: lowering K_{cov} through a soft covering, limiting in software the velocity v_{max}, lowering K_s via passive compliance, and designing for a low manipulator mass M_{link}.

With Domo, we adopted all but the first method. The angular velocity of the joints are limited on the DSP, SEAs provide a low K_s, and M_{link} is kept low by using a cable-drive transmission. As shown in Figure 4-12, when the arm is outstretched (and hand velocity highest), Domo's cable-drive manipulator has approximately half the inertia as its belt-drive counterpart. For Domo, $v_{max} = \left\|\mathbf{J\dot{\Theta}_{max}}\right\| \approx 1$m/s based on its kinematic model and estimated actuator saturation. In Figure 4-13 we see that when $v_{max} = 1$m/s, Domo's HIC is significantly improved over a non-cable-drive version of the manipulator and the Puma 560. We also see that a DSP limited velocity of $v_{max} = 0.84$m/s will keep Domo's manipulator safe for human interaction.



Configuration	HIC	$v_{max}(m/s)$
Domo	167	.84
Domo, non-cable-drive	238	.70
Domo non-SEA	469	.53
Domo, non-cable-drive, non-SEA	489	.52
Puma 560	550	.50

Figure 4-13: The Head Injury Criterion (HIC) for Domo's manipulator. An HIC of 100 is generally safe for human interaction while an HIC of 1000 can be fatal. (Top) Estimated HIC versus hand velocity for five manipulators. (bottom) The estimated HIC for each configuration given $v_{max} = 1m/s$ and also the maximum velocity v_{max} to achieve $HIC = 100$. We used the following model parameters for Domo: $K_s = 120kN/m$, $\gamma = 3000kN/m$, $K_{cov} = 25kN/m$, $M_{oper} = 4kg$, and $M_{rotor} = 12Kg$. $M_{link} = 2.8kg$ for Domo while a non-cable-drive configuration of the manipulator has $M_{link} = 4.6kg$. The Puma 560 has $M_{rob} \approx 25Kg$ (Zinn et al., 2004).

4.7 Discussion

A robot's design should be specialized to match its environment and the tasks expected of it. Human environments have particular characteristics that render most industrial manipulators inappropriate. Although Domo's design has many details that are unique to the robot, we can find general characteristics that make it suitable for human environments:

- Intrinsically safe for human interaction.
- Passive compliance at each joint.
- Force sensing and control at each joint.
- Human form to allow for working with objects designed for people.

These four characteristics can serve as useful design objectives for researchers as the field moves towards standardized platforms.

CHAPTER 5

Building Brains

In this chapter we present the organization of Domo's computation systems, a control architecture named SLATE, and a method for composing manipulation tasks out of SLATE's behavioral modules.

5.1 Computational Organization

The computational organization is a critical aspect of any complex robot. High degree-of-freedom robots that perform substantial perceptual processing require carefully designed architectures that can support scalable computation, real-time control, and low-latency data transmission. One approach that many humanoid projects have moved towards is to distribute the computation across many PCs and embedded controllers (Ly et al., 2004; Brooks et al., 1999; Metta, 2000).

We have followed a similar path, employing 15 Linux PCs networked by a 1Gb LAN. Our control decomposition is reminiscent of the layered NASREM architecture (Albus et al., 1989), where the controller rates decrease by an order of magnitude with each additional layer of abstraction away from the body. As shown in Figure 5-1, Domo's architecture is organized into four broad layers: the physical layer, the DSP layer, the sensorimotor layer, and the behavior layer. The first two layers are physically embedded on the robot while the

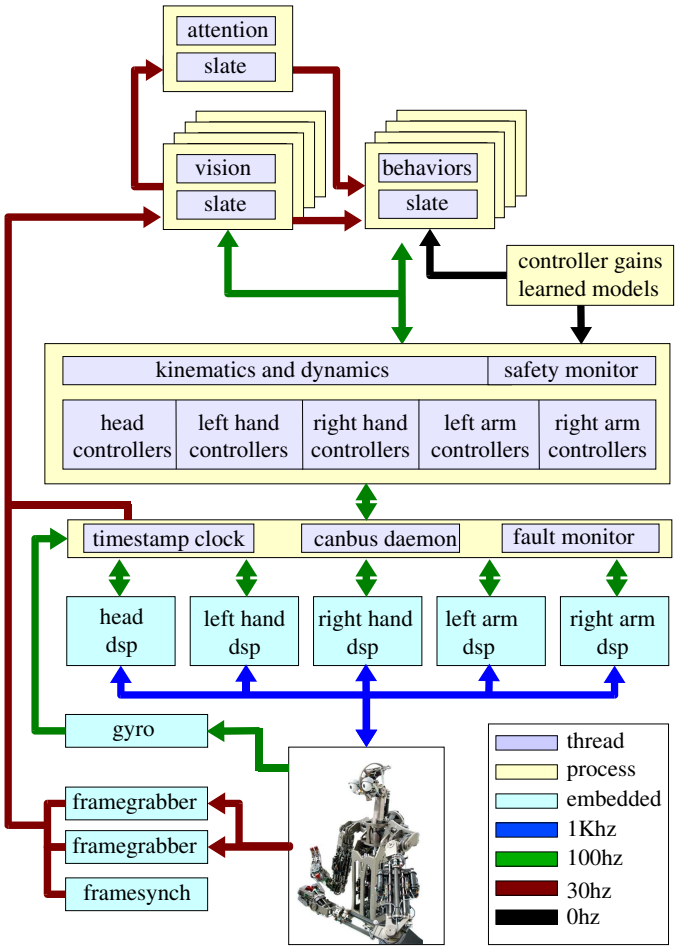


Figure 5-1: The robot's computation is distributed across five embedded DSP controllers and 15 Linux PCs. Processes (yellow) are run on dedicated PC nodes. The DSPs handle real-time control and sensor acquisition at 1kHz. Synchronized image pairs are captured to a dedicated PC at 30Hz. A daemon interfaces to the DSPs through four 1MHz CANbus channels. It also generates a timestamp from the CANbus clock to synchronize all image and sensor data. A 100Hz sensorimotor process computes kinematic and dynamic models, implements higher level motor control, monitors manipulator safety, and computes the commanded setpoints for the DSPs. Our behavior-based architecture, SLATE, coordinates the visual processing, attention system, and task behaviors. These run at parameterized rates of up to 100Hz . This decomposition is reminiscent of the layered NASREM architecture (Albus et al., 1989).

latter two run on the Linux cluster. Each layer is briefly describe in the next sections.

Physical Layer

The physical layer constitutes the electromechanical resources embedded in the robot. This includes: 12 brushless DC motors and amplifiers in the arms, 17 brushed DC motors and amplifiers in the hands and head, a force sensing potentiometer at 22 of the joints, a position sensing potentiometer at all 29 joints, a position sensing encoder for each of the 7 joints in the upper head, a gyroscope in the head, two cameras, a speaker, and a wireless microphone.

DSP Layer

The DSP layer provides real-time force, position, and velocity control of the actuators as well as sensor data acquisition. Each body appendage uses a 40MHz Motorola 56F807 DSP controller that controls up to 10 DOF at 1kHz. These controllers are networked together using four 1MHz CANbus channels interfaced to a Kvaser PCIcan 4xHS PCI card residing on a Linux PC. By using embedded controllers, we gain complete control over the real-time aspects of the sensorimotor system. This avoids pitfalls commonly encountered when using PC based controllers such as operating system timeouts and complicated startup routines. Also, each DSP continually monitors the traffic of its CANbus channel. If the CANbus is disconnected, the arms will switch into a zero-force mode, protecting the arms and nearby people. The arms will resume activity when the CANbus is plugged back in.

Sensorimotor Layer

The sensorimotor layer creates a coherent snapshot of the proprioceptive data coming from the DSPs, computes a kinematic and dynamic model, and implements a set of higher-level controllers. These 100Hz controllers include smooth visual tracking, inverse kinematic reaching, and operation space control of the arm (Khatib, 1987). We use the YARP software package developed by Metta et al. (2006) to provide TCP/IP interprocess communication among the Linux cluster's 1Gb LAN. We implemented a custom PYTHON-YARP interface, allowing us to dynamically define and transmit data structures between processes. Additionally, two FireWire framegrabbers provide synchronized image pairs to the cluster at 30Hz. Finally, the image and sensory data are timestamped using the hardware clock from

the CANbus PCI card. This ensures synchronization of the data up to the transmit time of the 1Gb LAN.

Behavior Layer

The behavior layer implements the robot's visual processing, learning, and task behaviors. These algorithms run at parameterized rates up to 100Hz within the behavior-based architecture described in the next section.

5.2 SLATE: A Behavior-Based Architecture

We would like robots to exhibit the creature robot qualities of coherent behavior, responsiveness to the environment, and adequacy in task execution. For Domo, we have developed a behavior-based architecture named SLATE. Reactive, or behavior-based, designs such as the subsumption architecture of Brooks (1986) are well suited to the creature robot objectives. What is a an architecture? Mataric (1992) provides the following definition:

> An architecture provides a principled way of organizing a control system. However, in addition to providing structure, it imposes constraints on the way the control problem can be solved.

Following Mataric, Arkin (1998) notes the common aspects of behavior-based architectures:

- emphasis on the importance of coupling sensing and action tightly
- avoidance of representation
- decomposition into contextually meaningful units

Roboticists have developed many variants of behavior-based architectures. We refer to Arkin (1998) for a review. Loosely stated, SLATE is a lightweight architecture for organizing perception and control. It is implemented as a programming abstraction in PYTHON, allowing one to easily define many small computational threads. These threads run at parameterized rates within SLATE's non-preemptive scheduler. Importantly, SLATE makes it easy to specify time-contingent behavior and to distribute computation across multiple machines. SLATE falls short of being a full behavior-based programming language such as the

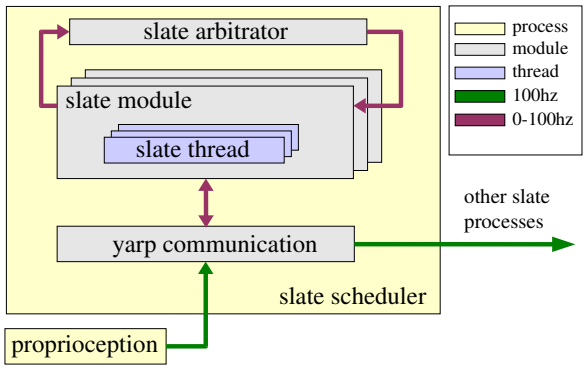


Figure 5-2: The organization of the SLATE architecture. Each SLATE process runs on a node within the Linux cluster. The process starts its own non-preemptive scheduler and a communication interface to external processes. At startup, the robot's proprioceptive stream is automatically imported into the process global namespace (yellow). Within a process, a module defines a namespace and a set of lightweight threads (and finite-state-automaton). These threads are scheduled at rates up to 100Hz. Threads communicate through the global namespace or through wires, and wires provide arbitration between conflicting writes after every scheduler cycle.

well known L-MARS (Brooks and Rosenberg, 1995) language, but its design draws from this work. SLATE also benefits from its use of a lightweight interpreted language such as PYTHON. PYTHON allows for rapid development cycles and provides a large toolbox of scientific, machine learning, and vision libraries such as the open-source package PYSENSE (Kemp, 2006). Computationally expensive algorithms in SLATE are optimized using PYTHON extensions in *C*.

5.2.1 Slate Components

The basic control structure of SLATE is shown in Figure 5-2. It is built out of the following components:

Process At the highest level, SLATE consists of many PYTHON processes distributed across our Linux cluster. Processes pass messages back and forth at rates up to 100Hz using TCP/IP via YARP. Messages can be data structures of arbitrary types and can be defined on the fly. Each SLATE process can also dynamically subscribe to other SLATE

processes, image streams, motor command streams, or proprioceptive streams. By default all SLATE processes subscribe to the robot's proprioceptive stream. Within a process, data can be shared in the global namespace.

Scheduler A SLATE process executes within the Linux kernel as a standard user process and it implements a non-preemptive scheduler that runs as a user thread. The scheduler is at the center of SLATE. It's task is to schedule, and execute, short pieces of code encapsulated within the methods of PYTHON classes. Each short piece of code is registered as a SLATE object within the scheduler. Each SLATE object has a defined update period specified in milliseconds but limited to the resolution of the scheduler's time quanta of 10ms. Every quanta, the scheduler iterates through its list of objects, determines which are scheduled for update, and executes their update method. SLATE objects can be threads, FSAs, and monostables. A typical process will maintain 20-200 objects within the scheduler. If all scheduled objects cannot be updated within the time-window of the quanta, the actual update rate will lag the desired rate.

Module A module is an instance of a PYTHON class. Its function is to encapsulate a group of related SLATE objects within a shared namespace. As modules are readily parameterized, it is straight forward to define multiple module instances acting on different robot limbs or perceptual streams, for example.

Thread A thread implements a small amount of computation such as processing an image frame or updating a sensor model. It is implemented as a class method of a module. A thread cannot yield and therefore the method must run within a fraction of the scheduler time quanta. The scheduler will call the class method at a defined update rate. Multiple threads within a module can communicate through shared variables in the module namespace.

Port A port allows YARP based TCP/IP interprocess communication between SLATE modules. Connections between modules are formed through a static naming scheme. A thread can read and/or write any type of data to a port, but we assume by convention that the data read from a port is of an expected format. Ports can also connect into the robot's raw sensor, camera, and DSP controller streams.

FSA An FSA implements a time-contingent finite state automaton. Each state of the FSA

is implemented as a class method of a module. The active state is periodically called by the scheduler at a defined update rate. At the end of each call, the class method optionally returns the next FSA state. Each state also has an associated timeout. If the FSA remains in a given state longer than the timeout, the active state will automatically advance to a defined next state. Optionally, an FSA will automatically reset to an idle state if it loses control of a robot resource, such as motor control of a joint.

Monostable A monostable is a simple timing element with a defined time value, in milliseconds. When a thread triggers a monostable, it is reset to the time value. On every cycle the scheduler decrements the value by the time quanta until the value is zero. Monostables are globally accessible to all modules within a SLATE process.

Wire A wire is a SLATE object that supports read/write operations. It is declared in the SLATE global namespace and allows communication between threads and FSAs. Data written to a wire is not available for read until the next scheduler cycle, ensuring that two threads do not have inconsistent views of the same data.

Arbitrator An arbitrator resolves write conflicts on a wire. Conflicts are resolved through a mix of dynamic and fixed priorities. A fixed priority for each module is hardcoded into its definition. During each scheduler cycle, any thread can increment a module's priority by some amount above its fixed priority. At the end of the cycle, the arbitrator grants control of the wire to the module with the highest priority. All data written to a wire by its controlling module will be available to a reading thread during the next scheduler cycle. At the start of each cycle, the module's priority reverts to its fixed value, so a thread must constantly send priority adjustments if it intends to maintain control. Also, a hysteresis setting in the arbitrator prevents rapid switching of wire ownership.

Tools Good development tools and libraries are extremely important when developing robots. SLATE makes use of numerous optimized PYTHON-C extensions for linear algebra, machine learning, and computer vision. In addition, we can easily write our own extensions using the SWIG package. We have also developed useful debugging tools. A SLATE thread can dynamically generate a remote GUI and stream data to

and from it. Additionally, SLATE can be run from the PYTHON interactive console, allowing for online coding and testing of behaviors.

5.2.2 Example Program

The following PYTHON pseudo-code demonstrates how SLATE modules are written. It implements a controller for Braitenberg's (1984) vehicles 2a and 2b using the *SeekLight* and *AvoidLight* modules. The modules are prioritized so the robot will drive towards a light source by default. However, if it senses too much light, then *AvoidLight* assumes control of the motor wire and the robot turns away from the light. The wire arbitrator ensures that *AvoidLight* is active for at least 1000ms. The motor threads run every 100ms while the inhibit thread runs every 500ms as it could potentially perform heavier perceptual computation.

```
#-------------------------------------------------------------
class SeekLight: #Vehicle 2b
     def motor_thread( ):
          light=slate.robot.sensors
          slate.wire_write('Wire:Motors',[light[1],light[0]])
#-------------------------------------------------------------
class AvoidLight #Vehicle 2a
     def motor_thread( ):
          light=slate.robot.sensors
          slate.wire_write('Wire:Motors',[light[0],light[1]])
#-------------------------------------------------------------
class MotorWriter
     def output_thread( ): #Output to controller
          desired=slate.wire_read('Wire:Motors')
          slate.robot.motors=desired
     def inhibit_thread( ): #Avoid too much light
          if (light[0]+light[1])>100:
               slate.priority_adjust('MotorArb','AvoidLight',3.0)
#-------------------------------------------------------------
```

```
# Slate declarations
priority={SeekLight:2.0, AvoidLight:1.0}
a=slate.arbitrator('MotorArb', priority, hysterisis=1000)
slate.wire('Wire:Motors',arbitrator=a)
seek=slate.module(SeekLight( ))
avoid=slate.module(AvoidLight( ))
writer=slate.module(MotorWriter( ))
slate.thread(seek.motor_thread,     ms=100)
slate.thread(avoid.motor_thread,    ms=100)
slate.thread(writer.output_thread,  ms=100)
slate.thread(writer.inhibit_thread, ms=500)
#-------------------------------------------------------------
```

5.2.3 Slate on Domo

SLATE is distributed across up to 15 Linux computers. Because the computation is not embedded on the robot, we can afford to conservatively run only one SLATE process per machine. The complete behavior system described in this thesis runs on about 10 machines, where the majority of the processes are concerned with real-time perception. We consolidate the distributed perception and control processes into a single task-level module. This process runs at 50Hz and implements roughly 125 threads, 40 wires, 10 arbitrators, and 35 FSAs.

5.3 Designing Tasks in SLATE

In this section we describe how manipulation tasks are composed out of SLATE modules. Domo's observable behavior is a consequence of the interaction of many modules with each other, people, and the environment. Unfortunately the word *behavior* has many meanings within the robotics literature, although typically it describes a simple transformation from sensor information to motor action (Arkin, 1998). We avoid the difficult semantics of the word *behavior* and instead use the word *module.* A module is a well define component of SLATE, corresponding to a perceptual algorithm, motor controller, or both[1].

[1]We name modules using two or three descriptive words in italics, such as *RelaxArm* or *PersonSeek*.

5.3.1 Coordinating Modules

As we described in Section 3.1, the work of Connell (1989) demonstrated that a robot's responsiveness and robustness can be improved by coordinating its behaviors indirectly, through the world, instead of through an internal data wire. In this form of interaction, one module executes a motor action that influences the perceptual state detected by another module. Consequently, modules will make control adjustments using constant perceptual sensing rather than potentially incorrect internal models.

Modules can also communicate by dynamically adjusting each others' arbitration priority. This increases the likelihood that a desired module will take control of a fixed resource, such as an arm controller, but does not guarantee it. Gaining control of a resource depends not just on the module's priority, but also on the module's internal estimate of its readiness based on perceptual conditions.

As an example, consider a module *PersonTouch,* that causes an arm to reach out and touch a person. As shown in Figure 5-3, *PersonTouch* works as follows:

1. Compute internal readiness based on a desired perceptual state
   (a) E.g., Can only act when module *PersonDetect* signals a person is present
2. Increase priority of modules that may cause readiness
   (a) E.g., Increase the priority of the *PersonSeek* module that will scan the room for a person
3. Constantly compute a desired action or perceptual feature
   (a) E.g., Compute a joint trajectory that will reach to the person
4. Output the readiness value and the computed action to a SLATE wire. Allow SLATE to decide if the action is executed.

We see that *PersonTouch* relies on the successful execution of *PersonDetect*, but is limited to increasing the priority of *PersonSeek*. Consequently, *PersonTouch* and *PersonDetect* communicate through the world. If the person suddenly hides from the robot, *PersonTouch's* readiness will become false. This will cause the *PersonTouch* to automatically relinquish

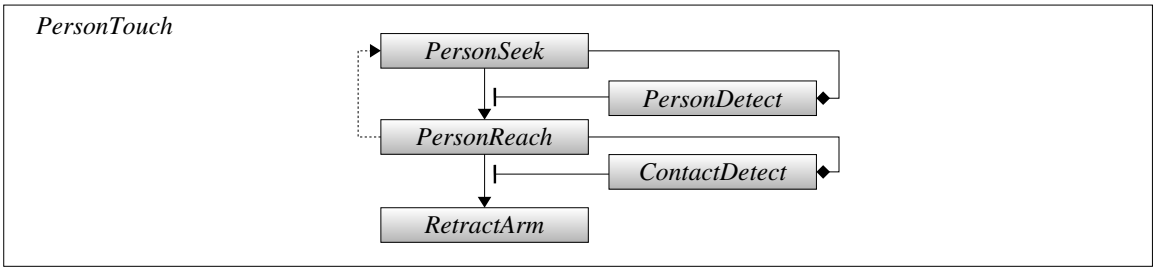


Figure 5-3: The flow of control within a SLATE module. While a module may activate other modules during execution, we depict only the significant modules. A module starts at the top-most state in the diagram and relinquishes control after the bottom-most state. A state transition (arrow) occurs contingent (bar) on perceptual feedback or the activation of another module. Often, modules will communicate *through the world* (diamond) by taking actions that increase the likelihood of another module detecting a perceptual feature. Exceptions within a module can cause reset transitions to a previous state (dashed line). In this example, the *PersonTouch* module causes the robot to reach out and greet a person through touch. First, *PersonSeek* is activated, causing the robot to look around and increasing the likelihood of *PersonDetect*. When *PersonDetect* signals that a person is present, *PersonReach* brings the hand near the person. This action increases the likelihood that *ContactDetect* will sense a person touching the arm. If *ContactDetect* is not signalled, the module reverts to search for a new person. Otherwise, *RetractArm* brings the arm down to the robot's side.

arm control to the next highest priority module. Although this type of coordination requires careful hand design, it can provide a rich responsiveness to unexpected dynamics in the world.

The flow of control within *PersonTouch* is shown in Figure 5-3. We will use this type of graphical depiction throughout this thesis.

5.3.2 Decomposing Manipulation Tasks

As shown in Figure 5-4, Domo's modules are integrated hierarchically into one SLATE process with a single root module, *HelpWithChores*. This integration allows Domo to behave as a creature robot instead of as a compendium of individual experiments. The hierarchical composition also allows for reuse of modules as the robot's skill set is expanded. The

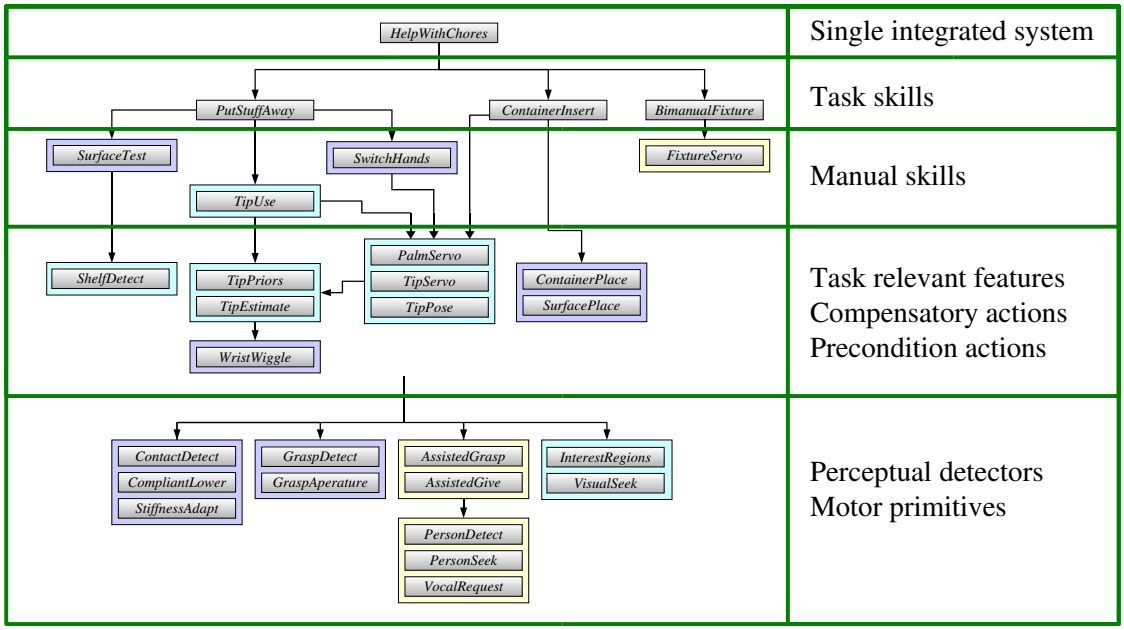


Figure 5-4: Domo's modules are integrated hierarchically into one SLATE process with a single root module, *HelpWithChores*.

hierarchy can be broadly decomposed into four broad layers. We briefly describe each of these layers next.

Perceptual Detectors and Motor Primitives

The lowest layer of the hierarchy implements shared perceptual and motor primitives. The robot perceives the world through simple detectors of sparse perceptual features. As we will see in subsequent chapters, the detector modules in SLATE include (among others):

- The aperture of the robot's grasp
- The distal tip of a grasped object
- Manipulator contact with the world
- The circular opening of a grasped container
- The contact surface of the robot's palm

The share motor primitives include (among others):

- Modifying joint stiffness
- Reaching towards a person
- Reaching along a camera ray
- Directing the eye gaze

Learning can be readily introduced into this layer of the hierarchy. We use offline learning to assist in feature detection, such as mapping the kinematic state of the hand to the grasp aperture (as we will see in Section 7.3). We use online learning to adapt to environmental variability, such as the apparent change in human skin tone during the day. When simple prior models are employed, we often use perceptual feedback to compensate for model inaccuracies. For example, if we use a prior model to predict the location of a visual feature, we then initialize a feature detector at the expected location and sense the true location.

Compensatory Actions, Precondition Actions, and Task Relevant Features

Within this layer of the hierarchy, one class of modules take *compensatory actions* in order to reduce perceptual uncertainty, as described in Section 2.1.3. For example, we will present the *ContainerPlace* module that rests a grasped container, such as a cup, on a table in order to passively align the container to a known orientation. Another class of modules take *precondition actions*. These are actions which pose the robot's body prior to control. For example, a precondition action might bring the hand within the field-of-view before visually controlling it.

A final class of modules localize task-relevant features. This can require precondition actions, compensatory actions, as well as the accumulation of detections over time to form robust estimates of stable features in the world. These modules also provide control with respect to these features.

Manual Skills

In this layer, fundamental manual skills are generated by coordinating the detection and control of task relevant features over time. One example we will present is the *SwitchHands*

module, which passes an object from one hand to the other. In Section 5.3.3 we describe a general algorithm for composing manual skills.

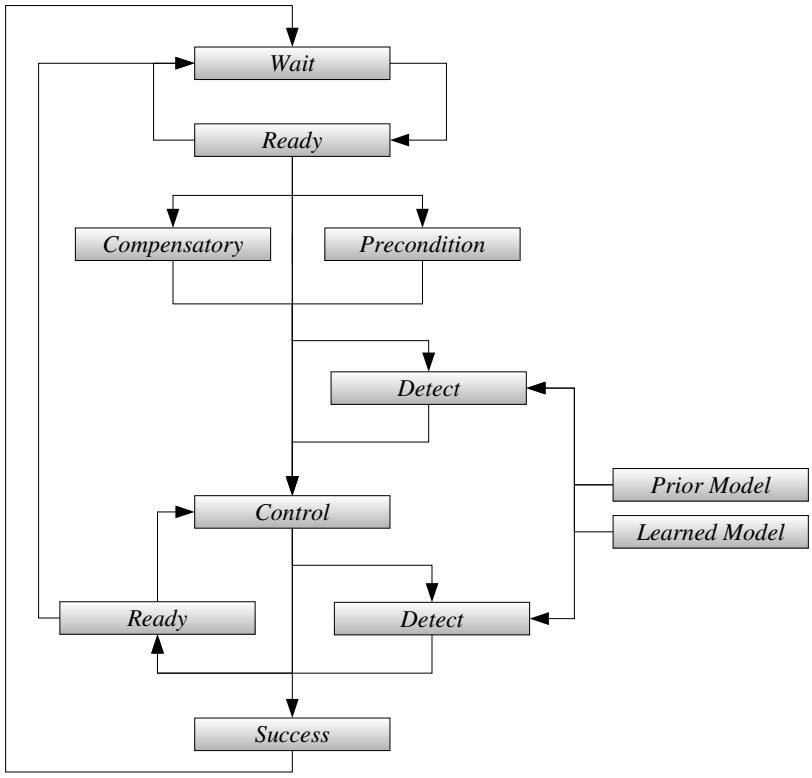
Task Skills

In this layer, manual skills are coordinated to accomplish manipulation tasks. A human collaborator is included in this coordination through interaction with the robot. For example, in *PutStuffAway*, the collaborator first verbally requests Domo to take an object from the person, and then requests Domo to put the object on a shelf. In this way, the person uses their knowledge about the task to coordinate task execution. The person can also readily detect failure and adapt the task plan accordingly. For example, if Domo drops an object, the person can pick it up and hand it back to Domo.

5.3.3 An Algorithm for Manual Skills

A manual skill within SLATE requires the careful coordination of many modules. For example, the *SwitchHands* skill requires the precondition action *WatchHand,* the compensatory action *StiffnessAdapt*, the feature detection of *PalmDetect,* as well as the control action *PalmServo*. We can generalize the flow of control required for this type of skill into the algorithm shown in Figure 5-5. We will use this algorithm to describe the various manual skills developed in this thesis.

We can describe this algorithm as the succession of stages shown in Figure 5-5. In the *Ready* stage, the module computes its readiness based on perceptual preconditions. It increases the activation of modules that may cause these preconditions to be met, and monitors the perceptual detectors that signal these preconditions. Throughout the algorithm execution, if the module loses its readiness (drops an object, for example), then the algorithm returns to a previous stage in order to reacquire readiness. Once a module is ready, and SLATE has granted it control of a resource (such as an arm), the module optionally initiates *Compensatory* and *Precondition* actions. These prepare the robot's body for control and act to assist in the detection of a task relevant feature. Next, in the *Detect* stage, the module coordinates the initial detection of a controllable feature (such as an object's tip). Both learned and prior models may be used in the feature's detector. In the subsequent *Control* stage, the manipulator is servoed to a control objective. This can be achieved using feedforward control based on the initial feature detection, or using feedback based on



Ready Monitor perceptual features required for readiness and activate modules that may cause these features to be detected.

Compensatory Take actions that assist perception and reduce uncertainty.

Precondition Pose the robot body to support detection of task relevant features and to prepare for control.

Detect Detect the task relevant feature(s), incorporating learned or prior models.

Control Servo the manipulator to a control objective, incorporating perceptual detection of the feature into the controller.

Success Monitor perceptual features that indicate success.

Figure 5-5: A generic control algorithm for manual skills. The flow of control begins in the *Wait* stage and ends in the *Success* stage.

updated feature detections. During control, the perceptual features related to task success are monitored. When *Success* is signaled, the module relinquishes control of the resource.

This algorithm integrates our three design themes into a single controller. The *Compensatory* and *Precondition* stages leverage the robot's embodiment. The *Detect* and *Control* stages accomplish the skill in terms of task relevant features. Finally, a collaborator can be brought into the loop by cueing the controller when it is required, when it has succeeded, or when it has failed.

5.4 Discussion

We have presented a control architecture for designing manipulation tasks within human environments. The behavior-based architecture, SLATE, allows us to design tasks by coordinating computational modules over time contingent on perceptual feedback. As it is a fairly modular decomposition, we can reuse modules for different tasks. However, the coordination of these modules is currently achieved by careful hand design. This can be a time consuming process. Ideally, this coordination would be learned from experience or from human demonstration. We have also presented a general control algorithm for manual skills that integrates our three design themes. This algorithm serves as a design template for the manual skills developed in this thesis. It should also be of practical use to other researchers as they develop manipulation skills.

CHAPTER 6

Visual Attention System

A visual attention system allows a robot to select interesting, salient items in a complex world filled with numerous competing distractions. It reduces the perceptual complexity of the environment to a small number of salient regions that can be analyzed in more detail using computationally more intensive perceptual processes. It can also provide the robot with a short-term perceptual memory through the spatial registration of perceptual features over time.

Models of human visual attention, such as Wolfe's *Guided Search 2.0* (Wolfe, 1994), have generated design strategies for many humanoid vision systems. For example, a system developed by Breazeal et al. (2001) combines several low-level image features into a single saliency map. Each feature's saliency is weighted according to the current motivation and drives of the robot. The region with the highest saliency is used to direct the gaze of the robot.

In the spirit of Itti et al. (1998), we have implemented a visual attention system as a means to consolidate many disparate perceptual streams into a single spotlight of attention. These streams provide a continuous source of simple, visual features that are relevant for cooperative manipulation tasks. Their perceptual algorithms run on different computers at different rates. Accordingly, we use a spatial representation called the Sensory EgoSphere (SES) as a mechanism for perceptual synchronization and spatial localization (Peters et al.,

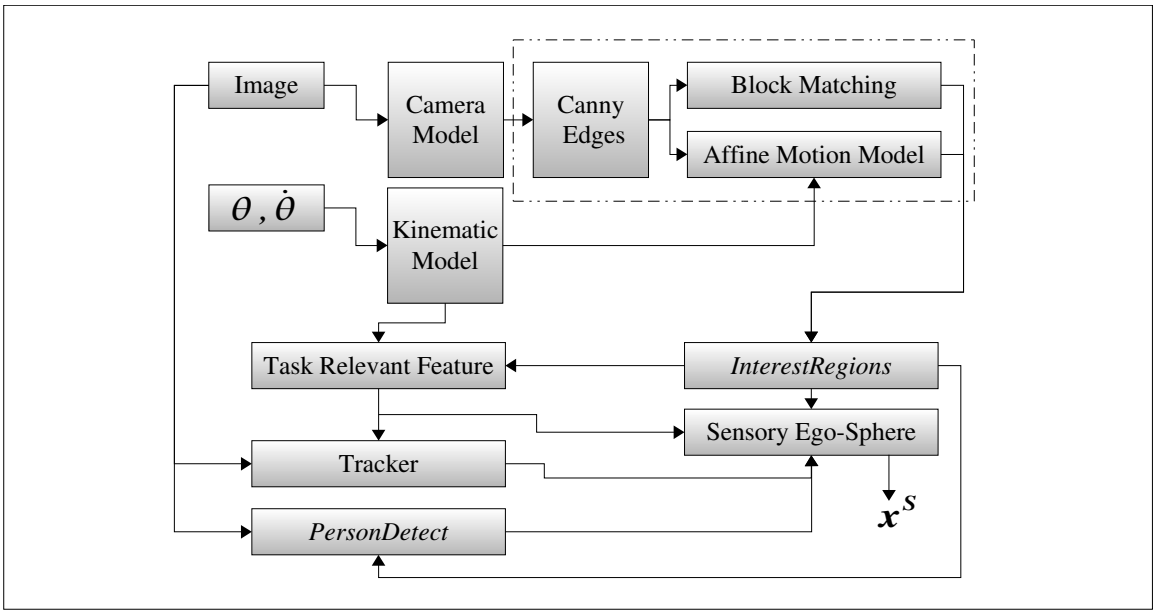


Figure 6-1: Overview of the visual attention system. Images are either 160 × 120 or 320 × 240 pixels. The visual motion model (dashed box) selects for fast moving edges in the foreground. It can be informed by the kinematic model to discount ego-motion and to select for the robot's hand. The *InterestRegions* module detects convex shaped edges, both moving and stationary, at multiple scales. This is the principal visual feature that we use throughout this thesis. A block-matching tracker is used to track these features during a task. The *PersonDetect* module integrates a face detector, skin color model, and interest points in order to detect and track human features. *PersonDetect* will be described in Chapter 8. Finally, the Sensory EgoSphere consolidates the visual features into a single attention point, $\mathbf{x}^S$, to direct the robot's eye gaze.

2001). The principal components of Domo's visual attention system are described in Figure 6-1. In this chapter we present the visual motion model, the *InterestRegions* module, as well as the Sensory EgoSphere.

6.1 Visual Motion

Visual motion can be a robust and powerful attentional cue for robot perception. For example, Fitzpatrick et al. (2003) use the motion generated through a robot's contact with the world to visually segment an object from its background. In human environments, visual motion often corresponds to objects under a person's or robot's control. However, there can be multiple sources of visual motion, such as the ego-motion of the head, a person within the environment, or motion of the manipulator. Segregation of multiple motion sources can be difficult. On Domo, we use a visual motion model to detect image points that are moving significantly with respect to the background. This model, developed by Kemp (2005), is briefly reviewed in the next section. We then show how kinematic predictions of head motion and manipulator motion can be used to segregate multiple motion cues.

6.1.1 Visual Motion Model

In the approach of Kemp (2005), the global background motion is estimated using by fitting a 2D affine model to the edge motion(Stiller and Konrad, 1999). Individual edges are then weighted based on their difference from the model's global prediction. Consequently, an edge's weight reflects both the estimated speed of an edge point and its difference from the global background motion.

The global background motion, $\mathbf{A}$, is represented as a 2×3 affine matrix that transforms a pixel location $[u_1, v_1]^T$ in image I_1 into pixel location $[u_2, v_2]^T$ in image I_2,

$$\begin{bmatrix} u_2 \\ v_2 \end{bmatrix} = \begin{bmatrix} a_1 & a_2 & a_3 \\ a_4 & a_5 & a_6 \end{bmatrix} \begin{bmatrix} u_1 \\ v_2 \\ 1 \end{bmatrix}. \tag{6.1}$$

This model can account for global changes in translation, scale, rotation, and shearing. The algorithm for estimating $\mathbf{A}$ is described in detail in Kemp (2005). It can be summarized as follows:

1. Find the edges in consecutive images I_1 and I_2 using a Canny edge detector (Canny, 1986).

2. Using the standard technique of block matching (5×5 block over 11×11 window), estimate the translation $\mathbf{t}_i$ of each edge pixel between images.

3. For each translation $\mathbf{t}_i$, also compute the covariance matrix, $\mathbf{C}_i$, of the block matching error.

4. Use weighted linear-least-squares to fit the model $\mathbf{A}$ to translations $\mathbf{t}_i$ when weighted by $\mathbf{C}_i$.

5. Iterate the fitting process in order to remove edge points that are unlikely to be part of the background motion.

6. Using the Mahalanobis distance, go back and weight each edge by how well it fits the motion model.

This algorithm generates a weighted edge map where the weight of each edge is proportional to its velocity and its difference from the global background motion. Consequently, it selects for fast moving edges in the foreground. By computing on edges instead of pixels, the model can be estimated in real-time and also reduces the use of uninformative points. The algorithm executes at approximately 25Hz using a 3GHz Pentium and 160×120 images and at 11Hz for 320×240 images.

The weighted linear least squares solution can be considered the maximum likelihood estimation of the model $\mathbf{A}$ where the error is Gaussian distributed according to the covariances $\mathbf{C}_i$. Also, the Mahalanobis distance is in units of image pixels, so working with these distances is intuitive. Domo's visual attention system selects the top n edge points with the largest weights as the most salient locations in the image. If this weight is below a conservative threshold, the detection is ignored. Sample output from the algorithm is shown in Figure 6-2. Because the algorithm selects for edges moving with respect to the background, small amounts of camera motion are tolerated. This is important when working with an active vision head such as Domo's.

Figure 6-2: Output from the visual motion model using head motion prediction (bottom) and without (top) while the camera tracks a moving person. The edge map (left) shows that the edges of the person are weighted more strongly when the prediction is included, allowing the person's head to be detected. The green circle shows the interest region selected by the *InterestRegions* module of Section 6.2. The direction and magnitude of the motion estimated by the affine model are shown by the green lines (magnified 2×).

6.1.2 Visual Motion Prediction

Predictions can tell perceptual processes at least two things: where to look for an event and how that particular event will appear. They enable limited computational resources to perform effectively in real time by focusing the robot's attention on an expected event (Endo and Arkin, 2003). A robot can know the translational and rotational velocity of its body through a kinematic model. This can be used to extend our visual motion model in order to predict the perceived visual motion as the robot moves its hand or head.

Head Motion Prediction

We would like to find an affine matrix $\mathbf{A}_{ego}$ that describes the global optic-flow due to ego-motion of the robot's head. $\mathbf{A}_{ego}$ predicts the motion of a pixel between image $I_1$and

image I_2 as

$$\begin{bmatrix} u_2 \\ v_2 \end{bmatrix} = \mathbf{A}_{ego} \begin{bmatrix} u_1 \\ v_1 \\ 1 \end{bmatrix}.$$

This prediction is used to inform the block-matching process in the visual motion model. The block matching estimated the translation of each edge pixel by searching over an 11×11 pixel window centered at $[u_1, v_1]^T$ in I_2. Pixel translations greater than 5 pixels cannot be matched because they fall outside of the search window. This limits the effectiveness of the model during fast camera motion. However, using $\mathbf{A}_{ego}$, we can center the search window at the predicted location $[u_2, v_2]^T$ in I_2. As shown in Figure 6-2, this allows the visual motion model to be robust during the rapid head motions that are common for an expressive, social robot head.

To estimate $\mathbf{A}_{ego}$, we use proprioceptive feedback to compute ${}^{C2}\mathbf{T}^{C1}$, the transform describing the motion of the camera between image frames I_1 and I_2. This is simply

$${}^{C2}\mathbf{T}^{C1} = {}^{C2}\mathbf{T}^{W}\,{}^{W}\mathbf{T}^{C1},$$

where ${}^{C1}\mathbf{T}^{W}$ is the world-camera transform when I_1 was captured. If head motion is not present, then ${}^{C2}\mathbf{T}^{C1} = \mathbf{I}$.

A stationary point in the world, $\mathbf{x}^W$, viewed from $\{C\}$ over time has coordinates

$$\begin{bmatrix} x_1 \\ y_1 \\ z_1 \end{bmatrix} = {}^{C1}\mathbf{T}^{W}\,\mathbf{x}^{W},$$

and

$$\begin{bmatrix} x_2 \\ y_2 \\ z_2 \end{bmatrix} = {}^{C2}\mathbf{T}^{W}\,\mathbf{x}^{W}.$$

With a pinhole camera, this corresponds to pixels $\mathbf{k}_1 = \left[\frac{fx_1}{z_1}, \frac{fy_1}{z_1}\right]$ and $\mathbf{k}_2 = \left[\frac{fx_2}{z_2}, \frac{fy_2}{z_2}\right]$. We can now describe the image motion of a stationary 3D point, $\mathbf{x}^W$, viewed through a moving camera:

$$\begin{bmatrix} \frac{z_2}{f}\mathbf{k}_2 \\ z_2 \\ 1 \end{bmatrix} = {}^{C2}\mathbf{T}^{C1} \begin{bmatrix} \frac{z_1}{f}\mathbf{k}_1 \\ z_1 \\ 1 \end{bmatrix}. \tag{6.2}$$

We can make the approximation $z_2 \approx z_1$ (weak perspective camera constraint) if the difference in depth is small compared to the average depth. For head ego-motion, this is a fair assumption if we assume that the background is far from the camera (Kemp, 2005). A characteristic of humanoid ego-motion is that the predominant source of optic flow is from camera rotation and not translation. Domo is currently stationary, so ego-motion will not be induced by motion of the body[1]. Consequently, we ignore image motion resulting from camera translation, allowing us to reduce Equation 6.2 into our desired form. This gives

$$\begin{bmatrix} u_2 \\ v_2 \end{bmatrix} \approx \begin{bmatrix} r_1 & r_2 & r_3 f \\ r_4 & r_5 & r_6 f \end{bmatrix} \begin{bmatrix} u_1 \\ v_1 \\ 1 \end{bmatrix}, \tag{6.3}$$

where we use the upper-left 2×3 submatrix, $\mathbf{R}$, of ${}^{C2}\mathbf{T}^{C1}$.

Hand Motion Prediction

We now consider how to select for visual edges that correspond to the robot's moving hand. The visual motion model generates a weighted edge map, where fast moving edges in the foreground are weighted highly. However, given an affine model, $\mathbf{A}_{hand}$, of the expected hand motion in the image, we can adapt the visual motion model to select for the moving hand but ignore other objects moving in the foreground. As in the previous section, we use $\mathbf{A}_{hand}$ to center the block-matching window at the predicted location of an edge pixel. Consequently, pixels that move according to the $\mathbf{A}_{hand}$ model should fall within the 11×11 search window. Motion of other objects in the environment, such as a person, will be naturally discounted.

There is one subtle point to this. The motion model selects for foreground edges that differ from the predicted background motion. However, we now wish to select for edges that match the prediction of $\mathbf{A}_{hand}$. Consequently, we must invert the weight assigned to each edge. Within the motion model, each edge is weighted by the Mahalanobis distance, where a large distance indicates foreground motion. If $\mathbf{M}$ is the matrix of Mahalanobis distances for each edge, then we substitute $\mathbf{M}_{hand} = max(0, k - \mathbf{M})$ for $\mathbf{M}$ using a constant k.

It is straightforward to estimate $\mathbf{A}_{hand}$, which has the same form as in Equation 6.1. We select a point of interest, $\mathbf{x}^H$, in the hand frame $\{H\}$ such as a fingertip or the tip of

[1]For mobile platforms, the ego-motion estimate can be limited to periods when the body is stationary but the head is moving.

a grasped object. When viewed from camera frame $\{C\}$ at the sample times of images I_1 and I_2, this point has coordinates

$$\begin{bmatrix} x_1 \\ y_1 \\ z_1 \end{bmatrix} = {}^{C1}\mathbf{T}^H \, \mathbf{x}^H,$$

and

$$\begin{bmatrix} x_2 \\ y_2 \\ z_2 \end{bmatrix} = {}^{C2}\mathbf{T}^H \, \mathbf{x}^H,$$

corresponding to pixels $\mathbf{k}_1 = \left[\frac{fx_1}{z_1}, \frac{fy_1}{z_1}\right]$ and $\mathbf{k}_2 = \left[\frac{fx_2}{z_2}, \frac{fy_2}{z_2}\right]$. If $\mathbf{x}^H$ undergoes translation but not rotation, then

$$A_{hand} = \frac{1}{f} \begin{bmatrix} f & 0 & \frac{x_2}{z_2} - \frac{x_1}{z_1} \\ 0 & f & \frac{y_2}{z_2} - \frac{y_1}{z_1} \end{bmatrix}.$$

Typically, we are only interested in selecting for motion corresponding to the point $\mathbf{x}^H$ and not the motion of the entire hand. Consequently, it is sufficient to ignore the effects of rotation .

6.2 *InterestRegions*

We saw previously that the visual motion model selects for strong moving edges in the image. The regions around these edges will often correspond to important visual features. In order to incorporate visual information distributed near strong motion edges, we use a multi-scale interest point operator developed by Kemp (2005). This algorithm is embedded in the *InterestRegions* module. We review Kemp's algorithm here.

The interest point operator detects the position and scale of significant shape features within the image. Several different shape features can be detected, including circles, parallel lines, and corners. In order to meet the real-time constraints of our work, we have only used the circular shape feature. Therefore, an interest region is defined as a circular image patch at a given radius and location in the image. The other shape features would certainly expand the descriptive potential of *InterestRegions*, and it would be straightforward to parallelize their detection across multiple machines.

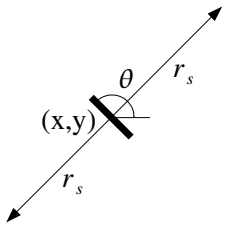


Figure 6-3: Voting by the interest point operator at scale s using radius r_s. The figure depicts the approximate locations in the image of the two votes cast by an edge with orientation θ and position (x, y) (Reproduced, with permission, from Kemp (2005)).

Traditional interest point methods, often characterized as blob-detectors, rely on constant contrast within an interest region. However, Kemp's method is an edge-based approach, making it compatible with the weighted edge map generated by our visual motion model. This algorithm has similarities to classic image processing techniques such as the distance transform, the medial axis transform, and the Hough transform for circles (Forsyth and Ponce, 2002).

The output of the interest point detector is a set of salient locations in the image and their associated radii, $[(u_1, v_1, r_1), (u_2, v_2, r_2), \ldots]$. The input to the detector is a set of weighted edges where each edge e_i has a weight, w_i, an image location, $\mathbf{x}_i$, and an angle, θ_i. This input is provided by our visual motion model.

A log-polar discretization is used over the space of feature scales and edge orientations. Kemp (2005) covers the discretization process in detail. Each scale space, s, corresponds to a circle of a given radius, r_s, in pixels. At each scale space, the edges each vote on two circle feature locations. As depicted in Figure 6-3, the two votes are approximately at distance r_s from the edge's location and are located in positions orthogonal to the edge's length. The angle θ_i denotes the direction of the edge's length and is in the range $[-\frac{\pi}{2}, \frac{\pi}{2}]$.

For each scale s there is a 2D histogram that accumulates the votes for interest points. The discretization of these histograms is determined by a integer bin length, l_s , which is a function of r_s such that larger scales spaces have appropriately lower histogram resolution.

The bin indices, (b_x, b_y), for the histogram at scale s are computed as

$$b_s(\mathbf{x}, \theta) = \text{round}(\frac{1}{l_s}(\mathbf{x} + r_s \begin{bmatrix} \cos(\theta + \frac{\pi}{2}) \\ \sin(\theta + \frac{\pi}{2}) \end{bmatrix})), \tag{6.4}$$

which adds a vector of length r_s to the edge position $\mathbf{x}$ and then scales and quantizes the result to find the appropriate bin in the histogram.

Now, the edges are iterated over. The two votes of edge e_i are weighted by w_i and added to the appropriate histogram bins. This results in the interest point detection maps, m_s. In order to soften the effects of the log-polar and histogram discretization, each 2D histogram, m_s, is low-pass filtered with a separable, truncated, FIR Gaussian. The response for each bin within m_s is ranked across all scales and the top 10 bins are selected as interest regions. These selections are then made available to the visual attention system.

The algorithm is suitable for real-time perception. When added to the motion model, the frame rate on a $3GHz$ Pentium is reduced from 25Hz to 17Hz. In our implementation using 160×120 images, we use 8 scale spaces ranging from 2 to 30 pixels, and the $[-\frac{\pi}{2}, \frac{\pi}{2}]$ range is discretized into 32 possible orientations. The algorithm can also be used without the motion model ($w_i = 1$) to detect circular edges within a single image. Sample output from the algorithm is shown in Figure 6-4.

6.3 The Sensory EgoSphere

The Sensory EgoSphere (SES) is short-term memory mechanism for a robot. The notion originated with Albus (1991) and was further developed by Peters et al. (2001). Using a mechanically identical head to Domo, Aryananda (2006) has used the SES in learning the spatio-temporal signatures of simple perceptual features. The SES, depicted in Figure 6-5, allows perceptual features that may be sensed in different modalities, coordinate frames, and at different times to be brought into a single, spherical coordinate frame centered on the robot. In doing this, the data is fused according to its spatio-temporal coincidence. The SES retains the spatial location of the data as the robot redirects its attention to other locations. For visual features, this involves transforming from pixel coordinates to spherical coordinates. If a visual feature is stationary in the world while the head is moving, then the feature will be stationary in the SES frame though not in the image frame. Thus, the SES

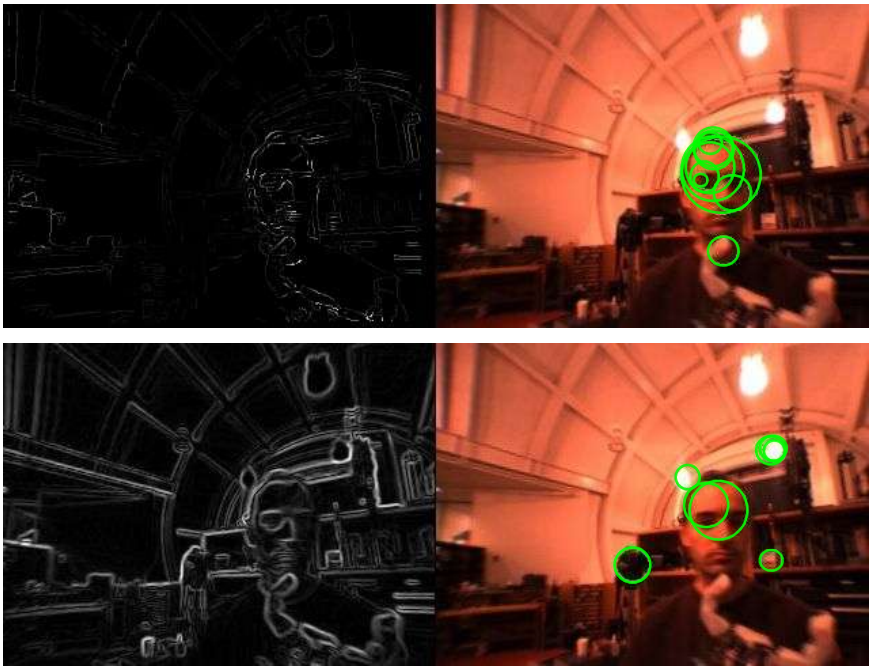


Figure 6-4: Output from the *InterestRegions* module. The motion weighted edge map (top, left) and unweighted edge map (bottom, left) are fed to the multi-scale interest point operator. The green circles indicate the location and size of circular interest regions. Weighting the edges by the foreground motion localizes the interest regions on salient features such as a person's head and the tip of the robot's finger (top, right).

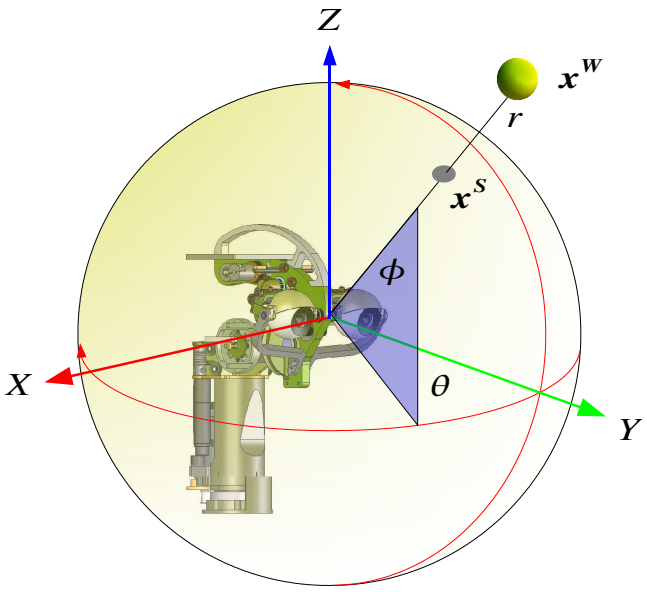


Figure 6-5: The Sensory EgoSphere (SES) transforms a point in the world, $\mathbf{x}^W$, into the stationary, spherical coordinate frame of the SES as the point $\mathbf{x}^S = [\theta, \phi, r]$. Pixel locations of unknown depth are projected onto a sphere of fixed radius of $r = 500$mm.

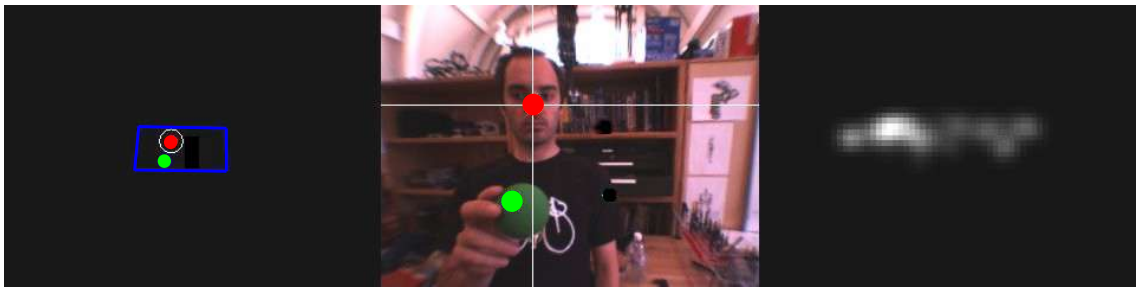


Figure 6-6: View of the Sensory EgoSphere (SES). Left**:** View of the SES showing the detection of a face (red) and a fiducial (green). The camera's field-of-view projected into the SES is shown in blue. Middle: The detected features in the image. Right: The spatial distribution of face detections over time within the SES.

provides a head-pose invariant frame of reference for perception, allowing for stable visual tracking when a tracked feature is detected intermittently or with high latency.

Domo uses a simplified version of Peters' SES formulation as its torso is stationary with respect to the world. A point in the SES frame $\{S\}$ has spherical coordinates $\mathbf{x}^S = [\theta, \phi, r]$, where θ is yaw in the transverse plane, ϕ is pitch in the sagittal plane, and r is the radius. The SES is defined in the $[-\frac{\pi}{2}, \frac{\pi}{2}]$ hemisphere with $[0, 0]$ pointing straight ahead of the body. As shown in Figure 4-1, we define $\{S\}$ as a translation, $\mathbf{t}^S_{Worg}$, of the world frame $\{W\}$ such that its origin is at the midpoint between the robot's eyes when it is looking straight ahead. Given this, we can project a point in the world $\mathbf{x}^W$ into SES by

$$\begin{aligned}
[x, y, z]^T &= \mathbf{x}^W - \mathbf{t}^S_{Worg} \\
r &= \left(x^2 + y^2 + z^2\right)^{\frac{1}{2}} \\
\theta &= \arctan\left(\frac{x}{z}\right) \\
\phi &= \arcsin\left(\frac{y}{r}\right) \\
\mathbf{x}^S &= [\theta, \phi, r].
\end{aligned}$$

In order to project an image pixel with unknown depth into the SES, we assume a fixed pixel depth of 500mm and then find its world coordinates $\mathbf{x}^W$. This assumes that the visual effects of head translation are small compared to rotation which is true for points far from the camera. We visualize Domo's SES in Figure 6-6.

6.3.1 Features in the SES

On Domo, the SES serves as a consolidation point for several types of features, including:

- A person's face
- A person's waving hand
- One of its hands
- The tip of a grasped object
- A randomly selected target
- The most likely location to see a face

- A colored fiducial

The perceptual algorithms for the non-trivial of these features will be described in coming chapters. Feature detections are added to the SES at rates from 10-30Hz. We low-pass filter a feature's location, allowing the head to smoothly track a feature even if it disappears momentarily or a false detection occurs. Because the SES is larger than the camera's field-of-view, we would like it to eventually forget out-of-view features. Each feature has a monostable timeout of 1s, after which it is removed from the SES unless a new detection has been received. At any time, the output from the SES is a single target in spherical coordinates, $\mathbf{x}^S$, which is used to direct the robot's gaze. This target is selected using a SLATE arbitrator and modules compete to direct the robot's gaze to a particular feature. For example, a *WatchHand* module can direct the gaze to the robot's hand if it has the highest dynamic priority of all writers to the SES arbitrator.

6.3.2 Spatial Distributions in the SES

We can accumulated evidence over time of a feature's spatial distribution within the SES. Some features will appear in predictable regions. For example, Figure 6-6 visualizes the spatial distribution of face detections over many hours. As we will see in Chapter 8, such prior information can allow the visual attention system to ignore unlikely feature detections. Torralba (2003) describes a framework that uses similar information to improve visual search and modulate the saliency of image regions.

We can model the spatial distribution of an image feature as the probability distribution $p(\mathbf{x}^S|f)$. This represents the chance of seeing feature f at location $\mathbf{x}^S = [\theta, \phi, r]$. The distribution is estimated using a 2D histogram over $[\theta, \phi]$. Each dimension of the histogram maps to a $[-\frac{\pi}{2}, \frac{\pi}{2}]$ range of the corresponding dimension of of the SES. The histogram used is 100×100 corresponding to a block size of 30×30 pixels in the image. The distribution is updated whenever the feature appears and it is saved to disk when the robot is not running, allowing the robot to estimate the distribution over a long period of time.

We estimate $p(\mathbf{x}^S|f)$ for visual feature, f, as

$$p(\mathbf{x}^S|f) \approx \frac{1}{\sum_{\mathbf{x} \in X_f} w(\mathbf{x})} \sum_{\mathbf{x} \in X_f} w(\mathbf{x}) \delta(\text{round}(\frac{100}{\pi}(\mathbf{x} - \mathbf{x}^S))), \tag{6.5}$$

where

$$\delta(d) = \begin{cases} 1 \text{ if } d = 0 \\ \\ 0 \text{ otherwise} \end{cases}$$

and X_f is the accumulated detections of f in the SES.

We use a weighting function $w(\mathbf{x})$ that reduces the influence of older detections. When a new detection, $\mathbf{x}_i$, is added to the distribution, its weight is initialized at $w(\mathbf{x}_i) = 1.0$. This value is decremented at a rate of Δw_f per second. Detection $\mathbf{x}_i$ is removed from X_f if $w(\mathbf{x}_i) < 0.01$. This allows the distribution to adapt to recent changes such as an object changing location.

CHAPTER 7

Let the Body do the Thinking

In this chapter we investigate how the robot's physical embodiment can be leveraged to simplify manipulation tasks. We describe several SLATE modules, highlighted in Figure 1-3, that illustrate this theme. *StiffnessAdapt* allows a module to control the manipulator stiffness during a task. *ContactDetect* is triggered when manipulator contact is made with the world. The *GraspAperture* module estimates the diameter of a grasp given the proprioceptive state of the hand. These modules are then combined into the *SurfaceTest* and *SurfacePlace* modules. *SurfaceTest* allows Domo to reach out and verify the uncertain location of a hypothesized flat surface. *SurfacePlace* exploits the robot's compliance to place unknown objects upright on a surface.

7.1 *StiffnessAdapt*

Despite its simplicity, the *StiffnessAdapt* module can be an effective tool for dealing with perceptual uncertainty. By lowering the stiffness of the arm, *StiffnessAdapt* trades off precise position control for compliance. Lowered arm stiffness is advantageous when contact with an unknown surface is anticipated. It allows the hand to maintain contact and adapt its posture without generating dangerous restoration forces. Typically, actuator saturation prevents a manipulator from responding with low stiffness during unexpected contact (Lawrence, 1989).

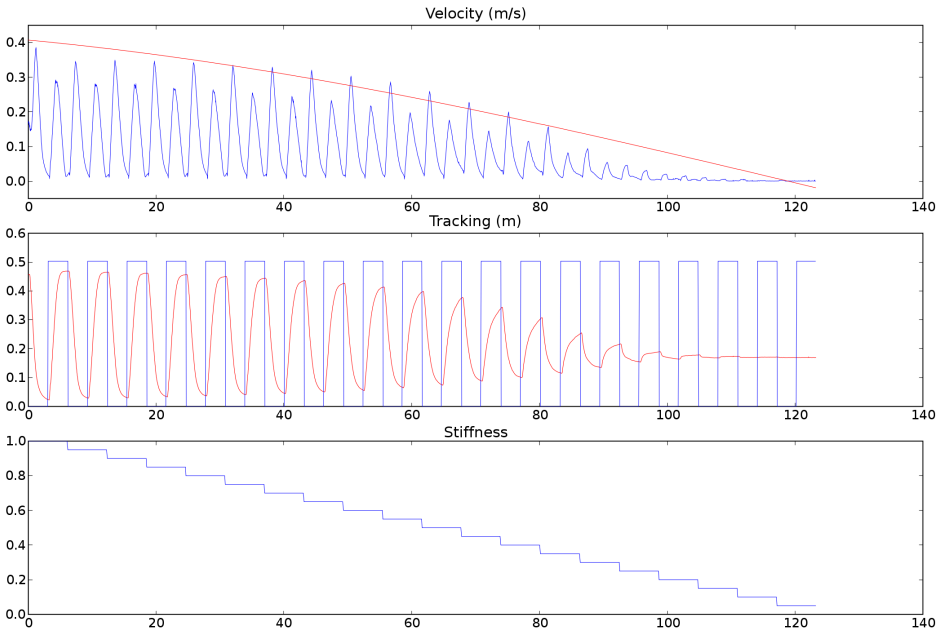


Figure 7-1: The effect of manipulator stiffness on controller response. The hand was servoed between two targets 0.5 meters apart every 3 seconds. (Bottom) The arm stiffness was ramped from $K_{ps} = 1.0$ to $K_{ps} = 0.0$. (Middle) Blue indicates the distance of the desired hand location from the first target. Red indicates the hand's distance from this target. (Top) Blue indicates the magnitude of the hand's velocity. The red line illustrates the learned decision boundary, $g_v(K_{ps})$, for the *ContactDetect* module described elsewhere.

However, the springs in Domo's actuators allow the manipulator to exhibit a low stiffness even above the control bandwidth of the actuators.

The stiffness for each joint of the arms and hands is specified by the controller parameter $0 \leq K_{ps} \leq 1$ from Equation 4.4. This DSP controller essentially simulates a virtual torsion spring at the joint with spring stiffness K_{ps}. Therefore, the stiffness is controlled in joint space and not Cartesian space. Figure 7-1 shows the effect of varying K_{ps} on the controller response. As stiffness is lowered, the manipulator position control performance degrades. However, an integral term in the controller, as well as secondary control loops such as visual servoing, can be used to improve the positional accuracy. Functionally, *StiffnessAdapt* simply provides arbitration among competing modules that require a particular arm stiffness. The request with the highest priority, as described in Section 5.2, is then transmitted to the DSP controller. Ideally, a module would learn a desired arm stiffness for a task, or use sensory feedback to adapt the stiffness. In our work, the joint stiffness requested by a module is determined experimentally.

7.2 *ContactDetect*

We would like to use the robot's proprioceptive state to detect when the manipulator makes contact with the world. Huber and Grupen (1994) have demonstrated a single detector that fuses joint torques, positions, and velocities to localize the contact location on a finger. Because of Domo's relatively coarse proprioceptive knowledge, we adopt a simplified approach that detects contact but does not localize the contact point. Our *ContactDetect* module partitions the perceptual task into two separate detectors. In the first, low manipulator stiffness is used to transform contact forces in to detectable motion at the hand. In the second, high manipulator stiffness allows contact forces to generate detectable errors in a dynamic model. *ContactDetect* determines which detector to use based on the manipulator stiffness commanded by *StiffnessAdapt*. Figure 7-2 illustrates the *ContactDetect* module on Domo.

7.2.1 Contact Motion

This method of contact detection simply monitors the stiffness of the manipulator and the velocity of its perturbed hand. When the arm has low stiffness, we expect that disturbance

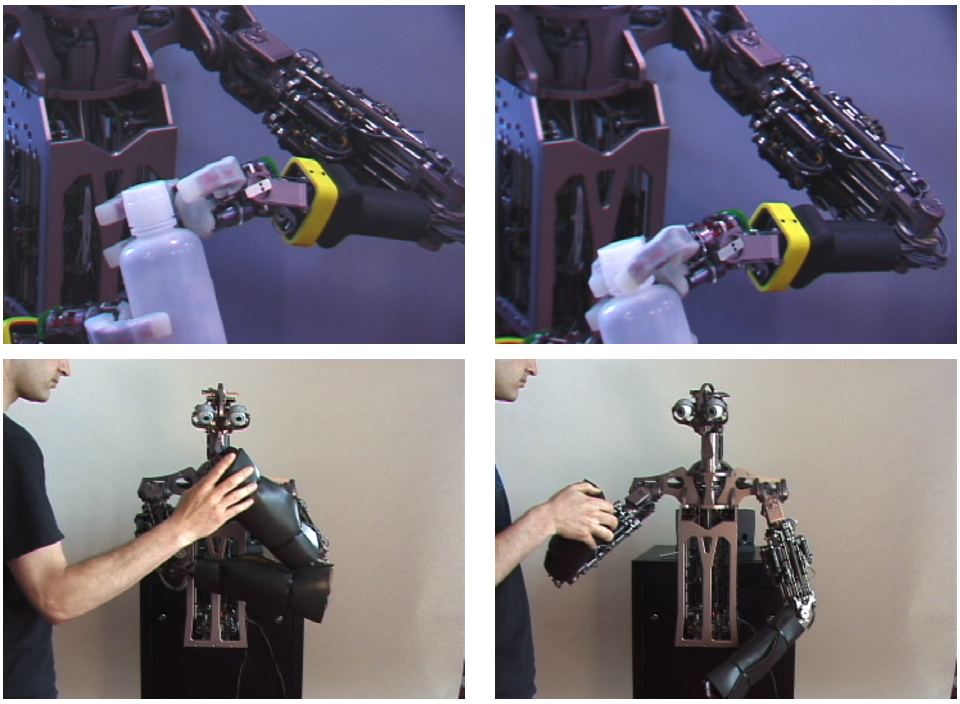


Figure 7-2: Detecting manipulator contact by using low stiffness and by using a dynamic model. (Top) A bottle is held by the right hand and the right arm has low stiffness. When contact forces generated by the left hand cause the right hand to to move, *ContactDetect* triggers a grasp reflex. (Bottom) When the arm has high stiffness, torques generated by human contact violate the prediction of a dynamic model. This triggers the *WatchHand* module to direct the eye gaze to the robot's hand.

forces will cause unexpected hand motion. As shown in Figure 7-1, the maximum velocity of the hand decreases with the manipulator stiffness as it tracks a target. The stiffness is defined by the controller gain K_{ps} while the magnitude of the instantaneous hand velocity is $\left\|\mathbf{J}\dot{\mathbf{\Theta}}\right\|$, where the Jacobian $\mathbf{J}$ converts joint rates to a Cartesian velocity at the hand. We define $v_{max} = max(\left\|\mathbf{J}\dot{\mathbf{\Theta}}\right\|)$ as the maximum velocity of the hand, over time, as it tracks a target.

In order to detect contact, we would like to learn a function $g_v(K_{ps})$ such that $g_v(K_{ps}) \approx v_{max}$. We used the LIBSVM package (Chang and Lin, 2001) for support vector regression (SVR) to learn $g_v(\cdot)$ offline using a Gaussian RBF kernel. First, the hand was servoed between two targets 0.5 meters apart every 3 seconds for two minutes. Simultaneously, the arm stiffness was ramped from $K_{ps} = 1.0$ to $K_{ps} = 0.0$. For each value of K_{ps}, we collected v_{max}. The SVR was then trained on each pairing of K_{ps} and v_{max}. The top row of Figure 7-1 plots $g_v(K_{ps})$ versus K_{ps}.

ContactDetect signals that external contact has been made when $\left\|\mathbf{J}\dot{\mathbf{\Theta}}\right\| - g_v(K_{ps})$ is above a threshold. This method is best suited for when the manipulator exhibits low stiffness and the disturbance velocity of the hand is large given contact. It is susceptible to false-positives when the stiffness is high. As a consequence, *ContactDetect* ignores detections when $K_{ps} > 0.5$.

7.2.2 Contact Forces

The joint-space form of manipulator dynamics is

$$\tau_{dyn} = \mathbf{M}(\mathbf{\Theta})\ddot{\mathbf{\Theta}} + \mathbf{V}(\mathbf{\Theta}, \dot{\mathbf{\Theta}}) + \mathbf{G}(\mathbf{\Theta}),$$

where $\mathbf{M}(\mathbf{\Theta})\ddot{\mathbf{\Theta}}$ is the torque due to mass accelerations, $\mathbf{V}(\mathbf{\Theta}, \dot{\mathbf{\Theta}})$ is the centrifugal and Coriolis torque, and $\mathbf{G}(\mathbf{\Theta})$ is the torque due to gravity (Craig, 1989). Given this model, the error between the predicted joint torques, τ_{dyn}, and the sensed torques, τ_{sense}, can be used to detect reaction forces between the manipulator and the world.

A complete model requires estimating the mass distribution of the arms. In practice, dynamic models can be difficult to obtain and calibrate. In addition, the joint acceleration is difficult to measure precisely due to sensor resolution and anti-aliasing errors. However, the presence of external contact forces can be detected using some common model simplifications. We assume that $\mathbf{V}(\mathbf{\Theta}, \dot{\mathbf{\Theta}}) = 0$ and approximate the inertia tensor of $\mathbf{M}(\mathbf{\Theta})$ using a

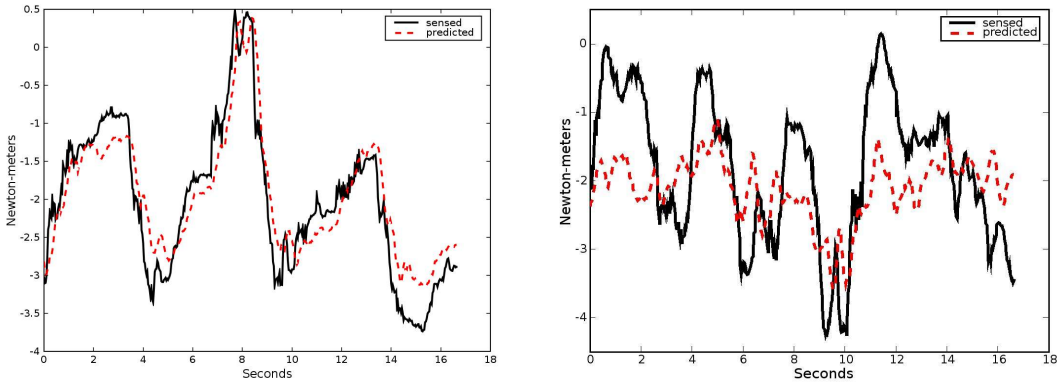


Figure 7-3: (Left) The sensed torque (black) and the predicted torque (red) for the shoulder pitch joint during non-contact reaching. (Right) The same measurement while a person makes contact with the arm during reaching.

point mass for the robot forearm and bicep. Using the recursive Newton-Euler formulation, we can predict the joint torques as $\tau_{dyn} = \mathbf{M}(\mathbf{\Theta})\ddot{\mathbf{\Theta}} + \mathbf{G}(\mathbf{\Theta})$ (Gourdeau, 2005).

The model error, defined as $\tau_{dyn} - \tau_{sense}$, is used to signal contact. As shown in Figure 7-3, the error is large when manipulator contact is made. However, during dynamic reaching, errors can also result from our model simplifications. Therefore, we distinguish between errors induced by contact and those induced by unmodeled dynamics.

This is done using supervised, offline learning of an error histogram for each joint given each type of error. Each of the 8 histograms (4 DOF, 2 categories) are computed using data collected during reaching movements sampled across the manipulator's workspace. Two trials were conducted. During one, the arm reached freely to its target. During the other, a person applied contact disturbances to the arm as it reached. During each trial, the error $\tau_{dyn} - \tau_{sense}$ (Nm) was sampled at 10Hz and labelled according to the trial. Figure 7-4 shows these histograms. We see that contact errors are largest for the first two joints (pitch,yaw) of the shoulder. For the other joints, the two error types are difficult to distinguish. Consequently, only the shoulder joints are used signal contact. Contact is signaled when the magnitude of the prediction error is above a threshold and it is unlikely that the error is due to reaching dynamics (as measured by the histograms).

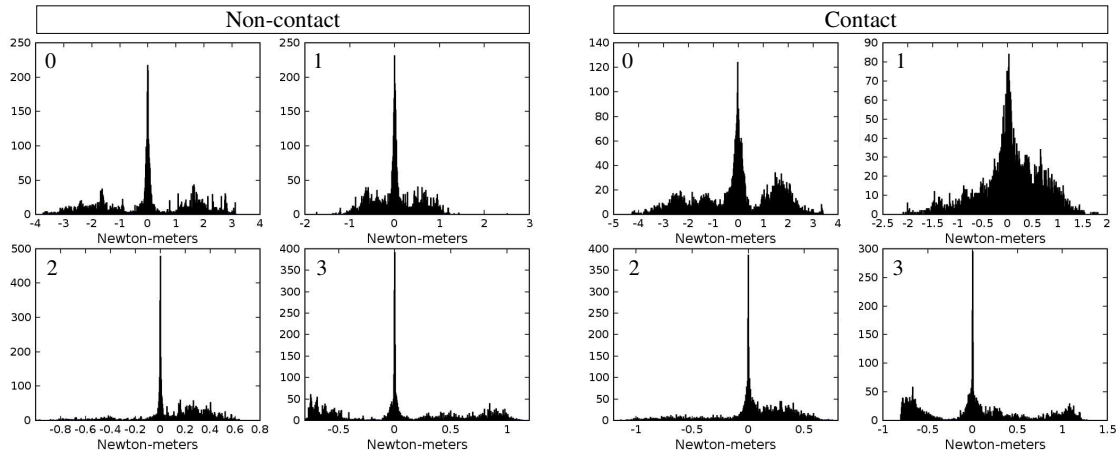


Figure 7-4: Error histograms for the torque prediction of the first four joints. The arm executed reaching movements sampled across its workspace. At 10Hz the error (Nm) between the predicted and sensed joint torque was measured. Error distributions were measured for both non-contact reaching (left) and when contact disturbances were applied by a person during reaching (right). During normal operation, contact is signaled for a joint when the measured error is unlikely given the non-contact distribution. Contact errors are largest for the first two joints (pitch,yaw) of the shoulder.

7.3 *GraspAperture* and *GraspDetect*

The grasp aperture, typically defined as the distance between the thumb and forefinger, is a common measure used when studying human manipulation. Prior to grasping an object, a person's grasp aperture varies according to the object being grasped and their perceptual uncertainty about the object (Mon-Williams and Tresilian, 2001). On a robot, the grasp aperture can be used to estimate the size of an unknown, grasped object. For example, the grasp aperture created by a power grasp on a cylinder is proportional to the cylinder diameter. Ideally, the grasp aperture is measured directly using a kinematic model of the robot's hand. However this can be difficult due to non-ideal joint sensing, unmodeled kinematics, and compliant effects in the robot's finger and skin. Also, the distance between the thumb and forefinger is not always a good measure of object size as this distance can vary depending on the grasp. For some grasps, these digits may not even make contact with the object.

On Domo, there is substantial compliance in the finger. As shown in Figure 4-10, the compliant fingertip and skin allow the finger surface to deform during grasping. However, this deformation cannot be measured directly as it occurs between the joint angle sensor and the object. Consequently, it would be difficult to measure the grasp aperture kinematically without modeling the complex effects of the compliance.

Instead of building a detailed hand model, we used support vector regression (SVR) (Chang and Lin, 2001) for supervised, offline learning of the map between the hand's joint angles and the grasp aperture. Training data was gathered for 50 trial power grasps formed on five cylindrical objects of known diameters between 25mm and 75mm. Because the power grasp is force controlled, we can manually apply disturbances to the object and cause displacement from its equilibrium pose. For limited displacements, the fingers will maintain contact with the object. In this way, we sampled the range of joint postures that result in a stable grasp for each object. As each object was displaced, the four joint angles of the hand were recorded and labelled with the object's diameter. Using the data from all trials, the function $g_a(\mathbf{\Theta})$ was learned using SVR and a Gaussian RBF kernel. $g_a(\mathbf{\Theta})$ predicts the diameter of a grasped cylinder given the hand configuration $\mathbf{\Theta}$.

In Figure 7-5 we show the predicted grasp aperture for test objects of different diameters from the training set. For each of the six test cylinders, the object was grasped and manually

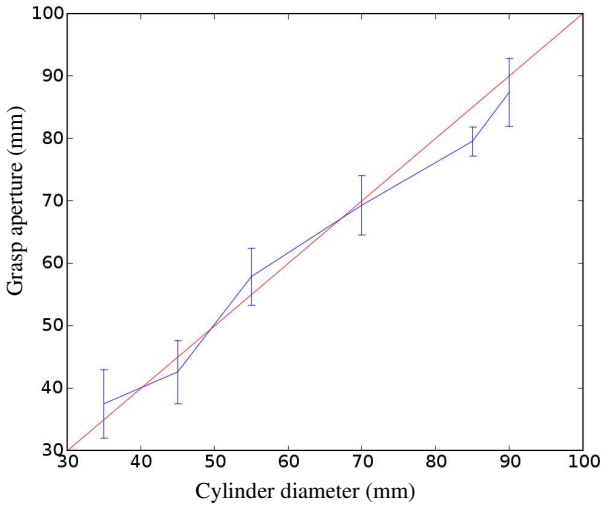


Figure 7-5: The predicted grasp aperture $g_a(\mathbf{\Theta})$ (Y axis) versus the diameter of a known, grasped cylinder (X axis). For each of six test cylinders, the object was grasped using force control. The object was then manually moved about the full grasp workspace that permitted all three fingers to remain in contact. The error bars show the maximum extent of the predicted aperture around its mean.

moved about the set of postures that permitted all three fingers to remain in contact. We see that $g_a(\mathbf{\Theta})$ can predict the cylinder size within 10mm across the set of stable grasps. In fact, the natural resting grasp posture does even better. As the object is moved far from this pose, the performance degrades. The *GraspAperture* module computes $g_a(\mathbf{\Theta})$ in real-time during task execution. As we will see, modules such as *SurfacePlace* can use the grasp aperture feature to adapt their behavior.

Related to *GraspAperature* is *GraspDetect,* which signals that a stable grasp has been made on an object. As in the work of Connell (1989), *GraspDetect* is informed by the sensory state of the hand in the world rather than the internal state of the grasp controller. It relies on three conditions to detect a grasp. First, it monitors the net torque applied by the fingers. If it is positive (closing) and above a threshold, then it is assumed that the controller is forming a grasp. Second, if the net angular velocity of the fingers is close to zero, it is assumed that the fingers are in a stable state. Third, if $g_a(\mathbf{\Theta}) > 20$mm, then it is assumed that fingers are wrapped around an object and not resting on the palm of the hand. If all three conditions are true, then *GraspDetect* signals a stable grasp.

7.4 *SurfaceTest*

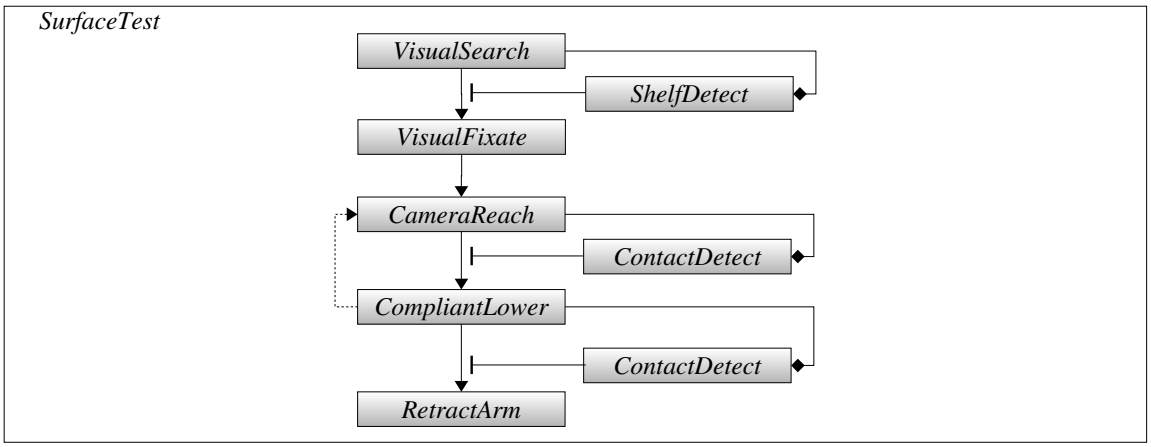
In unstructured environments, it can be difficult to use vision alone to detect with certainty the presence of an important feature. Fortunately, a robot can use its body to actively test a hypothetical feature location to learn more. In this vein, the *SurfaceTest* module uses the robot's arm to verify the existence and location of a reachable surface. People exhibit a similar behavior when placing an object on a surface in the dark. They will often first reach out and touch the surface. This action serves to reduce their perceptual uncertainty and confirm their hypothesis of where the surface is.

Our implementation of *SurfaceTest* is specialized for a shelf surface that has a visible leading edge, though we are not necessarily restricted to these surfaces. The shelf edge is a common, task relevant feature in domestic settings. In terms of the generic manual skill algorithm of Section 5.3.3, *SurfaceTest* is a compensatory action. The *SurfaceTest* algorithm is described in Figure 7-6.

To begin, *SurfaceTest* identifies a shelf edge through modules *VisualSearch* and *ShelfDetect*. *VisualSearch* causes the robot's gaze to scan a room until *ShelfDetect* is signaled. *ShelfDetect* uses an HSV color filter and blob detector to detect two green stickers marking the edge of a shelf. This simple detector is not especially robust as there are often other green objects in Domo's office environment. Also, the apparent color of an object changes as the lighting changes throughout the day. Fortunately, *SurfaceTest* acts to reduce this perceptual uncertainty. *ShelfDetect* could be readily replaced by a more sophisticated technique using stereo information or surface texture, allowing *SurfaceTest* to work on a variety of surfaces without markers.

Once a shelf edge has been identified, *SurfacePlace* first defines an image target at a fixed height above the edge midpoint, corresponding to a point $\mathbf{x}^S$ in the ego-sphere. Using *VisualFixate*, the head servos $\mathbf{x}^S$ into the center of the image. Once the target is centered, the camera is held fixed so that visual occlusions of the shelf do not effect the arm controller. The depth of the shelf edge is unknown, so *CameraReach* now reaches with an arm in depth along the camera's optical axis until contact is made. First, the arm closest to $\mathbf{x}^S$ is selected and its hand is brought to a defined location $[0, 0, z_{start}]^T$ in the camera frame $\{C\}$. Using the inverse kinematic model, the hand is extended along the optical axis towards $[0, 0, z_{end}]^T$.

We choose z_{start} and z_{end} so the arm starts close in to the body and ends at an arms



Ready Wait until *VisualSearch*, *ShelfDetect*, and *VisualFixate* find and center a candidate shelf edge in the image.

Precondition Use *CameraReach* to extend the arm along a ray from the camera to a fixed depth above the shelf edge.

Precondition If premature contact with the shelf is signaled by *ContactDetect*, reduce the reach depth.

Control Use the compliant force control of *CompliantLower* to move the hand down and make contact with the surface.

Success If *ContactDetect* signals contact with the shelf, record the posture, prior to descent, that led to success. Allow *RetractArm* to bring the arm back down to the robot's side.

Figure 7-6: The *SurfaceTest* module uses the robot's body to verify the location of a reachable surface.

length from the head. Often, the reach stops short of z_{end} due to premature contact between the shelf edge and the forearm. The arm can safely stop short of its target because its stiffness is kept low. If *ContactDetect* is signaled prior to reaching z_{end}, the arm posture is adjusted by retracting a fixed distance from the contact posture. At this point we assume that the hand is positioned just above the front edge of the shelf. This is typically true, though a richer search strategy could be implemented based on the contact forces sensed at the hand.

Next, the *CompliantLowe*r module brings the hand to rest on the surface. It uses the controller of Equation 4.4 to generate a downward force f_z at the hand such that $K_{ps} = 0$ and $\tau_{bias} = \mathbf{J^T}[0, 0, -f_z, 0, 0, 0]^T$. If a large hand displacement is detected, *SurfaceTest* assumes that either the shelf is out of reach or the feature detection was erroneous. Otherwise, *ContactDetect* detects the surface contact and the joint posture of the arm prior to *CompliantLower* is recorded. In this way, *SurfaceTest's* internal model of a surface is simply the joint angles that bring the hand above the surface. Other modules can now use this direct representation to easily control the arm relative to the surface.

7.4.1 Results

In total, we have run *SurfaceTest* on Domo for approximately one hundred trials. In one experiment, we tested *SurfaceTest* for 15 consecutive trials. For each trial the shelf was moved to an arbitrary location near the robot. The shelf height was varied between 50 mm and -150 mm relative to the shoulder. Some locations were deliberately out of reach of the robot, and some were too close in. A trial was successful if the robot robot's hand came to rest on the front edge of the shelf and *SurfaceTest* correctly verified that the shelf could be reached. The module was successful for all 15 trials, and the results of five of the trials are shown in Figure 7-7. The robot is shown executing the module in Figure 7-8. Although the algorithm is shown to be robust to the shelf location and height, the arm can get trapped under the shelf if it is too close to the body. In this case, coarse knowledge of the surface depth would be useful in guiding *CameraReach*.

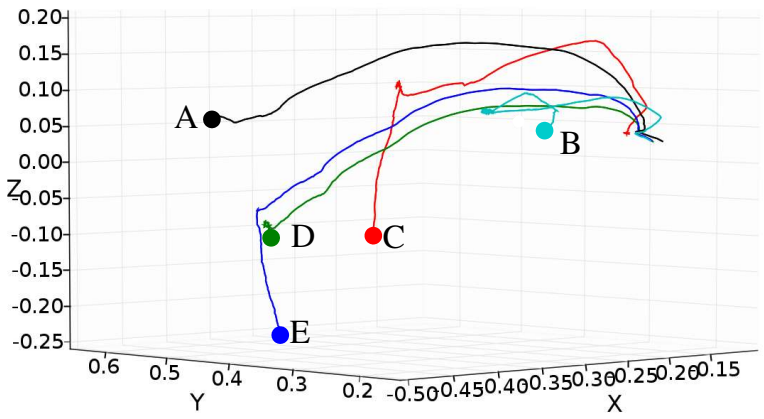


Figure 7-7: The reaching trajectory, in meters, of the hand during five consecutive trials of *SurfaceTest*. The final pose of the hand is annotated. The shelf surface is moved to five arbitrary locations in the robot's workspace and the shelf height varied. In trials A, B, and C, the shelf height is at 50mm relative to the shoulder. In trials D and E, the height is at -150mm . For trials A, B, and D, Domo successfully rests its hand on the front edge of the shelf. In trials C and E, Domo correctly detects that the shelf is out of reach. For trial B, the shelf is much closer in and the arm contacts the shelf prematurely. This is detected and the posture is adjusted to successfully find the shelf edge.

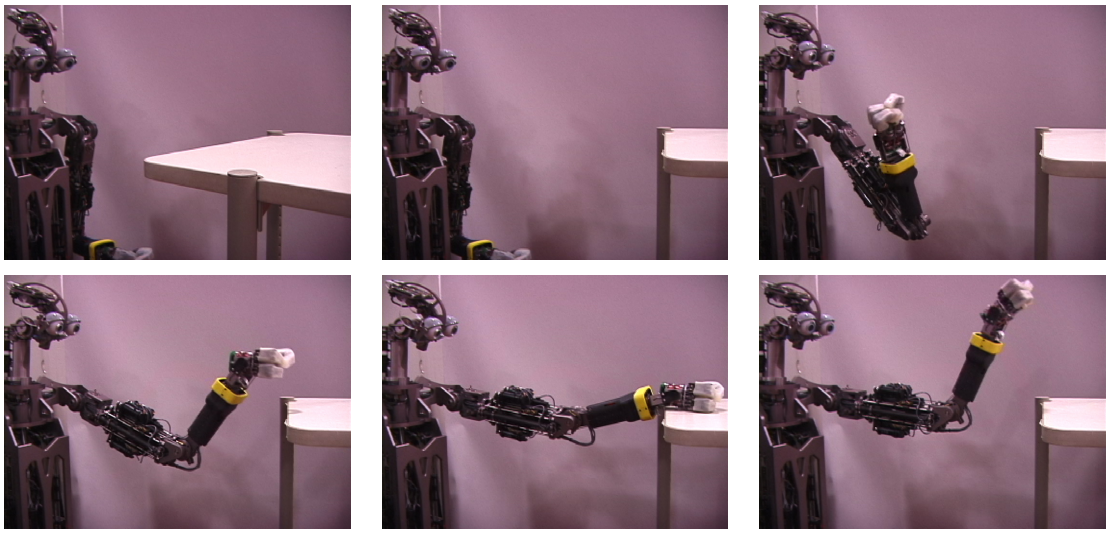
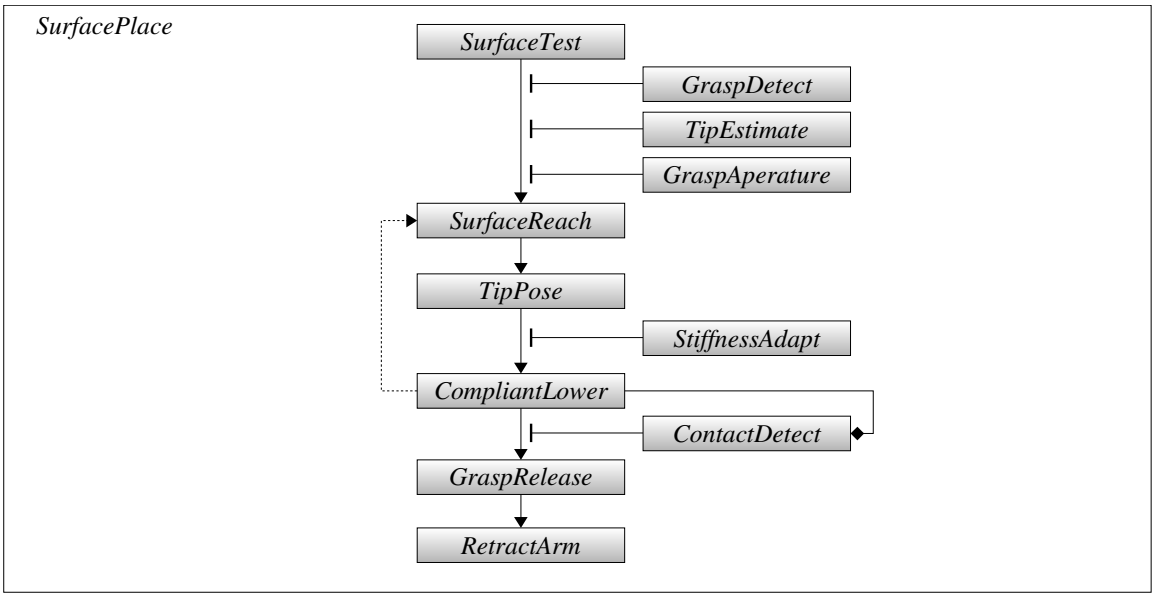


Figure 7-8: Domo's *SurfaceTest* behavior verifies the presence and location of a shelf. In this demonstration, the shelf location is first moved and then *SurfaceTest* reestimates the location.



Ready Wait until *SurfaceTest* detects a useable surface, *GraspDetect* signals a grasp on an object, and *TipEstimate* find the alignment axis of the object.

Precondition Estimate if the object is to be placed upright or lying down. Use *SurfaceReach* and *TipPose* to reach to above the surface and align the object to the surface.

Compensatory Use *StiffnessAdapt* to decrease the wrist stiffness, allowing the object to self-align during contact.

Control Use *CompliantLower* to lower the object on to the surface.

Success When *ContactDetect* is signaled, release the grasp and relinquish control to *RetractArm*.

Figure 7-9: The *SurfacePlace* module transfers an object from the robot's hand to a surface.

7.5 *SurfacePlace*

Human environments are full of flat surfaces, and many useful tasks involve placing objects on surfaces. Stocking goods, setting the table, arranging a product display, placing a part onto a conveyor belt, and putting away dishes are just a few of the everyday tasks that require surface placement. Fortunately, many man-made objects are designed so their intrinsic mechanics assist placement in a stable, canonical orientation. For example, a wine glass has a wide base allowing it to remain upright on a shelf. We would expect a pencil to rest on its side. There are exceptions of course. Books tend to lay on their sides, but convention dictates that they rest upright on a bookshelf.

In this vein, the *SurfacePlace* module takes a grasped object and places it on a nearby flat surface in one of two canonical orientations: upright or lying down. The orientation is chosen based on an estimate of the object's stability. The *SurfacePlace* algorithm is described in Figure 7-9 in terms of the generic manual skill algorithm of Section 5.3.3.

This algorithm requires a few assumptions about the object's shape in order to align its base to the surface. First, we define the object's alignment axis as $\mathbf{a} = \mathbf{x}_t^H - \mathbf{p}_t^H$, where $\mathbf{x}_t^H$ is the furthest tip of the object from the wrist's center of rotation and $\mathbf{p}_t^H$ is the center of the palm (See Figure 9-8 for an illustration). We assume that the grasped object's base is aligned when $\mathbf{a}$ is normal to the surface. This assumption is true for a variety of everyday objects such as bottles, books, and many hand tools. We will show how $\mathbf{x}_t^H$ is estimated by the *TipEstimate* and *TipPrior* modules in Section 9.1. The alignment axis can be controlled relative to the surface using the *TipPose* module of Section 9.2.

Once the alignment axis is detected and *SurfaceTest* has located a useable surface, *SurfaceReach* positions the arm just above the shelf surface. It then estimates the object's placement stability using both vision and haptics. We define the stability, η, as the ratio between the size of its base and its length. Assuming that the base size is well characterized by the grasp aperture, this is measured as $\eta = \frac{g_a(\mathbf{\Theta})}{\|\mathbf{a}\|}$. An object with $\eta < 0.25$ or $g_a(\mathbf{\Theta}) <$ 30mm is first rotated so $\mathbf{a}$ is parallel to the surface (lying down) because it would likely fall over if placed upright. Once the object has been aligned, *CompliantLower* drops the arm stiffness and brings the object down onto the shelf surface using force control. If *ContactDetect* is signaled, the grasp on the object is released and the arm retracted.

An important aspect of *SurfacePlace* is that it exploits compliance in the wrist and hand,

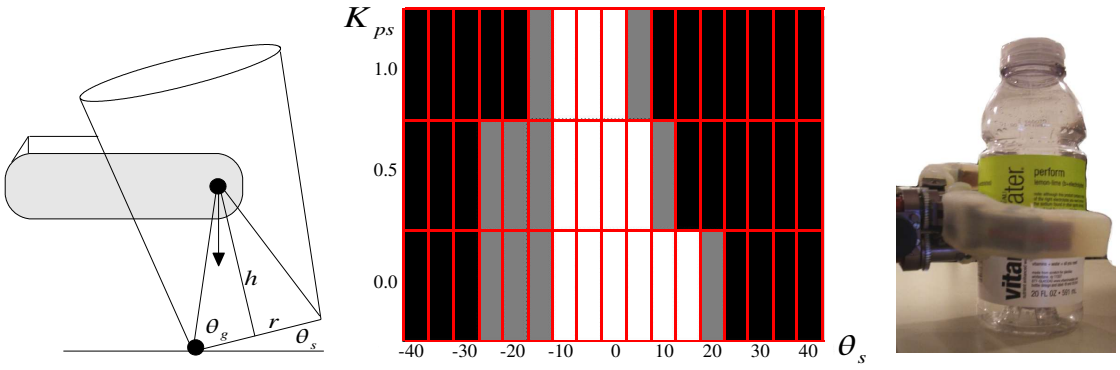


Figure 7-10: *SurfacePlace* uses contact with a surface to passively align the base of an object to the surface. (Left) A downward force applied to the object creates a realignment moment about the contact point. Ignoring the object's mass and friction, a lower grasp or a wider base will increase the robustness of this strategy. We can expect success when $\theta_s + \theta_g < \frac{\pi}{2}$ for $\theta_g = tan^{-1}\frac{h}{r}$. (Middle) A 2D histogram showing the number of successful trials (black=0, grey=1, white=2) in placing a bottle upright on a shelf as the wrist stiffness, K_{ps}, and the misalignment angle, θ_s, are varied. We tested $-40° \leq \theta_s \leq 40°$ in 5 degree increments and wrist stiffness $K_{ps} \in [0.0, 0.5, 1.0]$. As the stiffness is lowered, the tolerance to misalignment improves. (Right) The bottle used in this experiment.

as well as contact with the surface, to align the object's base to the surface. In this way, Domo can achieve a goal orientation despite uncertainty in the pose of the grasped object. This type of strategy has a long history in manipulation, where the intrinsic mechanics of an object in contact with the environment are used to reduce uncertainty. Notable examples include Mason's analysis of manipulation funnels (Mason, 1985), Inoue's use of force sensing for peg-in-hole tasks (Inoue, 1979), and the development of the remote-center-compliance (RCC) wrist for passive alignment of objects (Whitney and Nevins, 1979).

7.5.1 Results

SurfacePlace relies upon the manipulator compliance to assist in realignment of an object while placing it. As shown in Figure 7-10, we measured the success of *SurfacePlace* in placing a bottle upright on a shelf as the wrist stiffness, K_{ps}, and misalignment angle, θ_s, are varied. We define θ_s as the angle between the surface plane and the object's base. As Figure 7-10

illustrates, in the ideal case the expected misalignment tolerance is $\theta_s < \frac{\pi}{2} - tan^{-1}\frac{h}{r}$ for base radius r and grasp height h. We can deduce that *SurfacePlace* should be successful when $|\theta_s| < 21.2^\circ$ for the object used in this experiment, where $r = 35$mm and $h \approx 90$mm.

During the experiment, the shelf location was fixed and the bottle was repeatedly handed to the robot in a near identical pose. When grasped, the bottle's base is approximately parallel to the bottom of the hand, allowing for θ_s to be measured kinematically. We tested $-40^\circ \leq \theta_s \leq 40^\circ$ in 5 degree increments and wrist stiffness $K_{ps} \in [0.0, 0.5, 1.0]$. We conducted two trials of each of the 51 parameter combinations. The force at the hand was approximately $[0, 0, -4]^T$ Newtons. A trial was successful if the bottle was left standing upright on the shelf. The experiment results are shown in Figure 7-10 as a 2D histogram. They demonstrate that the misalignment tolerance improves as the wrist stiffness is lowered. For $K_{ps} = 0$, the tolerance is roughly ± 20 degrees as predicted.

In practice, many other factors complicate the success of *SurfacePlace.* Unstable objects can be disturbed as the hand is withdrawn. The alignment is only as good as the perception, and the object assumptions do not always match reality. In the absence of additional perceptual information, it seems that the best strategy is to keep the wrist as compliant as possible. This strategy is appropriate so long as the alignment is within the acceptable tolerance. However, if feedback from force, visual, or tactile sensing is integrated, then the stiffness could be increased in order to actively control the wrist.

In a second experiment, we tested *SurfacePlace* on the wide range of everyday objects shown in Figure 7-11. These objects include a stuffed animal, shoe box, large water bottle, headphone caddy, spoon, paint roller, hand broom, food tin, small water bottle, duster, and a spray bottle. In Figure 7-12, we show the robot's estimate of the object stability, η, versus the hand measured value for each of these objects. Most of the estimates fall within 10% of the hand measured values. However, the duster's long length causes it to leave the camera's field-of-view, which increased the error in its estimate for η. Likewise, the shoe box and headphone caddy have larger errors because their shapes violate the assumptions of *SurfacePlace*. The broom, duster, and spoon were correctly placed on their sides due to their low stability. The paint roller, with $\eta = 0.26$, is placed either upright or lying down. It is the most difficult to consistently place upright due to its small, 40mm diameter base. If better resolution force sensing were available at Domo's wrist, then the object's mass distribution could also be estimated to improve the robustness of the stability measurement.

Figure 7-11: A variety of everyday objects were successfully tested with the *SurfacePlace* module, including (left-right): a stuffed animal, shoe box, large water bottle, headphone caddy, spoon, paint roller, hand broom, food tin, small water bottle, duster, and a spray bottle.

Object	η measured	η sensed
animal	0.48	0.50
box	1.15	0.82
large bottle	0.56	0.54
caddy	0.61	0.49
spoon	0.21	0.24
roller	0.29	0.26
broom	0.24	0.23
food tin	1.20	0.98
small bottle	0.57	0.49
duster	0.13	0.24
spray bottle	.36	.42

Figure 7-12: *SurfacePlace* estimated the placement stability, η, for a variety of objects. η is sensed by the robot as the ratio between its grasp aperture and the object's length. The hand measured value is also shown for comparison. If $\eta < 0.25$, the object is placed lying down instead of upright.

In total, we have run *SurfacePlace* on Domo over two hundred times. Within the appropriate class of objects, it is nearly always successful at its task. Failures are predominantly caused by a poor grasp on the object such that the misalignment tolerance is exceeded. In these cases, *CompliantLower* creates moments on the object that cause it to rotate until it lies flat on the surface.

7.6 Discussion

In this chapter, we have demonstrated that a robot can leverage its physical embodiment to assist in perception and reduce uncertainty. It illustrates the design theme of *let the body do the thinking*. In particular, we have emphasized algorithms that do not require precise geometric and kinematic knowledge about the state of the robot and the world. For example, Domo leverages low manipulator stiffness and passive compliance to detect unexpected contact with the world or to allow a grasped object to reorient during placement. The *SurfaceTest* module demonstrates a compensatory action, where the robot takes action to test a perceptual hypothesis. We should also note that, as much as possible, modules such as *GraspDetect* use the sensory state of the robot in the world rather than an internal representation of its state. In our experience, this has increased the robustness and responsiveness of the robot, especially during unexpected task failures that would be difficult to include in an internal model.

The modules presented in this chapter could be expanded in several ways. Force sensing of the contact moments during *SurfacePlace* would allow Domo to actively adjust the object's orientation. *SurfaceTest* could be expanded beyond shelf edges to include any type of flat surface. The limited force sensing resolution and the simple model used in *ContactDetect* prevents Domo from knowing the wrench of interaction forces acting on the manipulator. This information would be useful in developing more adaptive algorithms. Finally, *StiffnessAdapt* simply sets the manipulator stiffness to a hand designed value at a modules request. It would be profitable for this stiffness to be autonomously adapted in response to sensory feedback.

CHAPTER 8

Cooperative Manipulation

In this chapter, we investigate several aspects of our cooperative manipulation design strategy. The relevant SLATE modules are highlighted in Figure 1-3. We present the *PersonDetect* and *VocalRequest* modules that enable Domo's real-time interaction with people. We then describe the *AssistedGrasp* module that cues a person for assistance in grasping an object. Finally, we test experimentally the hypothesis that people will intuitively assist Domo in a manipulation task without prior instruction.

8.1 Perception of People

People expect a robot collaborator to be responsive to their presence in a room, to understand the intent of their gestures, and to adapt to their feedback. Ultimately, cooperative manipulation requires that the robot can understand referential (looking and pointing) and goal-directed (reaching) cues. This is an open area of research involving many difficult perception problems such as gesture recognition and learning from demonstration. However, many cooperative manipulation tasks can be designed around simpler communicative cues so long as the cues are robustly detected and a person will generate them without much effort. For example, in the previous chapter we saw that *ContactDetect* can be used to detect a person grabbing the arm as it moves. A person will intuitively generate this cue to

Figure 8-1: During human-robot interaction, faces, hands, and fingers, are important perceptual features for visual attention. As shown in green, the *InterestRegions* module will often select for these features because they are well modeled as rapidly moving convex edges.

guide another person during a task. In this section, we present modules that allow Domo to detect and track a collaborator, react to a hand waving cue, and respond to vocal requests. In these real-time visual algorithms, we emphasize robustness to cluttered, everyday environments with variable lighting.

8.1.1 Detecting Hands, Fingers, and Faces

During a cooperative manipulation task, much of a person's attention is directed towards the hands, fingers, and face of their partner as well their own hands and fingers. Detecting these features can be a difficult task for a robot working in a dynamic and unstructured environment. However, in collaboration with Kemp, we have found that the *InterestRegions* module will select for these features during human-robot interaction (Kemp and Edsinger, 2006b). This result is illustrated in Figure 8-1.

The interest point operator used by *InterestRegions* detects convex edges that are moving rapidly with respect to the background. A person's hand will often produce the fastest moving edge in the image because it is the furthest point from the arm's center of rotation. It also has a roughly convex projection in the image. The same holds for a person's fingertips and head. During interactions with Domo, people naturally gesture with these features and bring them close to its cameras. The motion from these gestures can serve as natural cues to direct the robot's attention.

We evaluated the ability of *InterestRegions* to select for these features during human-robot interaction. First, the robot arm reached to random targets in its workspace. During

Figure 8-2: A random sample of image patches collected by the *InterestRegions* module. These predominantly contain hands, fingers, and faces. The lighter, grey colored patches correspond to the robot's hand.

this reaching, a person's interacted with the robot, creating visual dynamics such as full-body motion, hand waving, presentation of objects, as well as physical contact with the robot's arm as it moved. As the person interacted with the robot, we collected approximately 4000 image samples over 4 minutes.

For each sample, we used the *InterestRegions* module to select the most salient region in the image. An image patch was selected based on the scale and location of the interest region. Figure 8-2 shows a random sample of the collected image patches. The majority of the patches correspond to features relevant to human-robot interaction, including the person's head, eyes, hands, and fingers, and the robot's hand and fingers. We hand categorized 200 randomly selected image patches from the data set. Figure 8-3 shows that over 50% of these patches were of human features, and over 30% were of the robot's hand.

This experiment was conducted while keeping the robot's gaze stationary, but similar results have been obtained when the head is allowed to move. Although the *InterestRegions* module will detect many features that do not correspond to a person, the results suggest that it can be used to direct the robot's attention to locations for more specialized perception of human features.

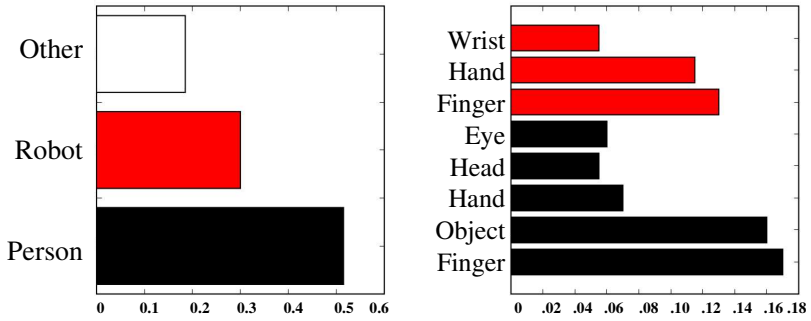


Figure 8-3: Statistics of feature categories detected during human-robot interaction. We hand-labelled categories for 200 image patches randomly sampled from the image patches collected by the *InterestRegions* module. A patch was labelled as a person (black bars) if it selected either a person's hand, finger, head, eye, or object in their hand. A patch was labelled as a robot (red bars) if it selected either the robot's hand, finger, or wrist. Patches that were neither person nor robot were labelled as other. The left plot shows the probability of each category and the right plot shows the probability of each sub-category.

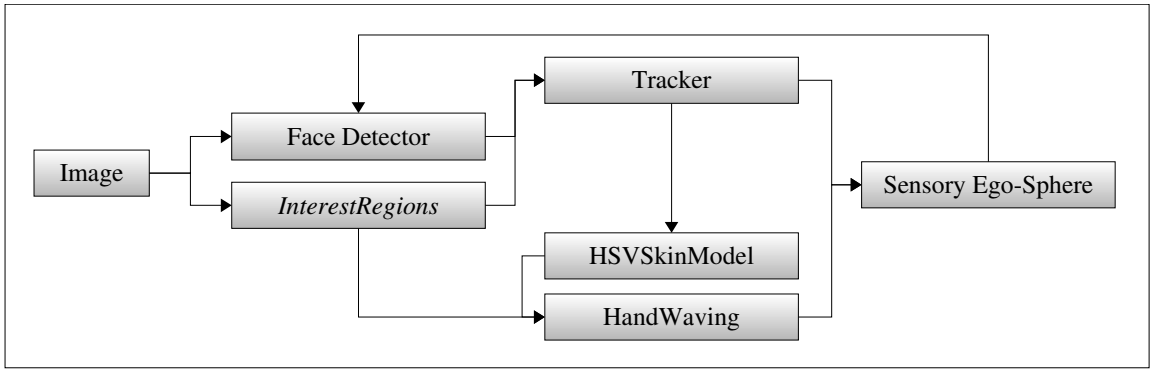


Figure 8-4: The *PersonDetect* module. A face detector initializes a tracker and the tracked face is used to build an online model of the person's skin color. This model is combined with *InterestRegions* to select for salient moving features on a person. These often correspond to the person's hand. Accumulated evidence of hand motion is detected as a hand waving cue. The face and hand waving features are made available to the SES in order to direct the robot's attention. The SES models the spatial distribution of faces, and the model is used to refine the face detector.

8.1.2 *PersonDetect*

PersonDetect is a module in the visual attention system dedicated to perception of the robot's cooperative partner. As illustrated in Figure 8-4, it detects and tracks a person's face, models the skin color of the face, and uses the skin color and *InterestRegions* module to detect a person's waving gesture.

Tracking Faces

We use the face detector of Viola and Jones (2004) provided in OpenCV to detect one or more faces in 160×120 monochrome images at 30Hz. Whenever a face is detected, a 4×4 pixel block centered on the face is used to initialize a tracker. The tracker uses block matching over a 6×6 pixel window to track the block between consecutive images. The location of a tracked face is continually refreshed within the SES. The tracker also produces a confidence measure, defined as the pixel sum difference between the current block and the initialization block. The tracker times out and assumes it has lost the face when the confidence is below a threshold for longer than 60 frames. Fortunately, people normally move slowly within a scene, and the timeout allows the face detector time to reacquire a frontal view of the face. The output from the face tracker is shown in Figure 8-5.

The Spatial Distribution of Faces

As Domo interacts with people and tracks their faces, *PersonDetect* models the spatial distribution of the face detections within the SES. This model is learned over an extended period of many human-robot interactions. The detections are used to estimate the probability distribution $p(\mathbf{x}^S|face)$ of Equation 6.5, which represents the chance of seeing a face at location $\mathbf{x}^S$. Figure 8-6 shows the distribution after many hours of interaction with the robot. Once learned, $p(\mathbf{x}^S|face)$ can provide a confidence measure on a new face detection. For example, *PersonDetect* will ignore a face detection if $p(\mathbf{x}^S|face) < 0.35$. This prevents the robot from falsely detecting faces on the floor or ceiling.

Modelling Skin Color

When a face is detected, the image patch within the detection bounding box is used to automatically update a model of the person's skin color. The patch is first converted from

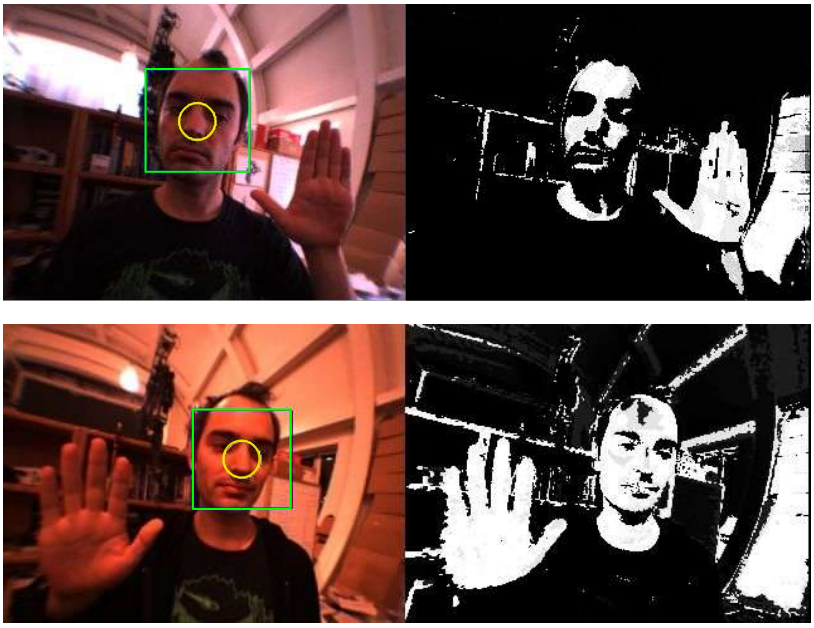


Figure 8-5: Output from the face tracker and skin color filter. The HSV skin color model adapts to the large shift in apparent skin hue between daylight (top) and night (bottom).

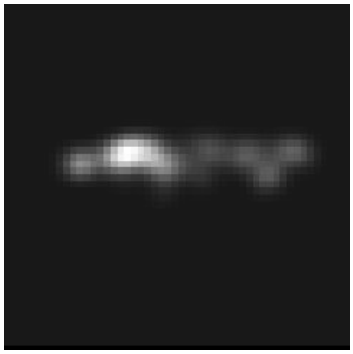


Figure 8-6: The 2D histogram representing the accumulated probability distribution, $p(\mathbf{x}^S|face)$, for face detections within the SES. The center of the histogram, $\mathbf{x}^S = [0, 0, r]$ corresponds to the robot looking straight ahead. (The asymmetry occurs because we typically work off to one side of the robot)

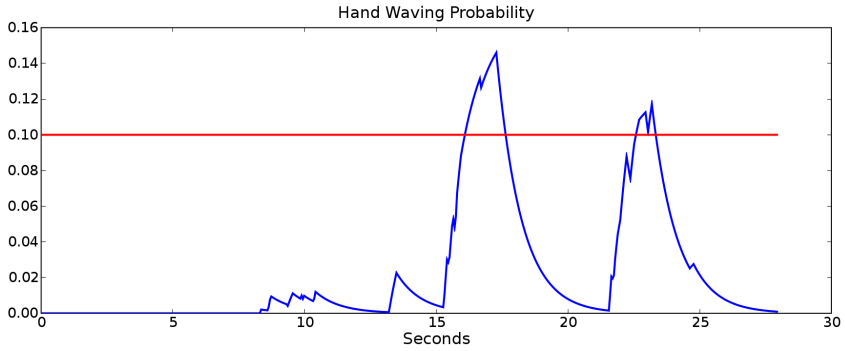


Figure 8-7: The probability of hand waving, $p(\mathbf{x}_h^S|hand)$, over a 30 second trial. The detector threshold $\epsilon_h = 0.1$ is shown in red. The skin color model is built during the first 10 seconds of the trial. Head motion appears during time 10s to 15s with a low probability. Two brief hand waving episodes are detected in the last 15 seconds of the trial.

RGB to *HSV* color space. The hue and saturation of each pixel is added to a 2D 16×16 bin histogram. This defines the probability distribution $p(\mathbf{k}|skin)$, which represents the chance that pixel $\mathbf{k}$ is skin colored. Now given a region $\mathbf{K} = [\mathbf{k}_1 ... \mathbf{k}_n]$ within the image, we compute the probability that the region is skin colored as the average of each pixel:

$$p(\mathbf{K}|skin) = \frac{1}{n} \sum_{\mathbf{k} \in \mathbf{K}} p(\mathbf{k}|skin).$$

Also, each contribution to the histogram is given an initial weight of 1.0. This weight is decremented by a small (.001) amount each time step, biasing the histogram toward more recent face detections. As shown in Figure 8-5, this allows the skin color model to adapt to new people and changing lighting conditions. This online approach is similar to that used on the ARMAR humanoid by Stiefelhagen et al. (2004).

Waving Detection

PersonDetect also detects hand waving. As we demonstrated previously, features selected by *InterestRegions* often correspond to hands and faces. An interest region $\mathbf{K}$ in the image corresponding to location $\mathbf{x}^S$ in the SES is labelled as a hand feature whenever the region is skin colored, it is not a face, and a face is present in the SES. In other words,

$$p(\mathbf{K}|skin) > \epsilon_s,$$

and

$$p(\mathbf{x}^S|face) < \epsilon_f,$$

for thresholds ϵ_s and ϵ_f. This simple hand detector is prone to false positives such as moving flesh colored objects and other moving body parts. However, false detections are generally distributed across the SES. Repeated hand motion in one location can accumulate evidence over time that a person is waving to the robot. Evidence is accumulated by modelling the spatial distribution of hand detections in the SES as $p(\mathbf{x}^S|hand)$ using Equation 6.5 with $\Delta w_{hand} = 0.1$. If we define $\mathbf{x}_h^S = \text{Argmax}\left(p(\mathbf{x}^S|hand)\right)$, then *PersonDetect* signals hand waving whenever $p(\mathbf{x}_h^S|hand) > \epsilon_h$ for some threshold ϵ_h. In Figure 8-7 we plot $p(\mathbf{x}_h^S|hand)$ as Domo interacts with a person. We see that the hand waving is readily detected.

8.1.3 *PersonSeek*

The *PersonSeek* module allows the robot's gaze to be responsive to a person in the room. Its algorithm is straightforward:

1. If a face is present in the SES, servo the head to track the face as they move around the room using *VisualFixate*.

2. If a face is not present in the SES, periodically (0.25Hz) redirect the robot's gaze to locations that have a high probability of finding a person. A gaze location is drawn from the distribution $p(\mathbf{x}^S|face)$.

3. Whenever hand waving is detected, or $p(\mathbf{x}_h^S|hand) > \epsilon_h$, redirect the gaze to the waving location $\mathbf{x}_h^S$.

The module gives priority to hand waving over face detection. This allows a person to guide the robot's attention away from the face to an object of interest. Once the robot is attending to the object, other modules can assume control of the head if desired. Otherwise, the robot's gaze is returned to the face after a 5 second timeout. In the absence of any competing modules seeking control of the robot's head, *PersonSeek* will be active. As a consequence, after a manual task is complete, *PersonSeek* automatically redirects the gaze to the person. This provides an intuitive cue to the person that the robot has finished its task.

8.1.4 *VocalRequest*

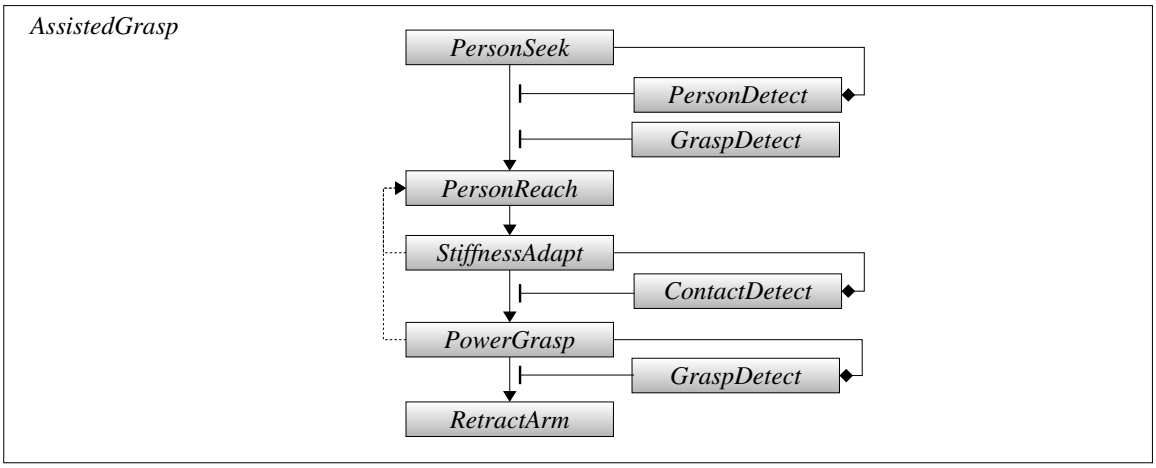
Domo has a simple speech interface, allowing it to respond to and generate vocal requests. Motor noise typically degrades voice recognition performance when using microphones attached to the robot's body. Consequently, we require that a person talking to Domo wear a wireless Sennheiser lavalier microphone. The *VocalRequest* module uses the CMU SPHINX2 voice recognition package that is freely available on Linux. SPHINX is configured to recognize a small vocabulary of commands such as "Take", "Give", and "Shelf". The prefix "Domo" is required, allowing *VocalRequest* to ignore speech that is not directed at it. In response to a command, Domo repeats the phrase using the DIGITAL DECTALK voice synthesizer, allowing the person to confirm that they were heard correctly. *VocalRequest* then increases the dynamic priority of the command's module. For example, saying "Domo, put this on the shelf" will increase the priority of the *SurfacePlace* module.

8.2 *AssistedGrasp* and *AssistedGive*

With the *AssistedGrasp* module, Domo is able to gain a person's help in grasping an object. The *AssistedGrasp* module simply extends the robot's open hand towards its collaborator. If this gesture is generated in the appropriate context, it should be understood by the collaborator as a request to be handed a particular object. If Domo is successful in enlisting the collaborator's assistance, Domo will detect the object being placed in its hand and form a stable grasp on it. This simple example of cooperative manipulation has also been recently demonstrated on the Robonaut platform (Diftler et al., 2004).

The *AssistedGrasp* algorithm is described in Figure 8-8 in terms of the generic manual skill algorithm of Section 5.3.3. The module can be activated so long as one hand is empty. First, it employs *PersonSeek* to find a person in the room. If *PersonDetect* finds a face at location $\mathbf{x}^S = [\theta, \phi, r]$, then *PersonReach* uses the inverse kinematic model to reach to the target $[\theta, \phi + \frac{\pi}{4}, r]$. This location should be near the person's midriff. The orientation of the extended hand and the pose of its fingers can also be used to cue the person how to hand the object. For example, a glass of water is usually grasped in a different orientation than a stirring spoon. This desired orientation is directed by an external module.

Once *PersonReach* has achieved its target and the arm is nearly stationary, *StiffnessAdapt* lowers the stiffness of the arm. This increases the likelihood of *ContactDetect* given



Ready Wait until *PersonSeek* and *PersonDetect* have located a person but *GraspDetect* is not signaled .

Precondition Use *PersonReach* to bring an open hand near the person while directing the eye gaze to their face.

Compensatory Using *StiffnessAdapt*, lower the arm stiffness to better detect contact.

Detect Using *ContactDetect*, signal when the object is placed in the hand.

Control Using *PowerGrasp*, grasp the offered object.

Success If *GraspDetect* signals success, allow *RetractArm* to lower the arm. Otherwise, re-cue the person.

Figure 8-8: The *AssistedGrasp* module gains assistance from a person in order to grasp an object.

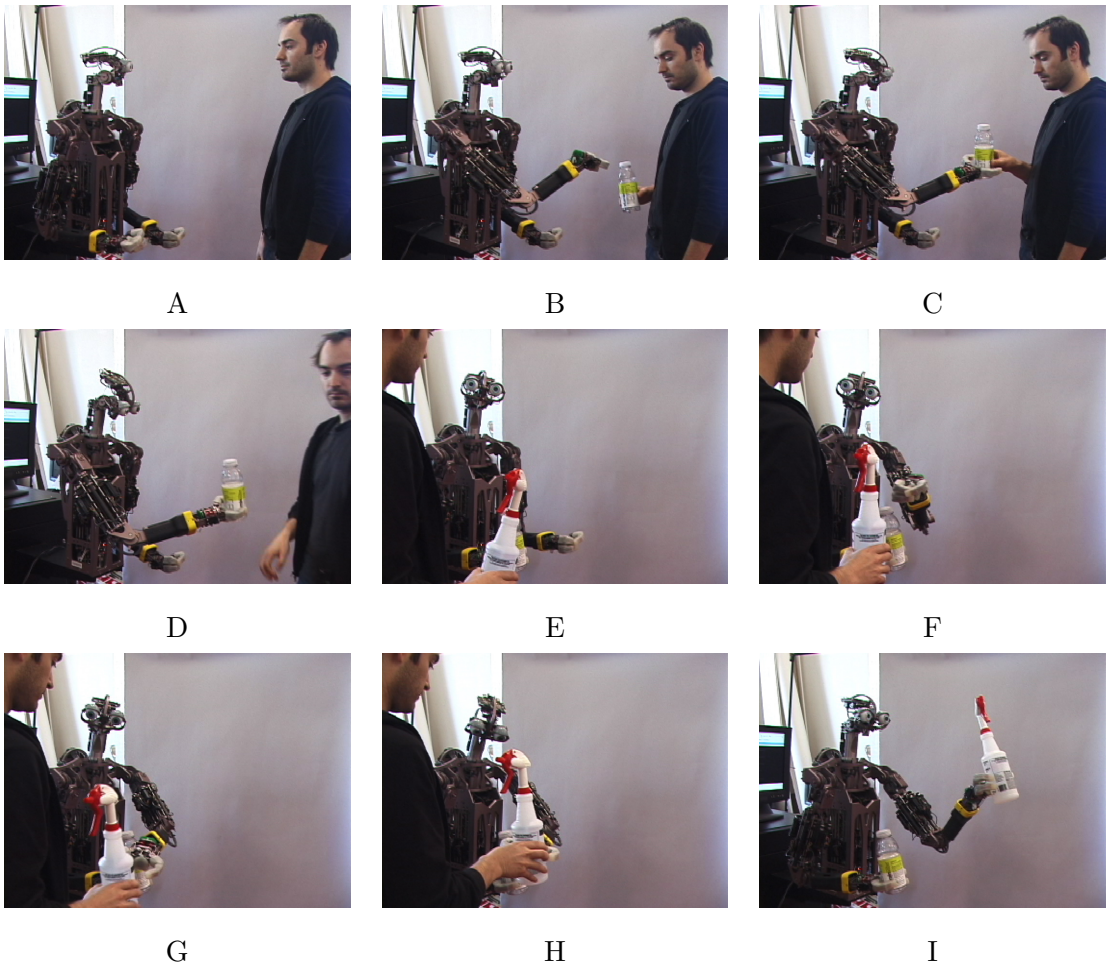


Figure 8-9: Domo executing the *AssistedGrasp* module to grasp a water bottle (A-D) and spray bottle (E-I). Domo visually detects the collaborator (A,E) and tracks him throughout the experiment. It reaches towards the collaborator with an open hand, signalling him to offer an object (B,G). Domo detects the object being placed in its hand (C,H) and forms a power grasp on it (D,I).

small contact forces. As the person gives Domo the object, small displacements of the hand are sensed by *ContactDetect*. *PowerGrasp* then closes the fingers around the object. If *GraspDetect* signals success, *RetractArm* brings the grasped object to the robot's side. However, if *ContactDetect* or *GraspDetect* fail, then *PersonReach* is reactivated and the robot cues the person again. Figure 8-9 shows the execution of *AssistedGrasp*.

AssistedGrasp is typically activated by other modules that require assistance in grasping an object. However, *AssistedGrasp* can also assist the collaborator by simply holding something for them. For example, a person can say "Domo, hold this" and *VocalRequest* will activate *AssistedGrasp*, causing Domo to reach out and take the object. In another scenario, a person can non-verbally request assistance by simply waving the object in front of Domo. The hand motion is detected by *PersonDetect* which will also activate *AssistedGrasp*. In these ways, *AssistedGrasp* allows for natural interaction between the robot and collaborator.

While *AssistedGrasp* takes an object from a person, the *AssistedGive* module hands a grasped object back. Its implementation is nearly identical to *AssistedGrasp,* only now *GraspRelease* is used instead of *PowerGrasp*. If a collaborator had asked for assistance by requesting "Domo, hold this", they can now request "Domo, give it" and *AssistedGive* will cause Domo to reach to the person and hand the object back.

8.3 Testing Cooperative Manipulation

If we take *cooperative manipulation* as a design strategy, to what extent can we include the actions of a collaborator into the task design? This is an important question if we wish to develop manipulation strategies that depend on people for task success. By placing the collaborator "in the loop", a robot can do more with less. However, this should also be a beneficial experience for the person, otherwise they won't be inclined to work with the robot. Not only should the robot do something useful, but the collaboration should occur in a natural, intuitive manner without requiring excessive training for the collaborator or mental effort in assisting the robot.

One way to induce such a collaboration is to have the robot use socially understood gestures and cues. We maintain that such cues will be interpreted correctly if they are generated in the appropriate context. For example, *AssistedGrasp* expects that a collaborator

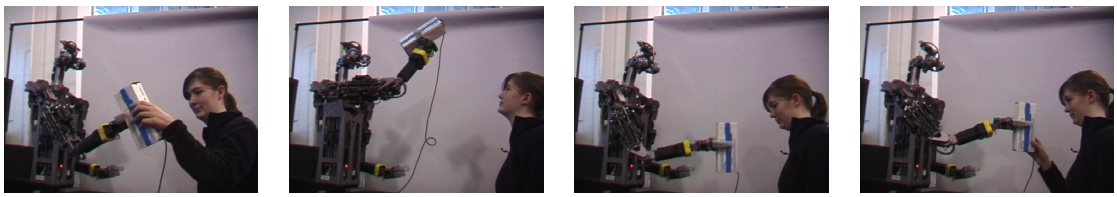


Figure 8-10: One trial of the Give and Take experiment.

will understand the reaching cue as a request for a particular object. This assumes that the collaborator will assess the task context to select the correct object. People exhibit this behavior when they pass their coffee cup but not their dinner plate to a waitress holding a pot of coffee.

We also maintain that a collaborator will develop an understanding of the robot's perceptual and motor limitations. A well intentioned collaborator will subsequently adapt their actions to increase task success. As we will see, during *AssistedGrasp,* a collaborator will intuitively hand the object to the robot in a pose that anticipates both the robot's use of the object and the limitations of its grasping.

This type of cooperative interaction as been investigated recently with the Dexter humanoid (Hart et al., 2006). Dexter learned that when a desired object is out of reach, the act of reaching towards the object can induce a person to move the object closer. However, the person's cooperation occurred as part of the experiment's design and not because they were compelled to. Can a robot compel a person to take on the role of collaborator without instruction? In this section we present an experiment that investigates this question.

8.3.1 The Give and Take Experiment

Experiment Setup

As shown in Figure 8-10, the subject sits in front of the robot. The robot is at a table and an oblong box (60mm $\times$ 85mm $\times$ 200mm) sits on the table. The box is instrumented with a gyroscope to measure its orientation. The subject is told only that the robot is performing an unspecified visual hand-eye calibration task, and that whenever the robot reaches to them, they are to place the box in the robot's hand. This explanation is to deter the subject from explicitly considering the way in which they hand the box to the robot. In a single trial, the robot reaches to the subject using *AssistedGrasp* with the hand open

in a power-grasp preshape. The subject places the box in the robot's hand and the robot grasps the box, brings the box up to its cameras, and appears to inspect it. Depending on the subject, it then lowers its arm in one of two ways.

In the first case, the robot reaches towards the person, bringing the box just in front of and above the table edge nearest the subject. It says the word "Done" and pauses for one second. It then releases its grasp, dropping the box onto the table, and retracts its arm. In the second case, the robot does an identical action as in the first case, but this time it reaches just past the table edge. Unless the subject takes the box from the robot, it falls to the floor. This marks the end of a trial. The robot pauses for 5 seconds and then initiates the next trial. Six trials are performed with each subject.

At the start of each experiment, the box is aligned to the robot's body and the gyroscope is calibrated with respect to frame $\{W\}$. We define the vector $\mathbf{b}^W$ as the longest edge of the box. We define the power-grasp axis as $\mathbf{z}^H$ which is the z-axis of frame $\{H\}$ in Figure 4-1. This axis corresponds to the long axis of a soda can held in a power grasp. In frame $\{W\}$ this axis is $\mathbf{z}^W$. The angle between $\mathbf{z}^W$and $\mathbf{b}^W$ is defined as the grasp alignment error. During each trial, we measure the average grasp alignment error during the 500ms just prior to the grasp being formed. We also vary the wrist rotation for each of the six trials such that the angle formed between $\mathbf{z}^W$ and gravity is $[0°, -45°, 90°, 0°, -45°, 90°]$.

Experiment Hypothesis

This experiment considers the following three questions:

1. When a subject hands the robot the box, do they adjust its orientation to match the pose of the robot's hand?

2. Will the subject correctly interpret the robot's reaching gesture, vocalization, and pause as a social cue to take the object?

3. Can a small incentive such as not having to pick up the object increase the subject's willingness to respond to the social cue?

We can use the measured grasp alignment error to directly answer the first question. We would expect to see $\mathbf{b}^W$ to track $\mathbf{z}^W$ as it varies between the three wrist orientations. The second question is more difficult to confirm. For each experiment, we measure the take-back

rate as the number of trials a subject reached to take the box back from the robot. We expect that robot's reaching gesture, vocalization, and pause will cause the subjects to take back the box. However, they are never instructed to take it back, so if they do, then it is likely that they correctly interpreted the cue, much as they would with a person. For 50% of the subjects, the robot drops the cylinder on the floor. If this serves as an incentive, we should see an increase in the take-back rate. Importantly, the arm postures achieved by the incentive-reach and the no-incentive-reach are nearly identical. Therefore, the effect of the reaching cue, prior to dropping the box, should be similar.

Experiment Results

Prior to the experiments, we first measured the average grasp alignment error when we deliberately oriented the box to match the robot's grasp. From repeated trials the mean error was 8.9°. Next, we measured the possible range of grasp alignment errors by freely rotating the box within the preshaped hand. In this case the distribution of grasp errors was fairly uniform between 0° and 60°. We tested the experiment using 10 subjects. All subjects were naive to the experiment objectives, and had never interacted with Domo before. In Figure 8-11, we the grasp alignment errors for each of the six trials of a typical subject. All subjects were matched the orientation of the offered box to the orientation of the robot's hand with surprising accuracy. In Figure 8-12 we see that the average error for each subject is near the average error achieved by the author. We also see that, when offered the box by the robot, nearly all subjects took the box back, particularly when the additional incentive was included. This experiment only scratches the surface of the potentially rich interactions that may occur during cooperative manipulation. However, it shows quantitatively that people will intuitively adapt to and assist the robot without instruction. We would expect that more substantiative assistance could be given if the person possessed greater contextual knowledge about the task and the robot could generate more nuanced cues.

8.4 Discussion

The interactions between a robot and its collaborator are potentially rich and varied. Consequently, much work is still needed in understanding how to perceive the intent of the

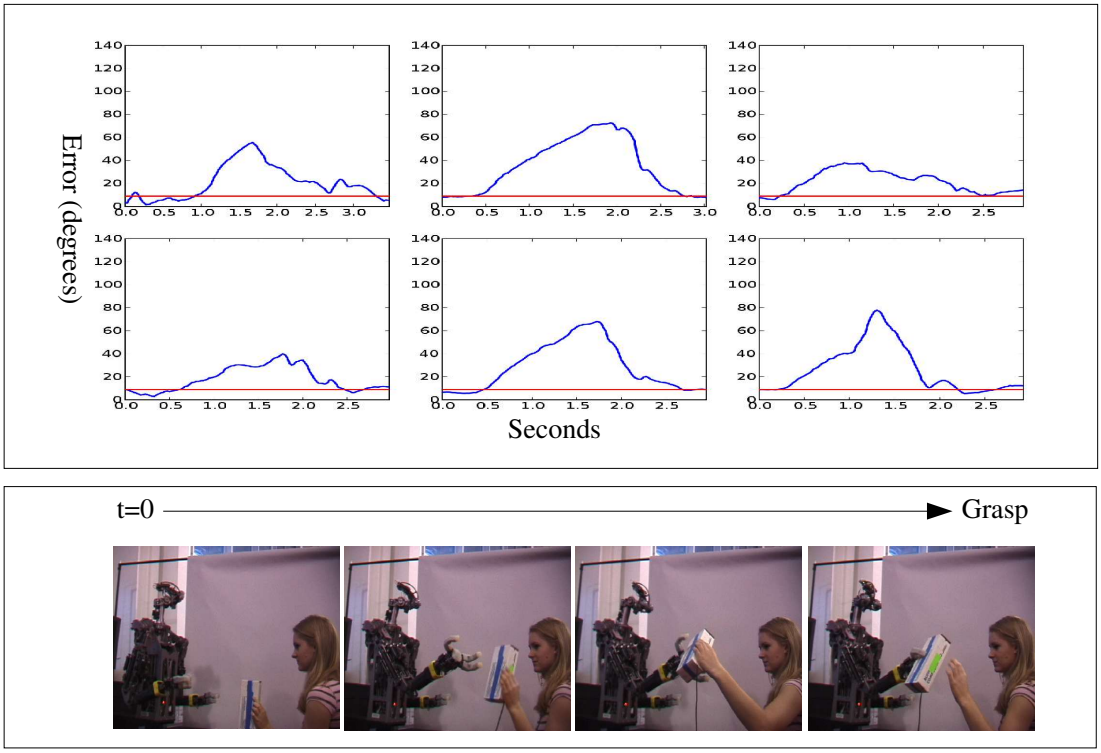


Figure 8-11: The grasp alignment errors for a typical subject during the Give and Take experiment. We see that for nearly all trials, the subject aligns the box within a few degrees of the best expected performance. (Top) Blue shows the grasp alignment error (degrees) of the box with respect to the grasp preshape. Red (horizontal) shows the mean error achieved when we deliberately aligned the box with the grasp. The X axis shows the trial time, in seconds, starting when the reach commenced and ending when the grasp was initiated. (Bottom) The execution sequence from the initiation of the reach until the grasp is formed.

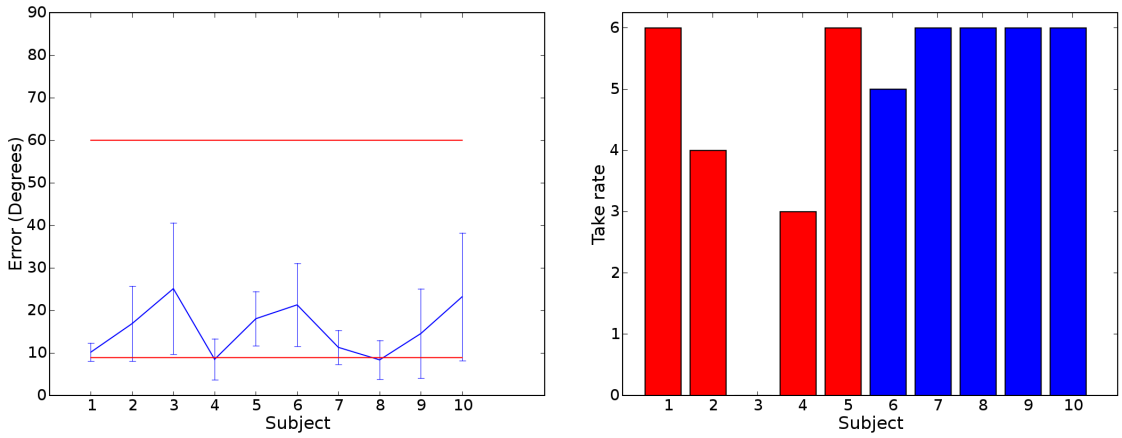


Figure 8-12: (Left) Results showing that the 10 subjects intuitively aligned the cylinder's axis with the grasp axis of the robot's hand when handing it. For each of 6 trials, we measure the average alignment error, in degrees, during the 500ms prior to grasping. The error bars show the mean error and standard deviation for each subject. The red lines indicate the best and worst expected performance. (Right) Results showing the number of trials each subject took the box back from the robot when cued. For subjects 1-5 (red), the robot dropped the box on the table, while for subjects 6-10, the box was dropped on the floor.

person, respond appropriately to their requests, and leverage their innate social machinery. In particular, perception of pointing, eye gaze, reach direction, and physical interaction cues could greatly expand a robot's repertoire of cooperative tasks. These are open research issues currently being studied within the social robotics community, and more recently, within the emerging area of physical human-robot interaction (Alami et al., 2006). Although preliminary, the results of this chapter suggest that even basic cooperative skills can be useful to robots in human environments. A person can bring years of experience to a task, not to mention capable motor and perceptual skills. Through careful design on the part of the roboticist, a person can be intuitively brought into the loop by the robot.

CHAPTER 9

Task Relevant Features

Everyday tasks such as placing a cup in a cabinet or pouring a glass of water can be described in terms of the perception and control of *task relevant features*. We have seen previously how the edge of a shelf can represent the presence of a stable, flat surface. In this chapter we show that the distal tips of many everyday objects can be treated as task relevant features as well. The relevant SLATE modules are highlighted in Figure 1-3.

We first present a robust method to estimate the 3D location of an object's tip in the hand's coordinate frame. This work was developed in collaboration with Kemp (Kemp and Edsinger, 2005). Next, we describe the *TipPose* and *TipServo* modules which provide manipulator control of this feature. We then generalize the visual detection of the tip to include a broader class of objects. This work was also done in collaboration with Kemp (Kemp and Edsinger, 2006a). We then show that a prediction of the feature location can be learned. Finally, we integrate these components into the *TipUse* module which enables Domo to take an object from a person, quickly find its distal tip, and control the tip for a task.

Figure 9-1: (Left) A typical view from the robot's camera of the hand holding a pair of pliers. A naturally lit, cluttered background ensures a non-trivial environment for perception. (Middle) The tool tip is detected as the maximum point in the weighted edge map of the visual motor model. (Right) The raw motion-based detection (black), the hand-labeled tool tip (white), and a prediction based on the estimated tool position (green) are annotated.

9.1 The Task Relevant Tool Tip

For a wide variety of tools and tasks, control of the tool's endpoint is sufficient for its use. For example, use of a screwdriver requires precise control of the tool blade relative to a screw head but depends little on the details of the tool handle and shaft. Radwin and Haney (1996) describe 19 categories of common power and hand tools. Approximately 13 of these tool types feature a distal point on the tool which can be considered the primary interface between the tool and the world.

The prevalence of this type of feature may relate to the advantages it gives for perception and control. For perception, it improves visual observation of the tool's use by reducing occlusion, and it assists force sensing by constraining the interaction forces to a small region. For control, its distal location increases maneuverability by reducing the possibility of collisions. A single tip also defines the tool's interface to the world as a simple, salient region. This allows the user to perceptually attend to and control a single artifact, reducing their cognitive load. Looking beyond human tools, one can also find this structure in the hand relative to the arm, and the finger tip relative to the finger.

In this section, we present a straight-forward monocular method for rapidly detecting the endpoint of an unmodeled tool and estimating its position with respect to the robot's hand. The process is shown in Figure 9-1. This can allow the robot to control the position of

the tool endpoint and predict its visual location. These basic skills can enable rudimentary use of the tool and assist further learning by helping the robot to actively test and observe the endpoint. We show successful results for this estimation method using a variety of traditional tools shown in Figure 9-3, including a pen, a hammer, and pliers, as well as more general tools such as a bottle and the robot's own finger.

To find the tip of a tool held in the hand, the robot rotates the tool while detecting the most rapidly moving point between pairs of consecutive images. It then estimates the 3D point in the hand's coordinate system that best explains these noisy detections. Given this protocol, motion serves as a powerful cue for detecting the endpoint of a wide assortment of human tools. The method makes few assumptions about the size and shape of the tool, its position in the robot's hand, or the environment in which the tool is being manipulated. Importantly, we use the robot's kinematic model to transform the perceptual detections into the hand's coordinate frame, allowing for the registration of many detections from multiple views. This makes the algorithm robust to noise from sources such as ego-motion and human interaction.

9.1.1 Related Work in Robot Tool Use

Research involving robot tool use often assumes a prior model of the tool or construction of a model using complex perceptual processing. A recent review of robot tool use finds few examples of robots using everyday, human tools (St. Amant and Wood, 2005). NASA has explored the use of human tools with the Robonaut platform. Robonaut used detailed tool templates to successfully guide a standard power drill to fasten a series of lugnuts (Huber and Baker, 2004). Approaches that rely on the registration of detailed models are not likely to efficiently scale to the wide variety of human tools. Williamson (1999) demonstrated robot tool use in rhythmic activities such as drumming, sawing, and hammering by exploiting the natural dynamics of the tool and arm. However, this work required careful setup and tools that were rigidly fixed to the hand.

A long history of work in AI and computer vision has focused on learning tool function, although the representations are typically symbolic and model based (Winston et al., 1983). More recently, researchers have considered a tool's function in the context of its action in the world. In Duric et al. (1996), a tool's function was associated with its prototypical motion based on visual observation of a person using the tool. Robots that can actively learn about

tool use have been the subject of more recent work. Bogoni (1998) investigated relating the physical properties of the tool to the perceptual outcomes of its use when tested by a robot. Stoytchev (2005) has explored learning a tool's function through its interactions with objects. However, the work of these researchers typically assume that a clean segmentation of the tool can be extracted from the image or that the tool features are known in advance.

In our method, we use our knowledge of how the robot's hand rotates while holding the tool to make 3D estimations about the location of the tool tip. This relates to methods for 3D scanning in which objects are placed on a rotating platform in front of a single camera (Fitzgibbon et al., 1998). These methods, however, typically rely on a well modeled background to cleanly segment the object, simple platform motion, and occlusion free views of the object. More generally, our estimation technique relates to the well-studied area of 3D estimation from multiple views (Hartley and Zisserman, 2004).

9.1.2 Tool Tip Detection

We wish to detect the end point of a grasped tool in a general way. The detection process looks for points that are moving rapidly while the hand is moving. This ignores points that are not controlled by the hand and highlights points under the hand's control that are far from the hand's center of rotation. Typically tool tips are the most distal component of the tool relative to the hand's center of rotation, and consequently have higher velocity. The hand is also held close to the camera, so projection tends to increase the speed of the tool tip in the image relative to background motion. The wrist is rotated and we detect the tip as the fastest moving point in the image using the method of Kemp described in Section 6.1.

9.1.3 Probabilistic Estimation of the 3D Tool Tip Position

As described, monocular visual motion is used to detect motion feature points that are likely to correspond with the tip of the tool in the robot's hand. After detecting these points in a series of images with distinct views, we use the robot's kinematic model to combine these 2D points into a single 3D estimate of the tool tip's position in the hand's coordinate system $\{H\}$. With respect to $\{H\}$, the camera moves around the hand while the hand and tool tip remain stationary. This is equivalent to a multiple view 3D estimation problem where we wish to estimate the constant 3D position of the tool tip $\mathbf{x}_t^H = \mathbf{x}_t$ in frame $\{H\}$. In an ideal

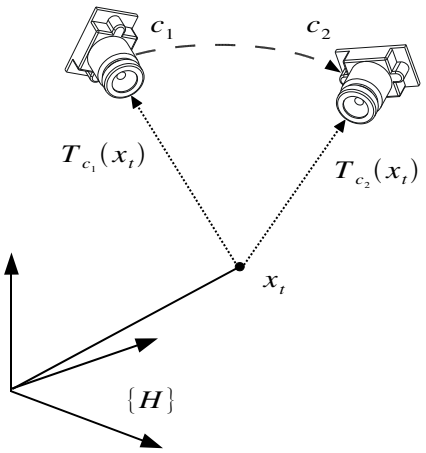


Figure 9-2: The geometry of the tool tip 3D estimation problem. With respect to the hand's coordinate system, $\{H\}$, the camera moves around the hand. In an ideal situation, only two distinct 2D detections would be necessary to obtain the 3D estimate. Given two observations with kinematic configurations c_1 and c_2, the tool tip, x_t, appears in the image at $T_{c_1}(x_t)$ and $T_{c_2}(x_t)$.

situation, only two distinct 2D detections would be necessary to obtain the 3D estimate, as illustrated in Figure 9-2. However, we have several sources of error, including noise in the detection process and an imperfect kinematic model.

A variety of approaches would be appropriate for this estimation, since only three parameters need to be estimated and we have plenty of data from a moderately noisy source. In this paper, we estimate $\mathbf{x}_t$ by performing maximum likelihood estimation with respect to a generative probabilistic model.

We first model the conditional probability distribution, $p(\mathbf{d}_i|\mathbf{x}_t, c_i)$, which gives the probability of a detection at a location in the image, $\mathbf{d}_i$, given the true position of the tool tip, $\mathbf{x}_t$, and the robot's configuration during the detection, c_i. We model the detection error that is dependent on $\mathbf{x}_t$ with a 2D circular Gaussian, $\mathcal{N}_t$, centered on the true projected location of the tool tip in the image, $T_{c_i}(\mathbf{x}_t)$, with variance σ_t. T_c is the transformation that projects $\mathbf{x}_t$ onto the image plane given the configuration of the robot, c_i. T_{c_i} is defined by the robot's kinematic model and the pin hole camera model for the robot's calibrated camera. This 2D Gaussian error model on the image plane can coarsely represent a number of error sources, including the selection of motion edges around the ideal location, and inaccuracies in the kinematic model. We mix this Gaussian with another 2D Gaussian, $\mathcal{N}_f$, centered on

the image with mean 0 and a large variance σ_f. This Gaussian accounts for false detections across the image that are independent of the location of the tool tip. In summary,

$$p(\mathbf{d}_i|\mathbf{x}_t, c_i) = (1-m)\mathcal{N}_t(T_{c_i}(\mathbf{x}_t), \sigma_t^2 I)(\mathbf{d}_i) + m\mathcal{N}_f(0, \sigma_f^2 I)(\mathbf{d}_i), \tag{9.1}$$

where m is the mixing parameter for these two Gaussians.

We model a series of detections $\mathbf{d}_1 \ldots \mathbf{d}_n$ with corresponding configurations of the robot, $c_1 \ldots c_n$, as being independently drawn from this distribution, so that

$$p(\mathbf{d}_1 \ldots \mathbf{d}_n|\mathbf{x}_t, c_1 \ldots c_n) = \prod_i p(\mathbf{d}_i|\mathbf{x}_t, c_i). \tag{9.2}$$

Using Bayes rule we have

$$p(\mathbf{x}_t|\mathbf{d}_1 \ldots \mathbf{d}_n, c_1 \ldots c_n) = \frac{p(\mathbf{d}_1 \ldots \mathbf{d}_n|\mathbf{x}_t, c_1 \ldots c_n)p(\mathbf{x}_t, c_1 \ldots c_n)}{p(\mathbf{d}_1 \ldots \mathbf{d}_n, c_1 \ldots c_n)}. \tag{9.3}$$

Since we are only looking for relative maxima, we can maximize

$$p(\mathbf{d}_1 \ldots \mathbf{d}_n|\mathbf{x}_t, c_1 \ldots c_n)p(\mathbf{x}_t, c_1 \ldots c_n). \tag{9.4}$$

We also assume that the tool tip position in the hand's coordinate system is independent of the configurations of the system at which the images were captured, so that $p(\mathbf{x}_t, c_1 \ldots c_n) = p(\mathbf{x}_t)p(c_1 \ldots c_n)$. Since $c_1 \ldots c_n$ are known and constant for the data set, we can drop their distribution from the maximization to end up with

$$\begin{aligned} \widehat{\mathbf{x}}_t &= \text{Argmax}_{\mathbf{x}_t} \left(p(\mathbf{d}_1 \ldots \mathbf{d}_n|\mathbf{x}_t, c_1 \ldots c_n)p(\mathbf{x}_t) \right) \\ &= \text{Argmax}_{\mathbf{x}_t} \left(\log\left(p(\mathbf{x}_t)\right) + \textstyle\sum_i \log\left(p(\mathbf{d}_i|\mathbf{x}_t, c_i)\right) \right). \end{aligned} \tag{9.5}$$

We define the prior, $p(\mathbf{x}_t)$, to be uniform everywhere except at positions inside the robot's body or farther than 0.5 meters from the center of the hand. We assign these unlikely positions approximately zero probability. A variety of methods could be used to find our estimate $\widehat{\mathbf{x}}_t$ given expression 9.5, including gradient ascent and brute force sampling. We use the Nelder-Mead Simplex algorithm implemented in the open source SciPy scientific library (Jones et al., 2001) to optimize this cost function. More efficient optimization algorithms are applicable, but this algorithm is easy to use since it only requires function evaluations. Even though each evaluation of the cost function requires $O(n)$ computation, where n is the number of detections, we found it to converge quickly given our small set of moderately

Figure 9-3: The approach was evaluated on a hot-glue gun, screwdriver, bottle, electrical plug, paint brush, robot finger, pen, pliers, hammer, and scissors.

noisy observations. There are many sources of error that we ignore in our model, including uncertainty about the hand's rotation (which will have a larger impact on long objects), the projection dependent aspects of the kinematic uncertainty, and uncertainty in the temporal alignment of the kinematic configuration measurements and the motion-based detections.

9.1.4 Tool Tip Estimation Results

Experiments were conducted on a variety of tools with differing lengths and endpoints (Figure 9-3). For each experiment, the 11 DOF kinematic chain from the camera to the robot wrist was servoed to maintain a fixed pose that ensured tool visibility in the wide-angle camera. The tool was placed in the robot's hand and the 2 DOF (pitch,roll) of the wrist were ramped smoothly to random positions in the range of ± 60 degrees for a short duration. The robot's joint angles and camera images were sampled at 30Hz. Approximately 500 samples (15 seconds of motion) were captured for each tool and randomly distributed into a training set of 400 samples and a test set of 100 samples. We then hand labeled the tool tip location for each frame of the test set.

Tool Tip Detection

Visual detection of the tool tip was computed using the motion model from Section 6.1. In these experiments, the localization was computed offline for each pair of sequential images, though real-time rates are achievable. A naturally lit, cluttered background was used (Fig-

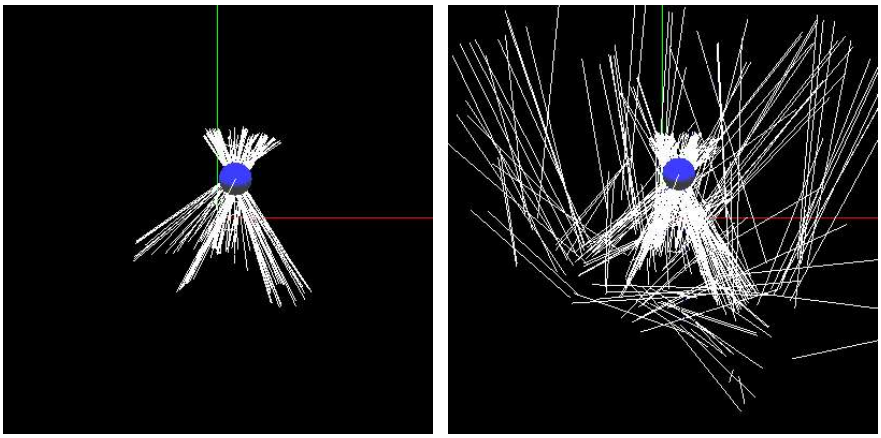


Figure 9-4: As the robot rotates a grasped object in front of its camera, its attention system generates monocular detections of the distal end of the object. Each detection is transformed into a ray within the hand's coordinate system (white). Many such rays intersect at the estimated tip location (blue). The left image visualizes these rays without background motion. The right image is noisier due to background motion, but the tip location is still robustly estimated.

ure 9-1A) to ensure a non-trivial environment for perception and background motion was allowed. The detection method is noisy, but as shown in Figure 9-7, the detections tended to match the hand-labeled tool tip locations. In the experiments we present, the camera and environment were nearly stationary and the affine model of background motion was estimated as close to identity. Without modification the algorithm can be used in situations with a slowly moving camera and other causes of global affine motion.

Tool Position Estimation

The position estimation accuracy was evaluated by first estimating the 3D tool position in the hand on the training data set as described in Section 9.1.3. We used the parameter values $\sigma_t = 5.0$, $\sigma_f = 150.0$, and $m = 0.5$. The 3D position was projected onto the image plane for each sample in the test set. The predicted tool tip location was measured against the hand labelled location to compute the mean pixel error. A baseline comparison can be made by performing the estimation process on the hand labeled data set. The resulting

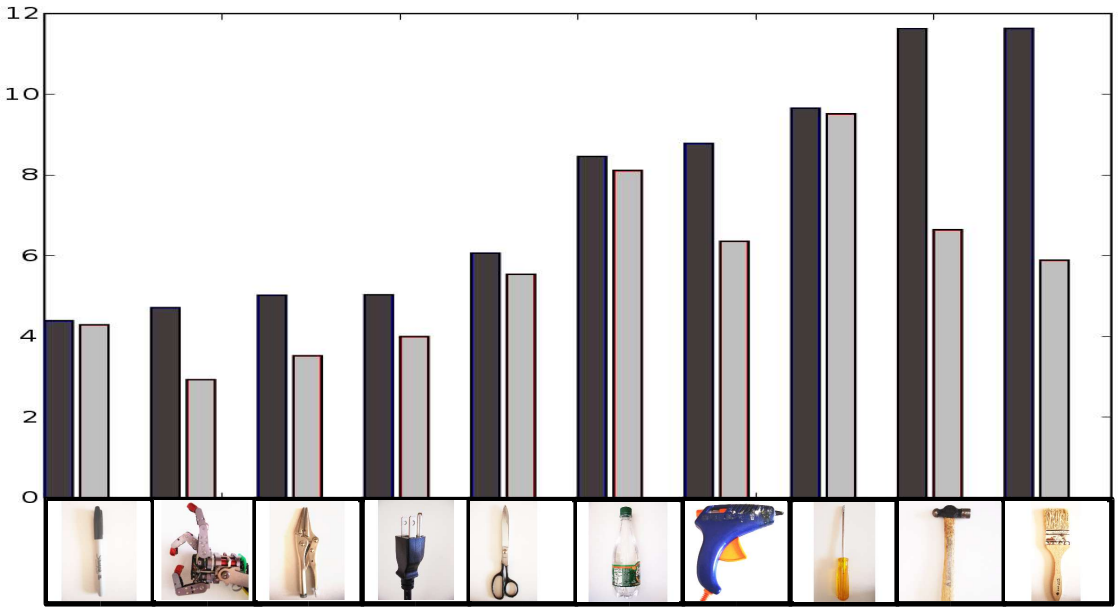


Figure 9-5: The mean prediction error, in pixels, for each tool, using 320×240 pixel images. The 3D tool tip position is estimated using two data sets: motion-based tool tip detection and hand-labelled tip detection (both in the image). The 3D estimates for both data sets are then projected onto the image plane for each sample in the test set and compared to the hand labelled detection. The left (dark) bar indicates the detector error and the right (light) bar indicates the hand labelled, baseline error. The baseline errors are an indication of inaccuracies in the kinematic model and the camera model.

error is indicative of inaccuracies in the kinematic model and the camera model. The algorithm performs favorably with respect to this baseline error. Figure 9-5 illustrates the mean prediction error, in pixels, across the set of tools. Figure 9-6 illustrates the typical tip prediction for each tool.

Discussion of the Results

As Figure 9-6 illustrates, the estimation process performs well as measured by the image prediction error. The wide angle camera from which the images were captured allows a larger variety of tool sizes to be explored, but the resolution of the tip was often low, on the order of 10 pixels. Errors can originate from poor calibration in the kinematic and camera model, as the baseline errors in Figure 9-5 demonstrate. We trained each estimator on a

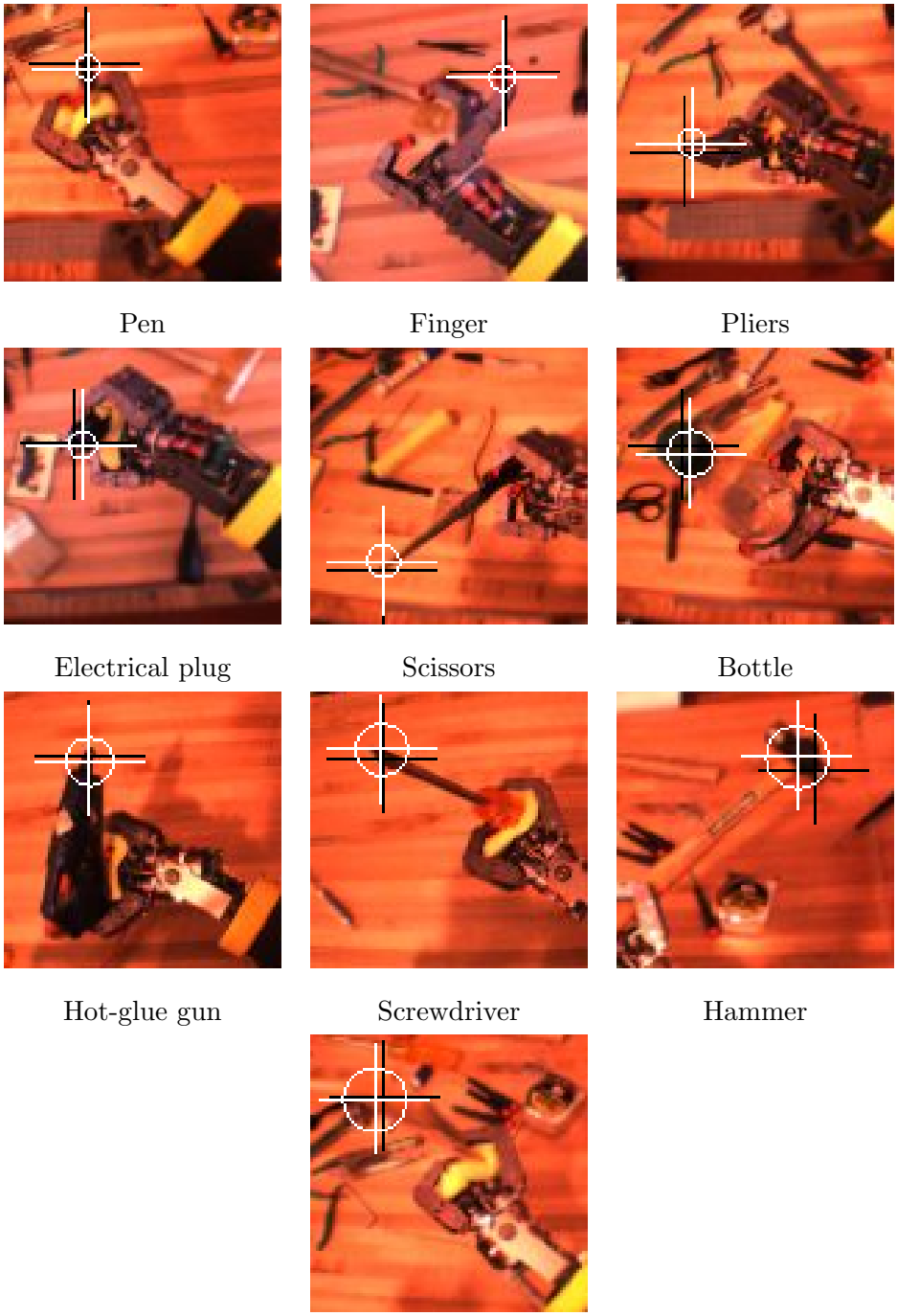


Figure 9-6: An example of the tip prediction for each tool. The white cross is centered at the prediction point and measures 40 pixels across for scale. The radius of the white circle indicates the tool's mean pixel error. The black cross indicates the hand labeled tool tip.

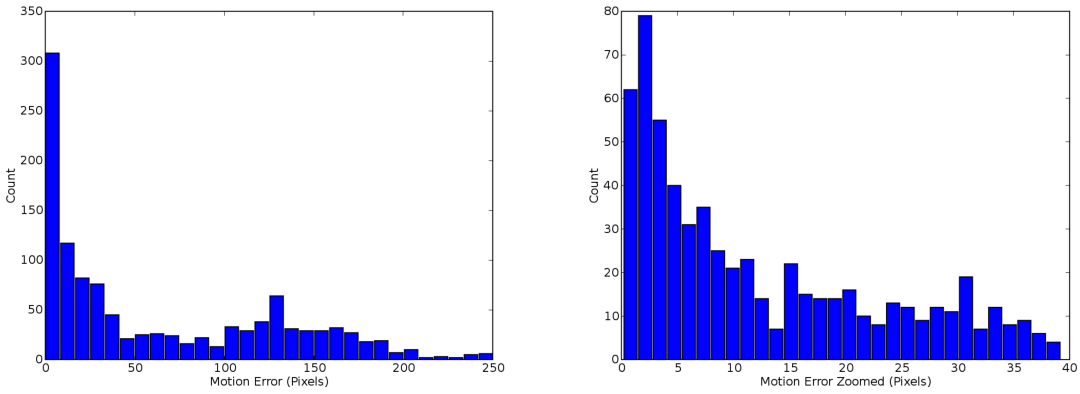


Figure 9-7: (Left) Error histogram, in pixels, for raw motion-based detection of the tool tip with respect to the hand-labeled tool tip for all tools. 320×240 pixel images were used. (Right) Detailed view of the left graph.

data set of 400 samples which is conservatively high given the effectiveness of the motion-based detector and the ideal requirement of only two distinct views. It is important that the wrist sample a large space of poses. In the extreme case of hand rotation occurring only in the image plane, the depth of the tool position would be indeterminate. Finally, we should point out that our method is analyzed with respect to the 2D projection of the tip into the image. The error between the 3D estimate, $\widehat{\mathbf{x}}_t$, and the true position $\mathbf{x}_t$ is dependent on errors in the kinematic calibration. The 13 DOF kinematic chain between Domo's hand and eye amplifies calibration errors, and the 3D estimation error typically varies between 15mm-50mm depending on the arm configuration. In Section 9.2.2 we will show how to use visual feedback to compensate for these errors.

9.1.5 Discussion

In this section we presented a straight-forward approach to detecting the end point of a tool in a robot hand and estimating its 3D position. The strength of our approach is that it assumes little prior knowledge about the tool or its position in the hand and avoids complex perceptual processing. Rather than segmenting the tool, estimating the 3D shape of the tool, or otherwise representing the details of the tool prior to detecting the tip, this method jumps directly to detecting the tip of the tool. The success of the method relies on two main observations. First, the natural utility of many human tools depends on the tool's

endpoint. Second, for many of these tools the endpoint can be detected by its rapid motion in the image when the robot moves its hand while holding the tool. For the results we present, the robot's hand is roughly human in size and shape and thus well-matched to human tools. This detection method might not perform as well with robot end-effectors that differ significantly from a human hand (for example they might be large with respect to the tool).

We estimate the 3D tool tip position in batch-mode by optimizing the cost function of Equation 9.5. A recursive filter, such as an extended Kalman filter, could provide an adaptive, online alternative to the estimation technique. This could be used to adjust for possible slip in the robot's grasp of the tool during use. As we will see in the Section 9.3, other perceptual cues can be integrated into this method. Motion does, however, have some especially beneficial properties for this type of detection. First, motion helps us to find elements of the world that are controlled by the robot's hand. Stereo analysis of a static scene could be used to select elements of the scene that are close to the hand, but without motion, stereo could not detect which points are under the hand's control. Second, by moving the hand and tool we are able to observe them from several distinct views, which reduces sensitivity to the particular position of the hand and increases overall robustness.

9.2 Control of the Tip

9.2.1 *TipPose*

The *TipPose* module places the tip of a grasped tool at a desired position and orientation in the world frame. The 3D estimate of the tip's location within the hand's coordinate frame, $\widehat{\mathbf{x}}_t$, can be used to effectively extend the robot's kinematic model by a link. This provides many options for control. In this section we describe a virtual spring method for control of the position and orientation of this task-relevant feature. Virtual spring control uses the well known Jacobian transpose method for relating manipulator forces to torques (Craig, 1989). It is a straightforward technique strongly related to virtual model control (J. Pratt and Pratt, 2001) and operation space control (Khatib, 1987). It works by simulating virtual forces at the tool tip that align the tool to a desired pose. This method typically assumes that the joint angles start close to their final state.

The Jacobian, $\mathbf{J}$, is known from the kinematic model. It relates a 6×1 force-moment

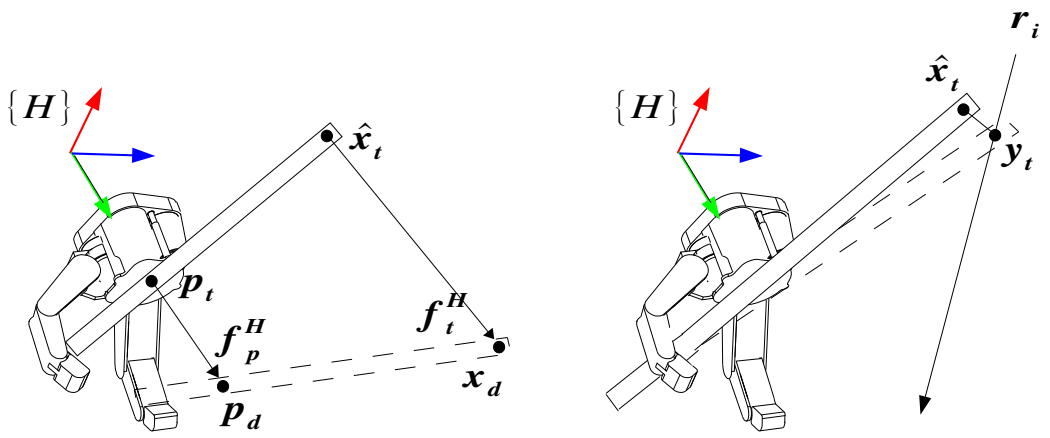


Figure 9-8: (Left) Virtual springs control the position of the palm $\mathbf{p}_t$ and the tool tip $\widehat{\mathbf{x}}_t$ in the hand's coordinate frame $\{H\}$. Forces $\mathbf{f}_t^H \propto (\mathbf{x}_d - \widehat{\mathbf{x}}_t)$ and $\mathbf{f}_p^H \propto (\mathbf{p}_d - \mathbf{p}_t)$ generate a virtual wrench at the end-effector which achieves the desired tip location $\mathbf{x}_d$ and the desired palm location $\mathbf{p_d}$. (Right) Visual tracking of the tool tip defines a ray, r_i, in $\{H\}$. The closest point to $\widehat{\mathbf{x}}_t$ on the ray is $\mathbf{y}_t$. This location is used to visually servo the tip to a target.

wrench at the hand to joint torques as $\tau = \mathbf{J^T f}^W$. Instead of controlling the arm's joint torque directly, we control the joint angle, and our controller takes the form of $\Delta\theta = \sigma \mathbf{J}^T \mathbf{f}^W$ for controller gain σ.

Using this controller, the position and orientation of the tip of a grasped object can be controlled by specifying the virtual wrench $\mathbf{f}^W$ at the end-effector. As shown in Figure 9-8, this wrench is created through two virtual springs in the hand's coordinate frame $\{H\}$. One spring controls the position of the tip by moving $\widehat{\mathbf{x}}_t$ to the target location, $\mathbf{x}_d$. The other spring controls the orientation of the tip by moving the robot's palm, $\mathbf{p}_t$, to the target location $\mathbf{p}_d$.

To compute the net wrench $\mathbf{f}^W$, we first introduce the force-moment transform (Craig, 1989),

$$\mathbf{M}(\mathbf{R}, \mathbf{t}) = \begin{bmatrix} \mathbf{R} & 0 \\ P(\mathbf{t})\mathbf{R} & \mathbf{R} \end{bmatrix}, \tag{9.6}$$

where $\mathbf{P}(\mathbf{t})$ is the cross product operator

$$\mathbf{P}(\mathbf{t}) = \begin{bmatrix} 0 & -t_3 & t_2 \\ t_3 & 0 & -t_1 \\ -t_2 & t_1 & 0 \end{bmatrix}. \tag{9.7}$$

Given the coordinate transform ${}^{A}\mathbf{T}^{B} = [{}^{A}\mathbf{R}^{B}|\mathbf{t}^{A}_{Borg}]$, the force-moment transform relates a wrench in $\{B\}$ to a wrench in $\{A\}$ as $\mathbf{M}({}^{A}\mathbf{R}^{B}, \mathbf{t}^{A}_{Borg})$. We can now relate a virtual wrench at $\widehat{\mathbf{x}}_t$ to $\{H\}$ using $\mathbf{M}(\mathbf{I}, \widehat{\mathbf{x}}_t)$. Likewise, a virtual wrench at the palm relates to $\{H\}$ through $\mathbf{M}(\mathbf{I}, \mathbf{p}_t)$. Therefore, each virtual spring generates a virtual wrench at the wrist as

$$\mathbf{f}^{H}_{t} = \mathbf{K}_t \mathbf{M}(\mathbf{I}, \widehat{\mathbf{x}}_t) \begin{bmatrix} (\mathbf{x}_d - \widehat{\mathbf{x}}_t) & 0 & 0 & 0 \end{bmatrix}^{T} \tag{9.8}$$

$$\mathbf{f}^{H}_{p} = \mathbf{K}_p \mathbf{M}(\mathbf{I}, \mathbf{p_t}) \begin{bmatrix} (\mathbf{p}_d - \mathbf{p}_t) & 0 & 0 & 0 \end{bmatrix}^{T}, \tag{9.9}$$

The spring stiffness $\mathbf{K}_t$ and $\mathbf{K}_p$ is adjusted in a diagonal gain matrix. We can also relate forces in $\{H\}$ to $\{W\}$ using $\mathbf{M}({}^{W}\mathbf{R}^{H}, 0)$. We arrive at the desired virtual wrench acting on the end-effector:

$$\mathbf{f}^{W} = \mathbf{M}({}^{W}\mathbf{R}^{H}, 0) \left(\mathbf{f}^{H}_{t} + \mathbf{f}^{H}_{p}\right). \tag{9.10}$$

This controller assumes a manipulator with a spherical 3 DOF wrist, allowing arbitrary rotation and translation of the hand frame. It also assumes that the manipulator is not near a singularity. Domo's wrist has only 2 DOF and consequently we must assume that the correct orientation is locally achievable with the restricted kinematics. If we initialize this controller from a known pose, then in most cases it is possible to avoid singularities and issues with the restricted kinematics. Related techniques such as Damped Least Squares could be used instead of the Jacobian transpose in order to reduce these restrictions (Buss and Kim, 2005).

9.2.2 *TipServo*

Visual servoing can allow a manipulator to be controlled using a coarsely calibrated hand-eye transform. There are numerous techniques for accomplishing this. An excellent survey of visual servoing for manipulation is provided by Kragic and Chrisensen (2002). In the absence of visual servoing, a robot will typically map a visual scene to a 3D object model in the camera frame, transform this model to the world frame, and then use a Cartesian

space controller to adjust its manipulator relative to the object. This approach requires a precisely calibrated model which is not always easy, or possible to produce.

TipServo adds visual feedback to the *TipPose* module in order to compensate for Domo's coarsely calibrated hand-eye transform which can cause errors in the estimation of the tool tip $\widehat{\mathbf{x}}_t$. When $\widehat{\mathbf{x}}_t$ is estimated, we minimize the pixel error between the feature detections and the projection of $\widehat{\mathbf{x}}_t$ into the image. Consequently, even if $\widehat{\mathbf{x}}_t$ fits the 2D detection data well, it can be inaccurate with respect to a coarsely calibrated 3D kinematic frame. As the arm moves away from the pose where $\widehat{\mathbf{x}}_t$ was learned, these errors often become evident. An open-loop controller like *ToolPose* is susceptible to these errors, especially in fine-resolution tasks such as the insertion of two similar sized objects.

TipServo visually detects the tip and adjusts $\widehat{\mathbf{x}}_t$ online. The algorithm is summarized as follows:

1. Visually detect the tip whenever the wrist rotates as observation $\mathbf{d}_i$.
2. Compute the probability that the observation is actually the tip as $p(\mathbf{d}_i|\widehat{\mathbf{x}}_t, c_i)$ using Equation 9.1.
3. Whenever $p(\mathbf{d}_i|\widehat{\mathbf{x}}_t, c_i) > \epsilon$, for some threshold ϵ, (re)initialize a visual feature tracker at $\mathbf{d}_i$.
4. Use the 2D tracked feature to create an online approximation of the 3D tip location $\mathbf{y}_t$.
5. Substitute $\mathbf{y}_t$ for $\widehat{\mathbf{x}}_t$ in the feedforward *TipPose* controller.

It should be pointed out that the manipulator is servoed in 3D using 2D information. This type of controller is classified by Kragic and Chrisensen (2002) as a position-based visual-servo system. As shown in Figure 9-8, we define $\mathbf{y}_t$ as the point on the ray $\mathbf{r}_i$ that is closest to $\widehat{\mathbf{x}}_t$, where $\mathbf{r}_i$ is the ray in the hand frame that passes through the tracked feature. This is equivalent to choosing the maximum likelihood location on $\mathbf{r}_i$ when the estimation error is distributed around $\widehat{\mathbf{x}}_t$ according to a 3D Gaussian model. Of course, the actual distribution is a complex function of the kinematic state, but this simplification suffices for our purposes.

Robustness is a well known issue for visual servoing, and typically it is achieved through the use of easy to detect markers, homogeneous backgrounds, and CAD models. Kragic and

Christensen (2001) describe the practical issues of visual servoing and describe a method for combining many low-level perceptual features to increase robustness. *TipServo* is able to circumvent some of these traditional robustness issues. First, our use of only motion features allows the use of everyday, cluttered backgrounds. Second, we expect visual observations to be near $\widehat{\mathbf{x}}_t$, allowing us to filter out background observations due to noise. And third, we do not control the tracked image feature $\mathbf{d}_i$ directly, but instead control $\mathbf{y}_t$. A visual servo controller is often susceptible to lags and drop-outs in the feature detector. However, $\mathbf{y}_t$ need only be updated intermittently so long as the error changes slowly across the manipulator workspace. By using $\mathbf{y}_t$, we are also able to reuse the *TipPose* controller.

9.2.3 Results

As shown in Figure 9-9, we tested the *TipServo* controller versus the *TipPose* controller. The robot rigidly grasped a long spoon with a color fiducial marking the tip. In the first experiment, the *TipPose* controller estimated the tip $\widehat{\mathbf{x}}_t$ and traced its projection in the image around an 100 pixel square within a 320×240 image. In the second experiment, visual feedback was introduced into the controller via *TipServo*. The initial estimate of the tip, $\widehat{\mathbf{x}}_t$, was first servoed to the start point on the square. Spoon motion was generated at the wrist for a period of 5 seconds in order to create an initial estimate of $\mathbf{y}_t$. Next, the *TipServo* controller traced the projection of $\mathbf{y}_t$ in the image around the same square. During this time, the spoon's motion was used to update $\mathbf{y}_t$. In both experiments, the desired orientation of the spoon was aligned with gravity and the tip was kept at a fixed depth from the camera of 350mm. The controller error was measured between the location of the fiducial in the image and the 100 pixel square. As the results show, the visual feedback significantly reduced the feedforward controller error. Inspection of the results for the *TipPose* controller shows a horizontal error in the estimation of $\widehat{\mathbf{x}}_t$. The *TipServo* controller compensates for this error.

9.3 Moving from Tool Tips to Everyday Objects

Previously we described a method for the autonomous detection of the tip of a grasped tool and the estimation of the tip's position in the robot's hand. The approach was demonstrated on a range of tools, including a screwdriver, pen, and a hammer. For these objects, we

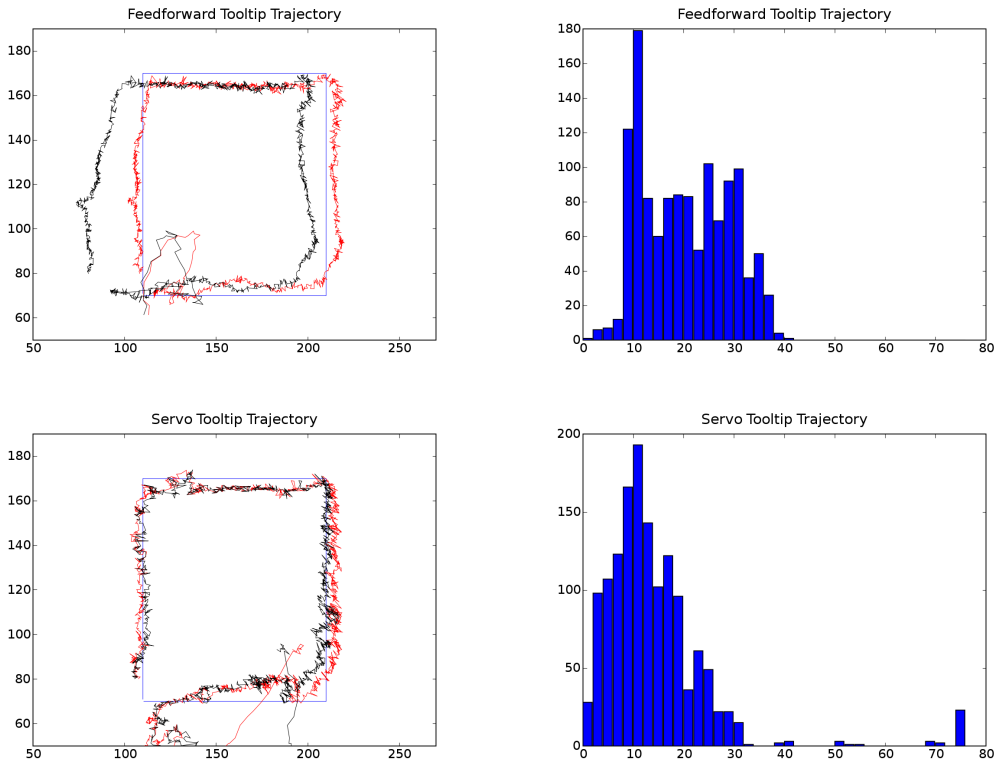


Figure 9-9: Results from execution of the *TipPose* (top) and *TipServo* (bottom) modules. The robot grasped a spoon and estimated its tip location. (Left) Each module then traced the projection of its estimated tip (red) around around an 100 pixel square (blue) in the image. For comparison, the true tip location (black) was sensed in the image using a color fiducial placed on the tip. As we see, the visual feedback used by *TipServo* allows it to reduce its error relative to the feedforward results of *TipPose*. (Right) For each module, we show an error histogram of the pixel error between the desired square (blue) and the actual tip (black).

assumed that a tool is characteristically defined by a single, distal point. However, this characterization does not fit objects such as cups, bowls, jars, brushes, pots, and bottles. These are not tools in a traditional sense, yet they still have a tip or endpoint that is of primary importance during control.

In this section, we extend our definition of a tool tip. We define the tip of a tool as occupying a circular area of some radius. In particular, we use the *InterestRegions* module to detect rapidly moving edges that are approximately tangent to a circle of some radius. This detector performs well on objects that do not have a sharp point, allowing us to expand our notion of the tip of an object to include such items as a bottle with a wide mouth, a cup, and a brush. This feature detector outperforms our previous method on these three objects. It also estimates the scale of the tip which can be used to build a visual model.

We use the same protocol as our tool tip detection method. An object is grasped and rotated at the wrist. Given a pair of sequential images, *InterestRegions* selects the top 10 bins of the interest point histograms that have the highest response. Of these 10 interest regions, we select the top three regions with the highest response. Each interest region is defined by its detection location, $\mathbf{d}_i$, its scale s, its scale radius r_{si}, and the kinematic configuration of the robot c_i. Once n interest regions have been accumulated, the detections, $\mathbf{d}_1 \dots \mathbf{d}_n$, are passed to the tool tip estimation algorithm and $\widehat{\mathbf{x}}_t$ is computed as before.

The size of the object's tip can now be estimated using the scale radius r_{si} of each detection $\mathbf{d}_i$. Given the tip estimate $\widehat{\mathbf{x}}_t$, we compute the distance of the tip from the camera as z_i given kinematic configuration c_i. The estimated tip size for $\mathbf{d}_i$ is then $\frac{2r_{si}z_i}{f}$ for focal length f. Next, we filter for only probable detections such that, given $\mathbf{d}_i$ and threshold ϵ, $p(\mathbf{d}_i|\widehat{\mathbf{x}}_t, c_i) > \epsilon$. This eliminates the influence of noisy detections that are not indicative of the tip size. Finally, the estimated size of the object's tip is taken as the average tip size for all probable detections.

9.3.1 Results

We validated our method on a bottle, a cup, and a brush, as pictured in Figure 9-10. The items were chosen for their varying tip size and length. When the tool is placed in the robot's hand, it automatically generates a short sequence of tool motion of about 200 detections over 5 seconds. For each tool, we compared the interest region detector to the original detector. For comparison, we also estimated $\widehat{\mathbf{x}}_t$ based on hand annotated detections

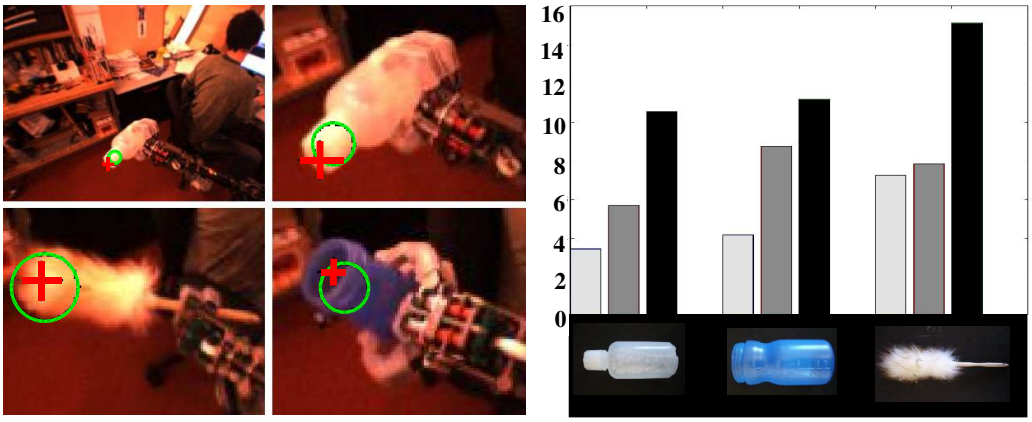


Figure 9-10: Detection and estimation of the generalized tool tip for three test objects. (Left) The upper left image gives an example of the images used during estimation. The movement of the person in the background serves as a source of noise. The red cross marks the hand annotated tip location and has a width equivalent to twice the mean pixel error for prediction over the test set. The green circle is at the tip prediction with a size equal to the estimated tip size. (Right) The mean prediction error, in pixels, for each of the three tools. The 3D tool pose is estimated in three ways: the hand labelled tips (left bar), detections from *InterestRegions* (middle bar), and the maximum motion point (right bar).

of the tip in the image. This provides a baseline for the best we can expect the algorithm to do given errors in the kinematic model.

Figure 9-10 shows the mean prediction error, as measured by the tool tip projection into the image, for the two detectors. The interest region detector significantly improves the predicted location for these three objects that have large, broad tips. We also see that the detector can correctly extract the size of the tool tip. In addition, the ability to extract the extent of the tip in the image enables online modeling of its appearance. Figure 9-11 shows the construction of a pose normalized visual model of the tip. This model facilitates the use of visual features that describe the appearance of the tip over its estimated spatial extent. For example, an HSV histogram of the tip appearance could be learned online to augment the detector's performance.

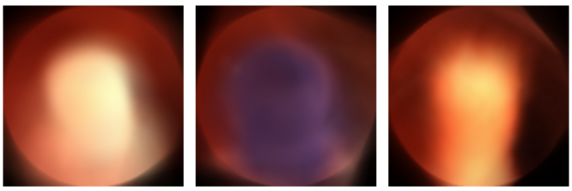


Figure 9-11: These average tip images give an example of acquiring a model of the tip's appearance. For the three test objects, square image patches of the tips were collected using the tip detector, tip predictor, and smoothing of the estimated state. They were then normalized in scale and orientation, based on the kinematic state of the robot, and averaged together.

9.4 Learning the Task Specific Prior

For many tasks, the tip of a tool being used will appear in a canonical region relative to the hand. When we write with a pen or use a hammer, we have a strong prior expectation of where the functional end of these objects will appear. For a given task, a robot could learn this prior from experience, allowing it to more quickly localize the feature and discount detections far from the expected location. Even though the exact location of the feature can vary substantially depending on the object and the grasp, the spatial distribution of feature locations across many task trials can represent a useful prior. Such priors are task specific. For example, a bottle being poured should have a different prior than a bottle being place on a shelf.

Learning the task specific prior allows the robot to predict where the feature might appear in the image. This relates to the work of Torralba (2003), who describes a framework that uses contextual priors to improve visual search and modulate the saliency of image regions. Prior knowledge of where to find a feature can also aid the robot's controllers. In order to localize a tool tip using a uniform prior, Domo must hold the object far from the camera and randomly sample from the full range of wrist postures. Given the task specific prior, Domo can now bring the object closer to the camera and only sample from wrist postures that provide unobstructed views of the expected tip location.

In Section 9.1.3 we defined the prior on the expected tool tip location, $p(\mathbf{x}_t)$, to be uniform everywhere except at positions inside the robot's body or farther than 0.5 meters from the center of the hand. In this section we consider how Domo can learn, off-line, a

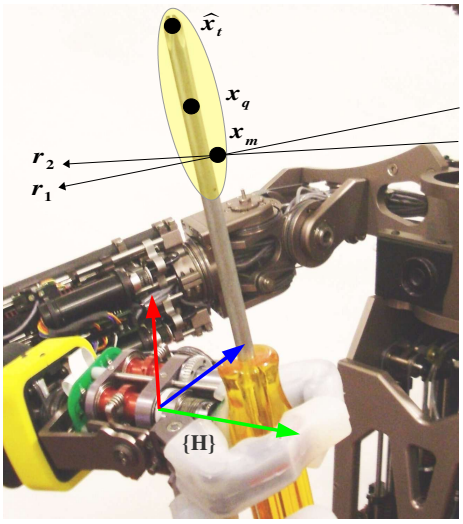


Figure 9-12: The task specific prior, $p(\mathbf{x}_t|q)$, describes a 3D spatial distribution (yellow) within the hand's coordinate system $\{H\}$. The maximum likelihood location, $\mathbf{x}_q$, represents the expected location of a task relevant feature on a grasped object, given the current task q. This prior is used to assist in perceptual estimation of the true feature location, $\widehat{\mathbf{x}}_t$. The distribution is represented as a 3D histogram, and a contribution to the histogram, $\mathbf{x_m}$, is defined as the nearest point of intersection between the rays $\mathbf{r}_1$ and $\mathbf{r}_2$ given two detections of the feature.

non-uniform prior $p(\mathbf{x}_t|q)$ conditioned on the specific task category q .

9.4.1 Density Estimation

As shown in Figure 9-12, the task specific prior, $p(\mathbf{x}_t|q)$, describes a 3D spatial distribution within the hand's coordinate system $\{H\}$. The maximum likelihood location, $\mathbf{x}_q$, represents the expected location of a task relevant feature on a grasped object, given the current task q. We estimate $p(\mathbf{x}_t|q)$ using a 3D histogram. To learn this distribution, the robot manipulates a small set of typical objects used in the task. In this work, we consider tool tip features, though other task relevant features could be used. Tool tips are detected as in Section 9.1.2, where each object is grasped and the robot rotates it while detecting the most rapidly moving point between pairs of consecutive images. Each detection corresponds to a ray in the hand's coordinate frame. In the ideal case, any two rays would intersect at the object's tip. In the non-ideal case, we can take the closest point between any two rays as a candidate

location of the tip. The spatial distribution of these candidate locations, accumulated over the set of task related objects, is then used to represent $p(\mathbf{x}_t|q)$. This approach is similar to the statistical triangulation method of Saxena et al. (2006).

Collecting 3D Detections

We first find the point $\mathbf{x}_m$ that minimizes the distance between two rays. A ray is defined with a start point $\mathbf{s}$ and a point on the ray $\mathbf{p}$. The set of points on a ray are defined by $\mathbf{x}_\alpha = (\mathbf{p} - \mathbf{s})\alpha + \mathbf{s}$ such that $\mathbf{x}_\alpha \cdot (\mathbf{p} - \mathbf{s}) > 0$. We can find the points on the rays that are closest to one another by using standard linear least squares and solving the following equation

$$\begin{bmatrix} \mathbf{p}_1 - \mathbf{s}_1 & \mathbf{s}_2 - \mathbf{p}_2 \end{bmatrix} \begin{bmatrix} \alpha_1 \\ \alpha_2 \end{bmatrix} = \mathbf{s}_2 - \mathbf{s}_1,$$

for $\alpha_1$and α_2. If the points associated with $\alpha_1$and α_2 are valid points on the rays, then $\mathbf{x}_m = \frac{1}{2}(\mathbf{x}_{\alpha_1} + \mathbf{x}_{\alpha_2})$ defines a point that minimizes the squared distance between the two rays. The distance between the rays is also defined as $\|\mathbf{x}_{\alpha_1} - \mathbf{x}_{\alpha_2}\|$.

Now, given a visual tip detection in the image, we can define a ray in the hand frame $\{H\}$ which passes through the detection pixel and the camera focal point. If the robot manipulates an object and generates n visual detections, we can construct $\frac{n(n-1)}{2}$ pairs of rays. We collect all pairs of rays that come within some distance, k_{ray}, of one another. Pairs beyond these thresholds are unlikely to have visual detections corresponding to the same feature. This computation is $O(n^2)$. We accumulate the location, x_m, for all likely ray intersections and repeat the process for all objects in the training set of q, resulting in the training data X_q.

3D Histogram

For each data set X_q, we model the spatial distribution of the feature using a 3D histogram in the hand's coordinate system $\{H\}$. Using this coordinate system allows for visual tip detections to be registered across multiple views of the same object, and across multiple objects that could be applied to the same task. The distribution $p(\mathbf{x}_t|q)$ is estimated using a $k_b \cdot k_b \cdot k_b$ bin histogram that corresponds to a cube centered on the hand with size k_d mm per side. Therefore, $\frac{k_d}{2}$ is the maximum possible extent of any grasped object. We define

the histogram binning function as

$$b_p(\mathbf{a},\mathbf{b}) = \delta(\text{round}\left(\frac{k_b}{k_d}\left(a_x - b_x\right)\right))\delta(\text{round}\left(\frac{k_b}{k_d}\left(a_y - b_y\right)\right))\delta(\text{round}\left(\frac{k_b}{k_d}\left(a_z - b_z\right)\right)), \tag{9.11}$$

where

$$\delta(d) = \begin{cases} 1 \text{ if } d = 0 \\ 0 \text{ otherwise.} \end{cases}$$

The probability distribution is then

$$p(\mathbf{x}_t|q) \approx \frac{1}{\sum_{\mathbf{x}\in X_q} w(\mathbf{x})} \sum_{\mathbf{x}\in X_q} w(\mathbf{x}) b_p(\mathbf{x},\mathbf{x}_t). \tag{9.12}$$

The weighting function $w(\mathbf{x})$ assigns a zero probability to candidate locations that are too close or to far away from the hand, such that

$$w(\mathbf{x}) = \begin{cases} 1 \text{ if } 50\text{mm} < \|\mathbf{x}\| < 500\text{mm} \\ 0 \text{ otherwise.} \end{cases}$$

The maximum likelihood location of the task feature is then taken as the maximal histogram bin, or $\mathbf{x}_q = \text{Argmax}\left(p(\mathbf{x}_t|q)\right)$.

9.4.2 Results

We estimated $p(\mathbf{x}_t|q)$ for four task categories using three objects for each category:

1. Placing a mug, paper cup or watter bottle on a shelf.
2. Covering a surface with a duster, large paint brush, or small hand broom.
3. Pointing a hot-glue gun, a heat gun, or a wooden toy gun.
4. Inserting a large stirring spoon, small stirring spoon, or condiment bottle into a container

These tasks were chosen for the unique, canonical location of the tip of the grasped object during task execution. For each object of each task, the 11 DOF kinematic chain from the camera to the robot wrist was servoed to maintain a fixed pose that ensured tool visibility in the wide-angle camera. The tool was placed in the robot's hand and the 2 DOF of the

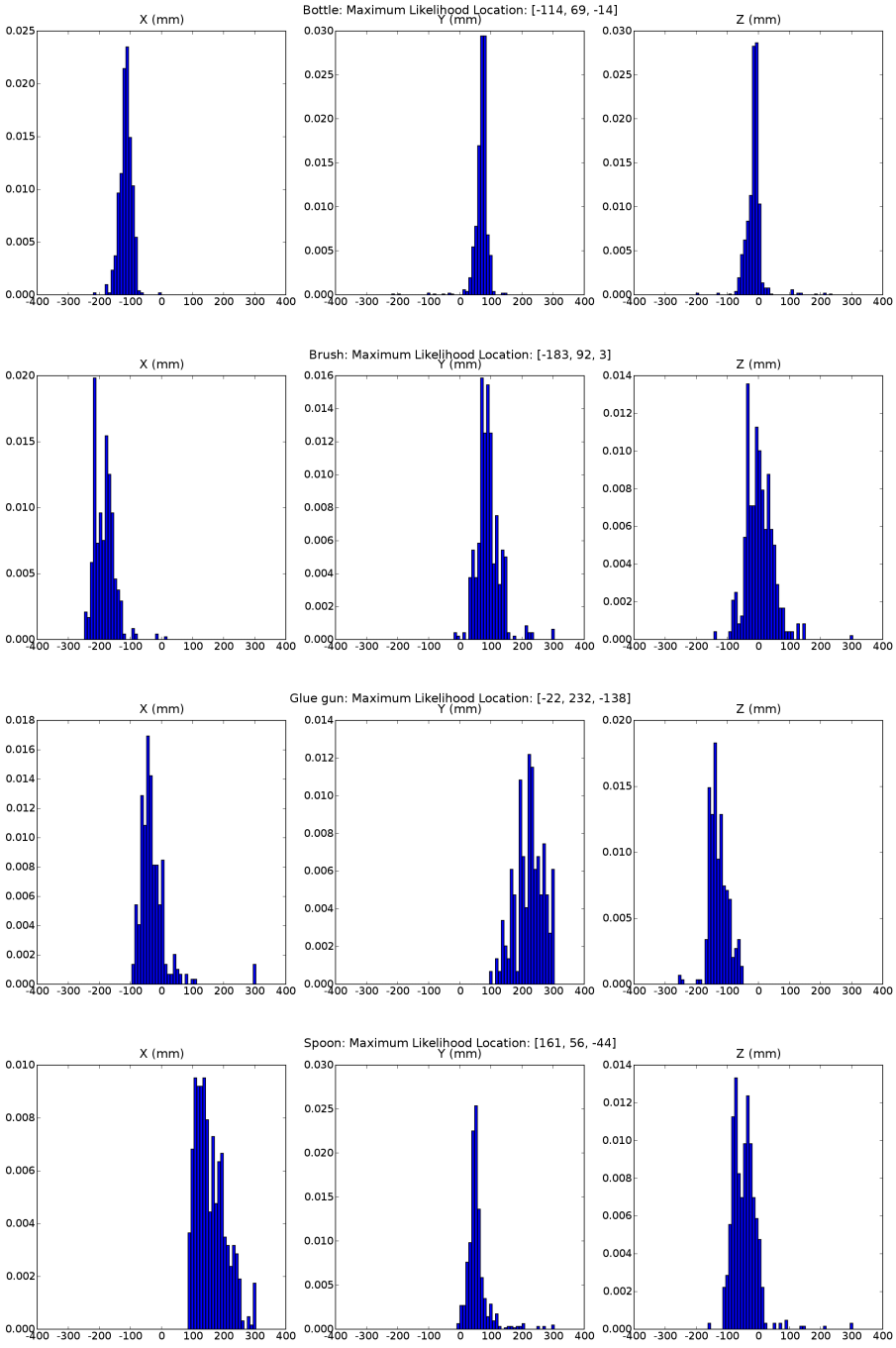


Figure 9-13: Probability distributions for the task specific priors.

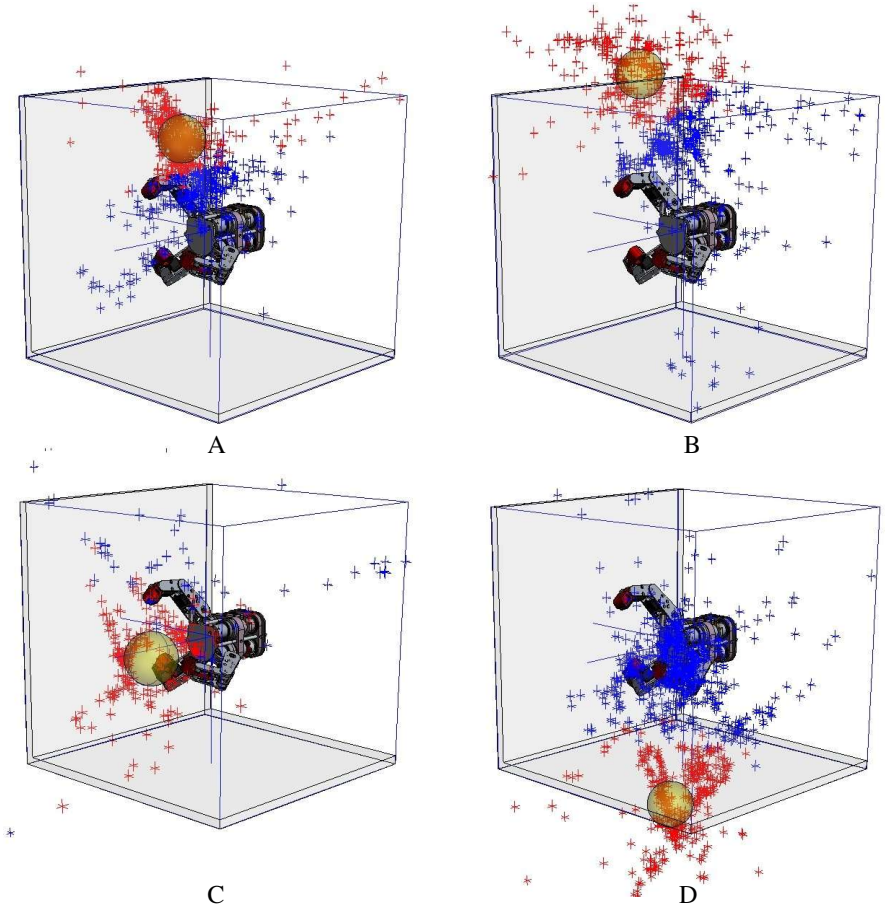


Figure 9-14: Results of learning the task specific priors for four different tasks. Each view shows a scatter plot of the collected 3D data. Each data point is a candidate location of the tip of the grasped object. For each task, three different objects in the same task category were used. The data is portioned into two clusters. The distal cluster (red) is used to estimate the prior for the task feature. The maximum likelihood location, $\mathbf{x}_q$, for each task is shown with the yellow sphere. The cube is 300mm per side for scale. The tasks include placement of a bottle on a shelf (A), using a feather duster (B), pointing a hot glue-gun (C), and inserting a spoon into a cup (D).

wrist were ramped smoothly to random positions in the range of ±60 degrees for a short duration. Visual detection of the fastest moving point was used to generate 50 detections of the object tip. For each of the approximately 1200 pairs of detections, $\mathbf{x}_m$ was computed and added to the dataset X_q subject to $k_{ray} = 10$mm. The histogram was constructed using $k_d = 500$mm and $k_b = 64$ bins.

We implemented an additional constraint on each valid pair of detections for practical reasons. If the two features were detected from similar wrist postures, then the two rays may be near parallel. This magnifies the influence of kinematic and visual error. To reduce this effect, we found it useful to discard detection pairs when the wrist rotated less than 15 degrees between samples. It was also found useful to first cluster X_q into two clusters using K-means. The visual detection algorithm often detects features on the hand when the object tip is obstructed by the hand. Consequently, a significant component of the data can correspond to the robot's hand. By first discarding the cluster that is closest to the hand, we reduce this effect.

Figure 9-14 shows the spatial distribution of the data around the hand for each task category, and the histograms are shown in Figure 9-13. As shown, the maximum likelihood location, $\mathbf{x}_q$, corresponds to a location we would expect for each task. The presence of noise is due to false detections typically resulting from ego-motion of the head, occlusion by the hand, or a person moving in the background. False detections tend to be randomly distributed when viewed from the moving coordinate frame $\{H\}$. In contrast, tip detections are stationary in $\{H\}$ and therefore localized around the true tip.

There are many sources of error that we ignore, including error sensitivity as a function of distance from the camera due to projection, uncertainty about the hand's rotation that will have a larger impact on long objects, and the higher likelihood of intersections at points that are close to the camera. However, this method is computationally efficient, easy to visualize, and produces good results.

9.5 Working with Tips

In the previous sections we presented the components required for a robot to localize and control the tip of a unknown grasped object. This ability is integrated into the *TipUse* module so that whenever the robot is handed a new object, it can quickly find the object

tip and begin controlling it for a task. In this section we describe *TipUse* and the modules required for its execution.

9.5.1 *TipPriors*

Using the task specific prior from the previous section, the *TipPriors* module simply computes the expected tip location $\mathbf{x}_q$ given the current task category q. The active module with control of the manipulator defines q. For example, when *SurfacePlace* is activated, *TipPriors* will compute

$$\mathbf{x}_q = \text{Argmax}\left(p(\mathbf{x}_t|SurfacePlace)\right).$$

The prediction $\mathbf{x}_q$ is then used by any module that *SurfacePlace* might activate such as *TipPose*. Although *TipPriors* limits the generality of *ToolUse* and is not required, it allows for greater robustness and efficiency when estimating $\widehat{\mathbf{x}}_t$. This is convenient for real-time tasks involving a collaborator.

9.5.2 *TipEstimate*

The *TipEstimate* module simply computes $\widehat{\mathbf{x}}_t$ using Equation 9.5. Typically, this is accomplished using approximately $200 - 300$ detections collected over approximately $10 - 15$ seconds. However, if the task category is defined by *TipPriors*, then *TipEstimate* uses the prediction $\mathbf{x}_q$ to initialize the optimization process. This allows increases efficiency and robustness and only 100 detections are used.

Often, a fast online estimation of $\widehat{\mathbf{x}}_t$ is required as the robot executes a task. The majority of time required by *TipEstimate* is spent collecting detections while the arm is held fixed. This can disrupt the flow of human-robot interaction. Therefore, if requested, *TipEstimate* can trade off accuracy for speed by using only a small number of detections as well as the tip prior. In this case, the online method for estimating $\mathbf{y}_t$ in *TipServo* (Section 9.2.2) is used.

9.5.3 *WristWiggle*

The *WristWiggle* module generates rotations at the wrist in order to allow *TipEstimate* to localize the tip. In the absence of tip prediction $\mathbf{x}_q$, *WristWiggle* randomly explores the wrist workspace. The 2 DOF of the wrist are ramped smoothly to random positions in the

range of ±60 degrees. This is often inefficient because the tip may go out of the field-of-view or be obstructed by the hand. Also, if the tip moves too quickly between two images, it may move outside the search window used in the visual block-matching algorithm. Significant motion blur may occur at the tip as well.

However, if $\mathbf{x}_q$ is known, then the *TipPose* module can be used to keep the tip within the field-of-view. Knowing $\mathbf{x}_q$ also allows *WristWiggle* to restrict the tip velocity. If the arm and wrist move between configurations c_1 and c_2 in time t, then the tip velocity in the image is approximately

$$\frac{\|T_{c_2}(\mathbf{x}_q) - T_{c_1}(\mathbf{x}_q)\|}{t},$$

where T_{ci} is the hand-to-eye transform. It is then straightforward for *WristWiggle* to limit the tip velocity as the wrist moves from c_1 to c_2.

9.5.4 *TipUse*

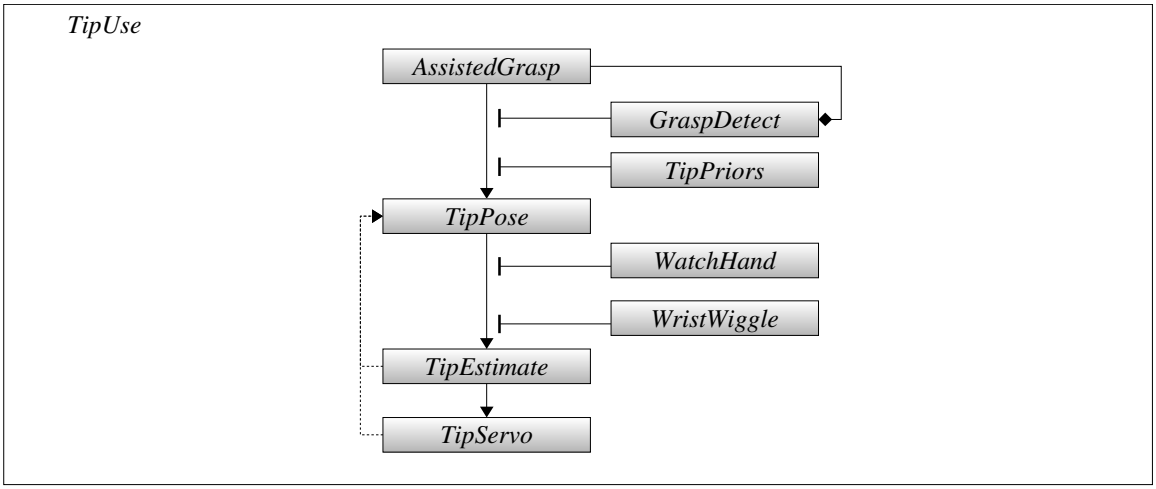
The *TipUse* module enables Domo to take an object from a person, quickly find its distal tip, and control the tip for a task. As shown in Figure 9-15, the *TipUse* algorithm integrates many of the modules we have presented. To begin, Domo acquires an object through *AssistedGrasp* and predicts the tip location, $\mathbf{x}_q$, with *TipPriors*. Given $\mathbf{x}_q$, the *TipPose* module positions the manipulator at a depth such that the tip remains visible by the camera. We compute this depth as

$$z_c = \|\mathbf{x_q}\| \frac{f}{u},$$

for focal length f and field-of-view $2u$ pixels. With the object pointing at the camera, *WatchHand* servos the eye gaze to the hand frame $\{H\}$. The arm and head maintain this posture but the wrist is allowed to move. *WristWiggle* generates small ±30° explorations about this pose, ensuring that the tip remains facing the camera. *TipEstimate* then computes the tip location $\widehat{\mathbf{x}}_t$ using the tip prior. Finally, *TipServo* controls $\widehat{\mathbf{x}}_t$ through a trajectory specified by some other module.

9.6 The Task Relevant Hand

The robot's hand is one of the most significant objects in the robot's environment. Perception and of the hand can also be cast in the framework of *task relevant features*. The importance various features of the hand is dependent on the task. In screwing a cap onto



Ready Wait for *GraspDetect* to signal a power grasp on an object.

Precondition Predict the location of the object's tip using *TipPriors*. Make the predicted tip visible to the camera using *TipPose*. Use *WatchHand* to direct the eye gaze to the hand.

Compensatory Use *WristWiggle* to assist perception by generating motion at the wrist.

Detect Use *TipEstimate* to detect the tip and estimate its true location.

Control Use *TipServo* to visually servo the tip through a desired trajectory.

Figure 9-15: The *TipUse* module allows the robot to take an object from a person, quickly find its tip, and control the tip for a task.

Figure 9-16: Left: The robot's view during execution of *Palm*Servo. This occurs in an everyday environment which includes a shelf, a person, as well as natural lighting and a cluttered background. The green circles mark the convex shape of the open hand, detected by *InterestRegions*. This detection is then controlled using *TipServo*. Right: The motion edges used in the detection of the palm by *InterestRegions*.

a bottle, we can consider the fingertip as the relevant feature. In flipping through papers, the tactile slip between the fingers is of primary importance. In picking up a soda bottle, the contact surface of the palm is critical. In this section we consider this last example. We present the *PalmServo* module which visually servos the contact surface of the palm as it is brought to an object for grasping. This module is an extension of the *TipServo* module presented earlier.

9.6.1 *PalmServo*

Even with precise hand-eye calibration and a 3D model of the hand, it can be necessary to visually detect the hand in the image. The *PalmServo* module uses motion cues to detect the contact surface of the palm. Because it doesn't require specialized knowledge about the hand, the approach is extensible to a wide range of robot hands. Many researchers have created related methods for visual hand detection through motion. Work by Metta and Fitzpatrick (2003b) used image differencing to detect ballistic motion and optic-flow to detect periodic motion of the robot hand. For the case of image differencing they also detected the tip of the hand by selecting the motion pixel closest to the top of the image. Natale (2004) applied image differencing for detection of periodic hand motion with a known frequency, while Arsenio and Fitzpatrick (2003) used the periodic motion of tracked points.

Michel et al. (2004) used image differencing to find motion that is coincident with the robot's body motion. These methods localize the hand or arm, but do not select a specific feature on the manipulator in a robust way. This is important if we wish to use visual feedback to control the manipulator in the image.

PalmServo uses motion cues to detect the palm feature. The palm is defined by the convex shape of the open hand projected into the image. The open hand is always directed towards the camera during execution of *PalmServo*, increasing the likelihood that the feature is visible. The convex shape of the palm is detected using the *InterestRegions* module of Section 6.2. However, we incorporate the hand motion prediction from Section 6.1.2 to increase the salience of moving hand edges. When combined with *InterestRegions*, we can robustly select for convex edges on the moving hand. Using the robot's kinematic model, we can also produce a rough prediction as to the scale and location of the palm feature in the image. This is used to filter unwanted detections on the hand such as the fingertips.

Now, as the open hand moves in the image, *PalmServo* passes detections of the palm to the *TipServo* controller. As shown in Figure 9-16, it visually servos the contact surface of the palm when the hand is open. *TipPriors* also provides a prediction of the palm location, $\mathbf{x}_q$, based on the measured location of the palm feature in the hand frame. We present results for *PalmServo* as part Section 10.1 which describes the *SwitchHands* module.

9.7 Discussion

In this chapter we have considered how a robot can detect and control the distal tip of an unknown, grasped object, including the robot's own hand. The work presented exemplifies our notion of task relevant features. By considering only a single point of an object instead of its complete 3D geometry, we were able to develop specialized algorithms that are robust to many of the environmental factors that typically confuse robots. While this single-point representation could be extended to include a constellation of local features, control of a single point is sufficient for simple manual tasks. Not only does the approach work with a cluttered background, background motion, and under variable lighting conditions, but it is also extensible to a variety of everyday objects. Although our approach used a monocular camera, depth information from stereo cameras could be beneficially integrated to increase the efficiency and robustness of tip localization.

CHAPTER 10

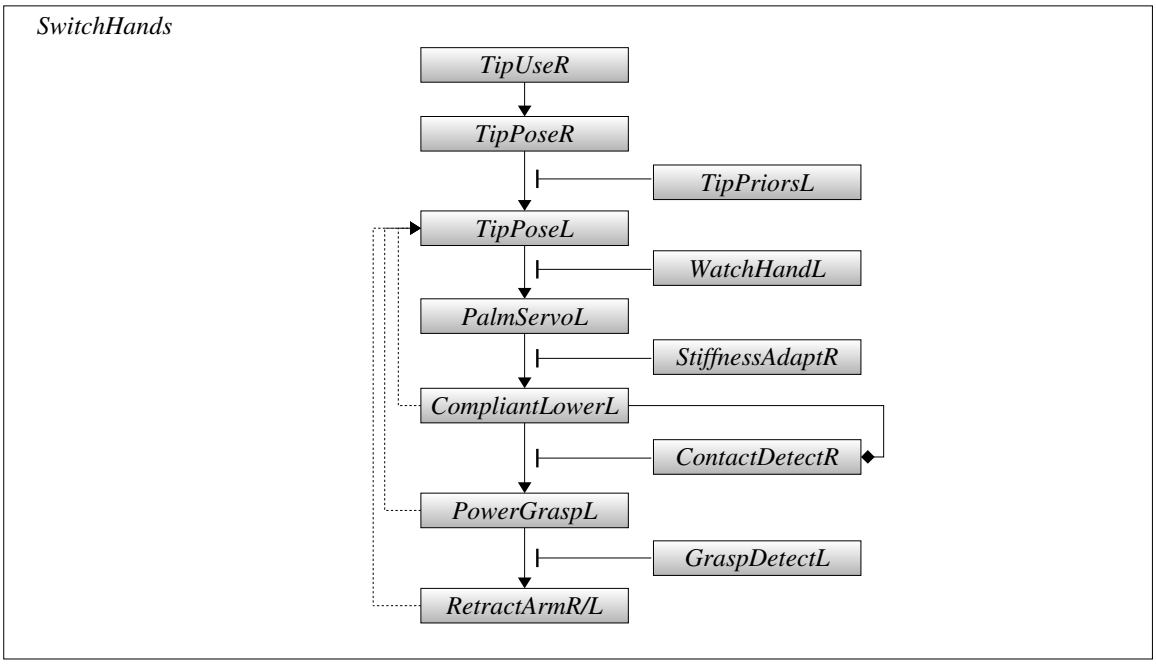
Putting It All Together

In this chapter we demonstrate how the modules presented thus far can be integrated into useful, cooperative manipulation tasks.

10.1 *SwitchHands*

Transferring an object from one hand to another is a fundamental component of a person's manipulation repertoire. For example, a person taking notes while on the phone will transfer the phone to their non-writing hand. A person putting away dishes will remove a plate from the dishwasher with one hand, pass it to the other, and then place it in a cabinet. These seemingly simple acts increase a person's versatility while expanding their workspace of reachable locations. The *SwitchHands* module, shown in Figure 10-1, allows Domo to pass a roughly cylindrical object from one hand to another. The *SwitchHands* algorithm is described in Figure 10-1 in terms of the generic manual skill algorithm of Section 5.3.3.

Let's consider the case where an object is passed from the right hand to the left. *SwitchHands* begins by activating *TipUseR* in order to acquire an object in the right hand and find its distal tip. Instead of servoing the object, *TipPoseR* poses the object at a canonical location and points the object's tip at the camera. Next, *TipPoseL* brings the left hand to a fixed pose relative to the right hand. The top of the hand also points towards the camera.



Ready Using *TipUse*, acquire an object and localize its distal tip.

Precondition Using *TipPose* and *WatchHand*, bring both hands into the field-of-view.

Compensatory Lower the stiffness of the grasping arm to assist contact detection. Use *TipPose* to point the object tip at the camera.

Control Using *PalmServo*, visually servo the palm of the empty hand to be just below the object's tip.

Control Using *CompliantLower*, lower the empty hand through force control.

Control When *ContactDetect* is signaled by the grasping arm, use *PowerGrasp* secure a grasp with the empty hand.

Success Detect success when *GraspDetect* is true for both hands. Release the initial grasp and relinquish control.

Figure 10-1: The *SwitchHands* module passes an object between the robot's hands.

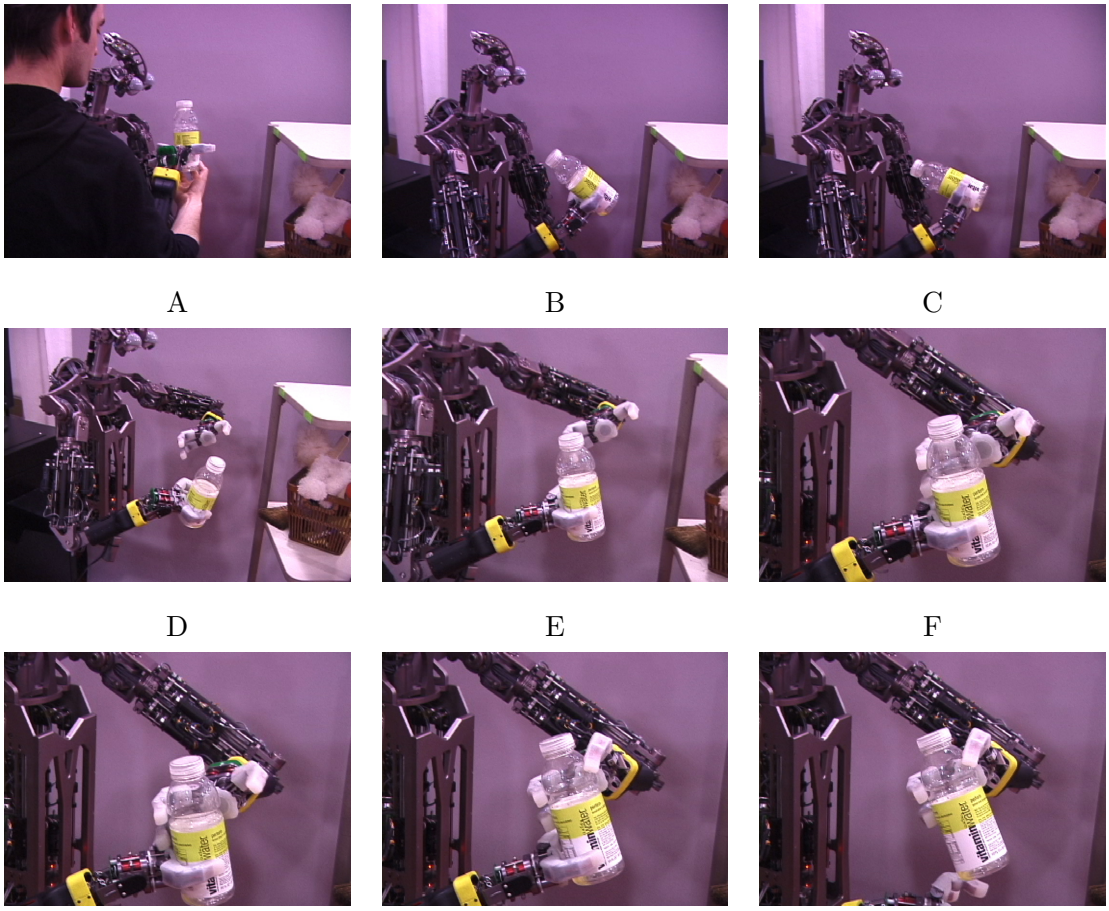


Figure 10-2: Execution of the *SwitchHands* module. The execution time is approximately 20 seconds. (A) A bottle is grasped using *AssistedGrasp*. (B-C) The tip of the bottle is estimated. (D) The arms are moved into canonical poses. (E) The top of the hand and the bottle tip are pointed to the camera. (F-G) *PalmServo* brings the palm surface to the bottle. (H) The palm pushes down on the bottle and *ContactDetect* signals the hand to close. (I) The other hand releases its grasp.

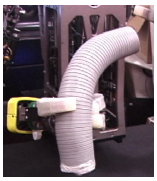
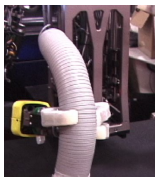
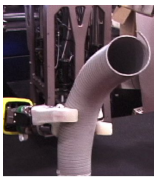
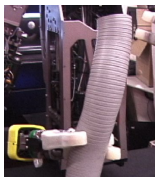
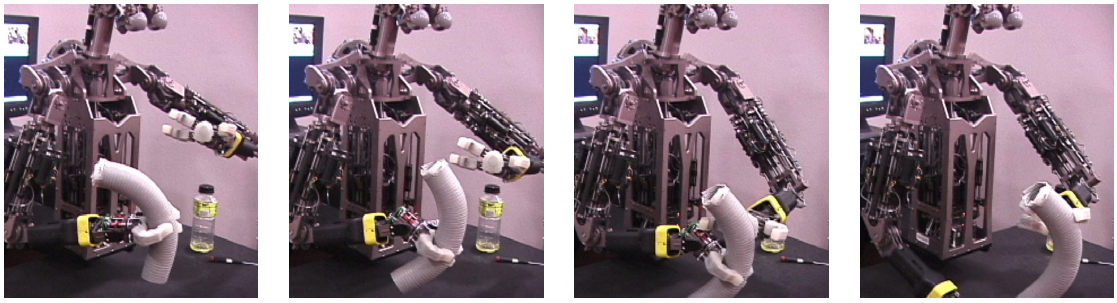
					
	A	B	C	D	E
SwitchHands	3/3	3/3	3/3	2/3	3/3
Open-loop	0/3	0/3	1/3	0/3	2/3

Figure 10-3: Experiment results for *SwitchHands*. (A-D) A poseable dryer hose was bent by about 60° and grasped in one of four orientations by the hand. In the fifth case (E), the hose was unbent and cylindrical. *SwitchHands* and an open-loop controller were tested for three trials on each of the five poses. *SwitchHands* incorporated visual estimation of the object's tip, while the open-loop controller used a fixed estimate for a cylinder. The table shows the number of trials that the object was not dropped. For non-cylindrical objects, visual realignment is critical.

This configuration of the manipulators is shown in Figure 10-2-E. Next, *WatchHandL* keeps the hand within the field-of-view as *PalmServoL* visually servos the palm of the hand to the object. If the object tip is at $[x_c, y_c, z_c]^T$ in the camera frame, the target location for the left palm is just below the tip at $[x_c, y_c, z_c + \Delta z]^T$ for positive offset Δz. When the palm has been servoed into place, *StiffnessAdaptR* lowers the right arm's stiffness while *CompliantLowerL* brings the left hand down onto the object. The displacement of the right hand is detected by *ContactDetectR,* signaling the left hand to form a power grasp on the object. If *GraspDetectL* is signaled, the right hand releases the object and both arms are retracted. If a grasp is not made or contact is not detected, *SwitchHands* reattempts the process. Figure 10-2 shows the execution of *SwitchHands.*

10.1.1 Results

In the ideal case, *SwitchHands* could be implemented without vision, using open-loop control. The robot could simply reach to a fixed location relative to a known object held at a known pose. This approach has been taken by Blackmore and Block (2006), who describe object transfer between two WAM manipulators. Although impedance control is used to



A B C D

Figure 10-4: Visual realignment of the object's tip during one trial of the *SwitchHands* experiment. (A) Initially, the dryer hose is poorly aligned for transfer. (B) After the *TipEstimate* process, *TipPoseR* aligns the hose into the canonical orientation. (C) The left hand is servoed to the hose. (D) Transfer is complete.

allow for modest manipulator misalignment, visual feedback is not used. As they note, variation in the initial grasp on the object, as well as errors in the kinematic calibration, can affect the system performance. On Domo, these issues can cause unwanted collisions between the fingers and the object, dislodging the object before a secure grasp can be formed. Also, if the grasp aperture of the open hand is close to the size of the object, the margin for error is reduced further. While haptic feedback could remedy some issues, visual feedback is necessary for large variations in the object shape and in the grasp.

SwitchHands improves on this open-loop approach by visually estimating the object's pose in the hand and then reorienting the object into a canonical orientation. We experimentally measured this improvement relative to the open-loop controller. As shown in Figure 10-3, the robot passed a poseable dryer hose between its hands. The hose was bent by about 60° into one of four configurations. *SwitchHands* was then tested using the visual estimate of the object's tip and, in the open-loop case, using an ideal fixed tip location of a cylinder. Three trials were conducted for each object pose, and a trial was successful if the object was not dropped. As the results show, the visual realignment of the object is critical for success. However, the open-loop performance is adequate when the hose is cylindrical. In Figure 10-4 we show the execution of one trial when visual realignment is used.

SwitchHands is a robust module for roughly cylindrical objects held in a power grasp, so long as the object extends at least 75mm past the top of the hand. We have used it successfully on soda cans, watter bottles of various shapes and sizes, paint brushes, and a

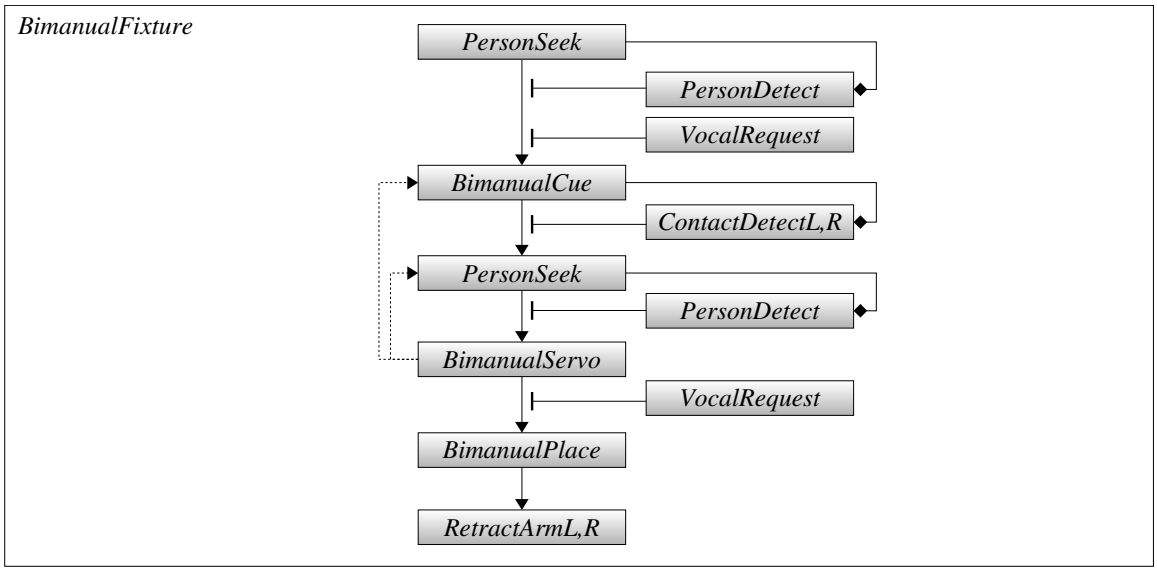
mixing spoon. Of course, many everyday objects in human environments are not roughly cylindrical and would require more sophisticated planning of the grasp, or haptic feedback during contact. In this vein, the work of Saxena et al. (2006) in learning grasp points fits our general approach and could serve as an interesting extension to the *SwitchHands* module. In total, we have executed *SwitchHands* in nearly one hundred trials, largely during the *PuttingStuffAway* task of Section 10.4.

10.2 *BimanualFixture*

In a bimanual fixturing task, a robot holds an object between its two hands in order to assist a person. For example, a robot could hold a laundry basket while the person loads in the clothing, or a robot could hold a serving tray at a dinner table. In this section we describe how Domo controls the position and orientation of a box, grasped between both hands, in order to assist a person loading objects in to it.

Bimanual control of an object's pose and internal forces has been widely studied. For example, Platt et al. (2004) describe a method for composing controllers that can maintain wrench closure while bimanually transporting a large ball. However, these methods typically require that the grasp wrench can be sensed and applied with high-fidelity. This would be difficult in practice given Domo's actuators and kinematic limitations. Instead, we consider a restricted form of the problem and leverage the robot's inherent compliance to maintain stable control of the box. During the grasp, the object is squeezed between the palms of the hands and the palms are aligned to the box surface. The size of the box is unknown but assumed to be of a reasonable size and shape for the manipulators to achieve the task.

The *BimanualFixture* algorithm is described in Figure 10-5 in terms of the generic manual skill algorithm of Section 5.3.3. To begin, when *PersonDetect* and *VocalRequest* signal that a person requires assistance, *BimanualCue* reaches with both arms to the person to cue them to hand the box. *ContactDetect* is used to detect when the person puts the box in one of the hands, at which point *BimanualServo* brings the arms together and forms a stable grasp between the palms. *BimanualServo* continually monitors *PersonDetect* and repositions the box to be near the person. In the next section we describe the *BimanualServo*. If *BimanualServo* loses track of the person, then it waits until *PersonSeek* redetects the person. A person can also remove the box from the robot's grasp during *BimanualServo*



Ready Wait for *PersonDetect* to signal the presence of a person and *VocalRequest* to signal a request for assistance.

Precondition Using *BinmanualCue*, raise both arms to signal the person to hand the box.

Detect Lower the arm stiffness and detect placement of the box between the hands using *ContactDetect*.

Control Use virtual spring controllers and low arm stiffness to stably grasp the box between the palms.

Control Position the box near the person using *BimanualServo*.

Detect Use *PersonDetect* and *PersonSeek* to update the *BimanualServo* controller as the person moves.

Success Upon *VocalRequest*, lower the box onto a nearby table if it is present.

Figure 10-5: The *BimanualFixture* module positions a box, grasped between both hands, to be near a person.

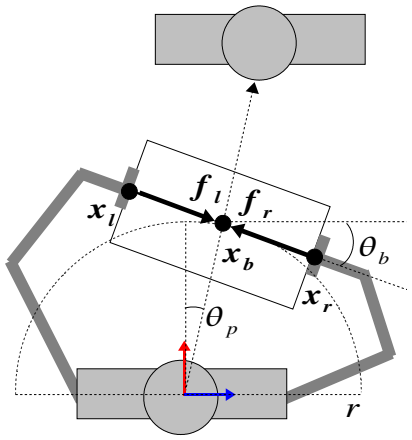


Figure 10-6: During bimanual grasping, the robot keeps a box near its collaborator in order to assist in a task. The object's position, $\mathbf{x}_b$, and orientation, θ_b, are controlled to keep it on the arc of radius r and pointing in direction θ_p. Virtual springs create the squeezing force $\mathbf{f}_l - \mathbf{f}_r$ which keeps the robot's palms, at $\mathbf{x}_r$ and $\mathbf{x}_l$, in contact with the object.

and the robot will reattempt to grasp the box using *BimanualCue*. Finally, when the person tells the robot when the task is complete, and *BimanualPlace* lowers the box onto a table in front the robot and the arms are retracted.

10.2.1 *BimanualServo*

As shown in Figure 10-6, *BimanualServo* controls the position, $\mathbf{x}_b$, and orientation, θ_b, of a grasped box within the X-Y plane of $\{W\}$. We assume the object is already grasped, so the robot's two palms at $\mathbf{x}_l$ and $\mathbf{x}_r$ are in contact with the sides of the box. *BimanualServo* acts to keep $\mathbf{x}_b = \frac{\mathbf{x}_r + \mathbf{x}_l}{2}$ at a fixed radius r from the body and at a height z_b. For the moment, we assume that the palms act as point contacts on the box.

When *PersonDetect* detects a person at $\mathbf{x}^S = [\theta_p, \phi_p, r_p]$, the desired box location is $\mathbf{x}_d = [rsin(\theta_p), rcos(\theta_p), z_b]^T$. The desired box orientation is θ_p such that the box remains tangent to the arc. We infer the box width from the current arm posture as $d = \|\mathbf{x}_r - \mathbf{x}_l\|$, giving the desired location of the right palm $\hat{x}_r = \mathbf{x}_d + [\frac{d}{2}cos(\theta_p), \frac{d}{2}sin(\theta_p), 0]^T$. Likewise, the desired location of the left palm is $\hat{x}_l = \mathbf{x}_d - [\frac{d}{2}cos(\theta_p), \frac{d}{2}sin(\theta_p), 0]^T$.

However, it is not sufficient to just control the pose of the box. *BimanualServo* must also maintain a squeezing force on the box. In the model of Figure 10-6, the manipulators

impart only forces but no moments at the palms due to the non-prehensile grasp. Given this, it is straightforward to show (Williams and Khatib, 1993) that the resultant forces $\mathbf{f}_b$, interaction forces $\mathbf{t}_b$, and moments $\mathbf{m}_b$ on the box due to the applied forces $\mathbf{f}_l$ and $\mathbf{f}_r$ are

$$\mathbf{f}_b = \mathbf{f}_l + \mathbf{f}_r,$$

$$\mathbf{t}_b = \mathbf{f}_l - \mathbf{f}_r,$$

and

$$\mathbf{m}_b = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & \frac{d}{2} & 0 & 0 & -\frac{d}{2} \\ 0 & -\frac{d}{2} & 0 & 0 & \frac{d}{2} & 0 \end{bmatrix} \begin{bmatrix} \mathbf{f}_l \\ \mathbf{f}_r \end{bmatrix}.$$

The equilibrating forces, $\mathbf{f}_b$ and $\mathbf{m}_b$, maintain equilibrium under an external load such as gravity. The interaction force, $\mathbf{t}_b$, corresponds to the squeezing force on the box. Virtual spring controllers create $\mathbf{t}_b$ by applying opposing forces along the ray $\mathbf{x}_r - \mathbf{x}_l$. These forces are defined as $\mathbf{f}_r = -k_f(\mathbf{x}_r - \mathbf{x}_d)$ and $\mathbf{f}_l = -k_f(\mathbf{x}_l - \mathbf{x}_d)$. The gain k_f should be sufficiently large for the grasp to support the weight of the box. The virtual spring forces are generated through bias errors introduced into the manipulator joint angle controller [1]. For example, a bias error $\Delta\theta_r = \sigma \mathbf{J}^T \mathbf{f}_r$ in the right arm will cause the controller of Equation 4.4 to approximately generate force $\mathbf{f}_r$ at the hand given the appropriate scalar σ.

BimanualServo can now simultaneously control the pose and squeezing force on the box. The commands to the left and right joint angle controllers are $\theta_{desired-l} = IK(\hat{x}_l) + \Delta\theta_l$ and $\theta_{desired-r} = IK(\hat{x}_r) + \Delta\theta_r$, where $IK(\cdot)$ is the inverse kinematic function. For a smoothly changing θ_p, this has the desired affect of incrementally adjusting $\mathbf{x}_b$ and θ_b while maintaining a constant squeezing force on the box. Finally, the height of the box can be controlled within the joint limitations of the manipulators by varying z_b.

In practice, the palms do not form point contacts with the box. The interaction between the palm and box is a complex function of the geometry of the box, the palm, the pose of the fingers, and the joint torques of the arm. This uncertainty will often cause an undesired wrench on the box. To accommodate this, the controllers rely on the compliance in the arms and hands, as well as the large contact surface of the palm, to adapt the box pose and stabilize the grasp. An independent 2 DOF wrist controller acts to keep the surface normal

[1] When implementing virtual springs on Domo, we use the joint angle controller instead of the force controller in order to take advantage of the controller's safety limits.

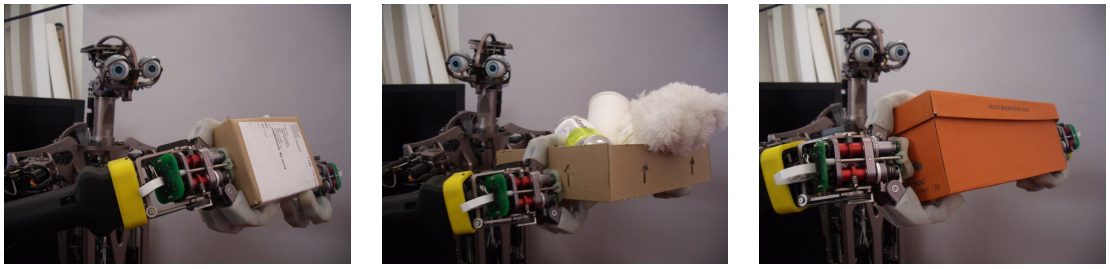


Figure 10-7: The virtual spring controller is able to maintain a stable grasp on boxes of varying sizes. For each of the three boxes tested, a person applied position disturbances to the box of up to 0.2m. The controller was able to maintain the grasp and restore the box position.

of each palm aligned with the ray $\mathbf{x}_r - \mathbf{x}_l$. The controller stiffness is kept low so that the wrist compliance, combined with the skin compliance, allows the box and wrists to realign themselves.

10.2.2 Results

We first tested the ability of the virtual spring to maintain a stable grasp on boxes of varying sizes. As shown in Figure 10-7, three boxes of different sizes were successfully tested. For each, the box was grasped and servoed to a fixed position for 8-10 seconds. A person applied external disturbances of the box of up to 0.2m while the controller maintained its grasp and restored the box position. Figure 10-8 shows the controller response for one of the boxes.

We then demonstrated the complete *BimanualFixture* module. As shown in Figure 10-9, Domo is able to assist a person in putting away objects. The module is also able to fail gracefully in case the box is removed from its grasp or the person is no longer in view. In total, we have tested *BimanualFixture* in dozens of trials. Given an appropriately shaped box, the module encounters few difficulties. However, if it is allowed to work with a person for an extended period of time, the grasp on the box tends to drift, and no mechanism is implemented to adapt to this drift.

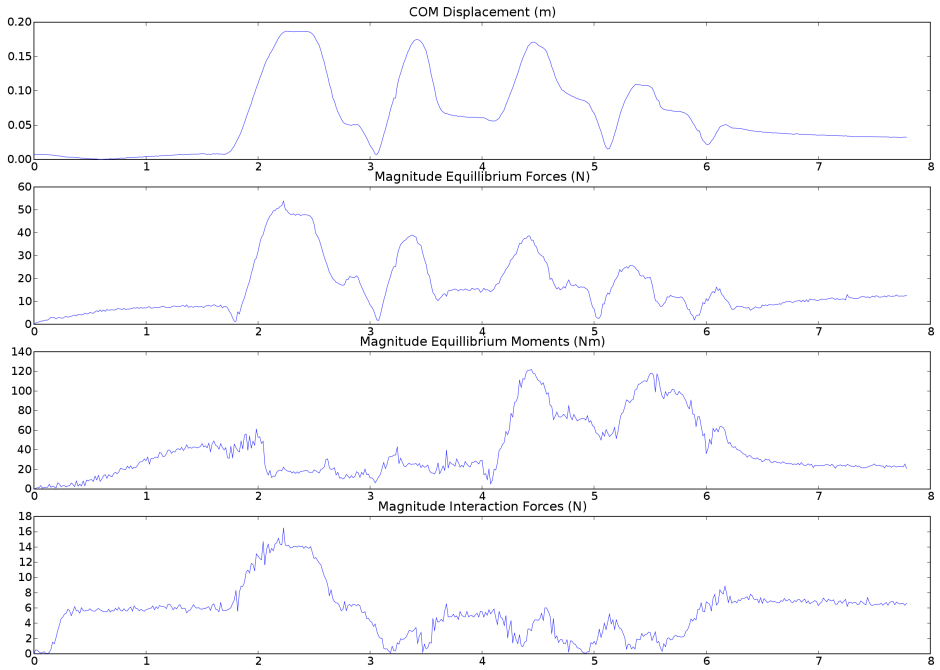
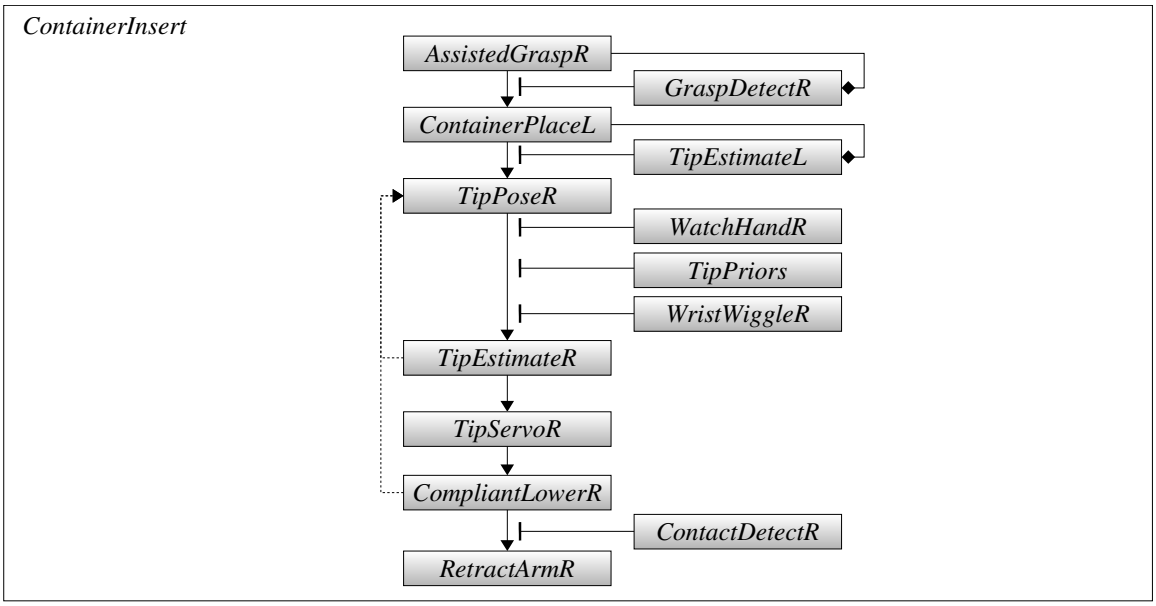


Figure 10-8: The magnitude of the equilibrating and interaction forces to a disturbance during bimanual fixturing. A box was grasped and servoed to a fixed position for 8 seconds. (Top) During this time a person displaced the box from its rest position by up to 0.2m. (Middle) The controller responded with equilibriating moments ($\|\mathbf{m}_b\|$) and forces ($\|\mathbf{f}_b\|$) to restore its position. (Bottom) Interaction forces ($\|\mathbf{t}_b\|$) were maintained throughout the experiment despite the disturbances. Although they momentarily approach zero in several places, a stable grasp is maintained. This is likely due to adaptation of the fingers and wrist to the displaced surface.

Figure 10-9: Domo assists a person in putting things away using the *BimanualFixture* module.



Ready Activate *AssistedGrasp* and wait for *GraspDetect* to be signaled.

Compensatory Using *ContainerPlace*, lower the container onto a table and detect its insertion opening.

Precondition Fixate the gaze on the hand. Using *TipPose*, bring the tip of the utensil , as predicted by *TipPriors,* approximately above the container opening.

Compensatory Using *WristWiggle*, assist tip detection by generating utensil motion.

Detect Estimate the location of the utensil tip using *TipEstimate.*

Control Using *TipServo*, visually align the tip to the center of the opening.

Control Using *CompliantLower*, insert the utensil into the container.

Success Detect contact between the utensil and the bottom of the container.

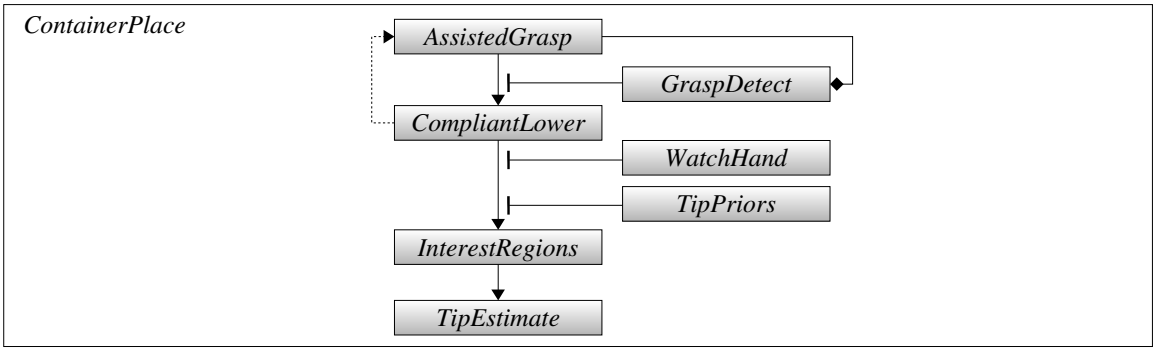
Figure 10-10: The *ContainerInsert* module inserts a utensil grasped in one hand into a container grasped in the other hand.

10.3 *ContainerInsert*

In the *ContainerInsert* module, Domo grasps a common utensil such as a stirring spoon in one hand and a container such as cup or coffee mug in the other hand. It inserts the utensil into the mug and then optionally stirs the contents. This task involves visually guided bimanual insertion of two objects. It is related to the classic peg-in-hole task often studied in model-based manipulation under uncertainty (Lozano-Perez et al., 1984). However, bimanual insertion is much less common. In the bimanual case, the ability to control the pose of the second object allows for greater versatility and sensing during task execution. However, it can also increase the uncertainty of the relative pose between the two objects. A compliant grasp by the second manipulator is also helpful, allowing the pose of the object to adapt to the first.

The *ContainerInsert* algorithm is described in Figure 10-10 in terms of the generic manual skill algorithm of Section 5.3.3. The module begins with a compensatory action, using *ContainerPlace* (described in the next section) to align the container to the top of a table and to localize the container opening. It then uses the *TipPriors* prediction to bring the utensil tip near the opening. The utensil tip is then localized by *TipEstimate*. The estimated tip is realigned to the opening using *TipServoR*, and finally, *CompliantLower* uses force control to complete the insertion.

ContainerInsert is primarily a composition of other modules. The critical step is the localization and servoing of the utensil tip. We adopt an approach similar to Inoue (1979) where the *TipPose* module aligns the utensil at a 45 degree angle to the table. This prevents visual obstruction of the tip by the hand and expands the range of acceptable misalignment when performing the insertion. During *TipServo,* the tip is kept on the visual ray to the center of the container opening. The depth of the tip is then increased along the ray until the tip is just above the insertion location. The manipulator's wrist stiffness is kept low while performing the insertion, allowing for greater misalignment. Although *CompliantLower* does use the interaction forces for insertion feedback, *ContactDetect* signals the utensil's interaction with the bottom of the container or the table. The algorithm for *ContainerInsert* could be applied with little modification to other insertion-type tasks such as pouring a bottle of water. However, kinematic limitations in the manipulator wrist make this difficult.



Ready Activate *AssistedGrasp* and wait for *GraspDetect* to signal a grasp.

Compensatory Using *CompliantLower* and low arm stiffness, lower the object down onto a table, allowing it to align with the surface.

Precondition Using *WatchHand*, direct the eye gaze near to the container opening.

Detect Using *InterestRegions*, visually detect the container opening near the location predicted by *TipPriors*. Using *TipEstimate,* compute the likely opening location in the hand frame.

Figure 10-11: The *ContainerPlace* module assists *ContainerInsert* by localizing the opening of a grasped container.

10.3.1 *ContainerPlace*

ContainerPlace assists *ContainerInsert* by reducing the location uncertainty of a container's insertion opening. The *ContainerPlace* algorithm is described in Figure 10-11 in terms of the generic manual skill algorithm of Section 5.3.3. Like the *SurfacePlace* module, *ContainerPlace* takes advantage of a flat surface to align the object. This is shown in Figure 10-12. Also, the table is used as a stable support during insertion, much like a person resting their cup on a table before pouring a cup of coffee. We require that the container can be grasped with one hand, and that it possesses a roughly circular opening for insertion. We consider the insertion opening to be a task-relevant feature, inclusive of a variety of objects such as drinking glasses, bowls, small boxes, and coffee mugs.

ContainerPlace detects the a roughly circular opening using the *InterestRegions* module. However, the module is configured to ignore motion cues and to find interest regions in

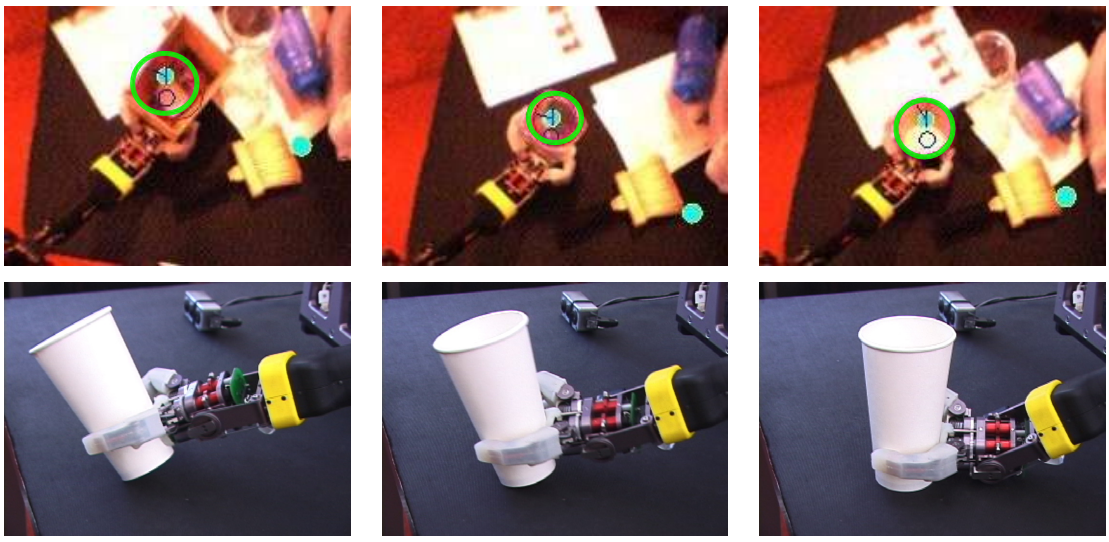


Figure 10-12: Execution of the *ContainerPlace* module. (Top) The *InterestRegions* module finds the roughly circular opening of a box, jar, and bowl. The detector is robust to cluttered backgrounds. (Bottom) A demonstration of *CompliantLower* using the table to align a cup.

single images. The grasp aperture of the hand and the prediction of *TipPriors* are used in selecting the most salient interest region as the opening. Regions far from the prediction or of an unlikely size are discounted. *TipEstimate* computes the likely tip location using this detection given the prior. Figure 10-12 shows opening detections on a variety of objects. Importantly, use of the tip prior allows for robust detection even on cluttered tables.

10.3.2 Results

ContainerInsert can generalize across a variety of insertion objects and containers due to our use of task relevant features. In total, we have executed *ContainerInsert* in nearly one hundred trials with a variety of objects. To demonstrate its performance, we tested *ContainerInsert* in two experiments. In the first experiment, we tested the insertion of a mixing spoon, water bottle, paint roller, and paint brush into a paper cup. In the second experiment, we tested the insertion of the mixing spoon into a paper cup, bowl, coffee mug, and jar. On these objects, the size of the container opening varies between 75-100mm and the size of the tool tip varies between 40-60mm. In each experiment, seven trials were conducted on each object pairing.

In a single experiment trial, the object was handed to the robot in an orientation that

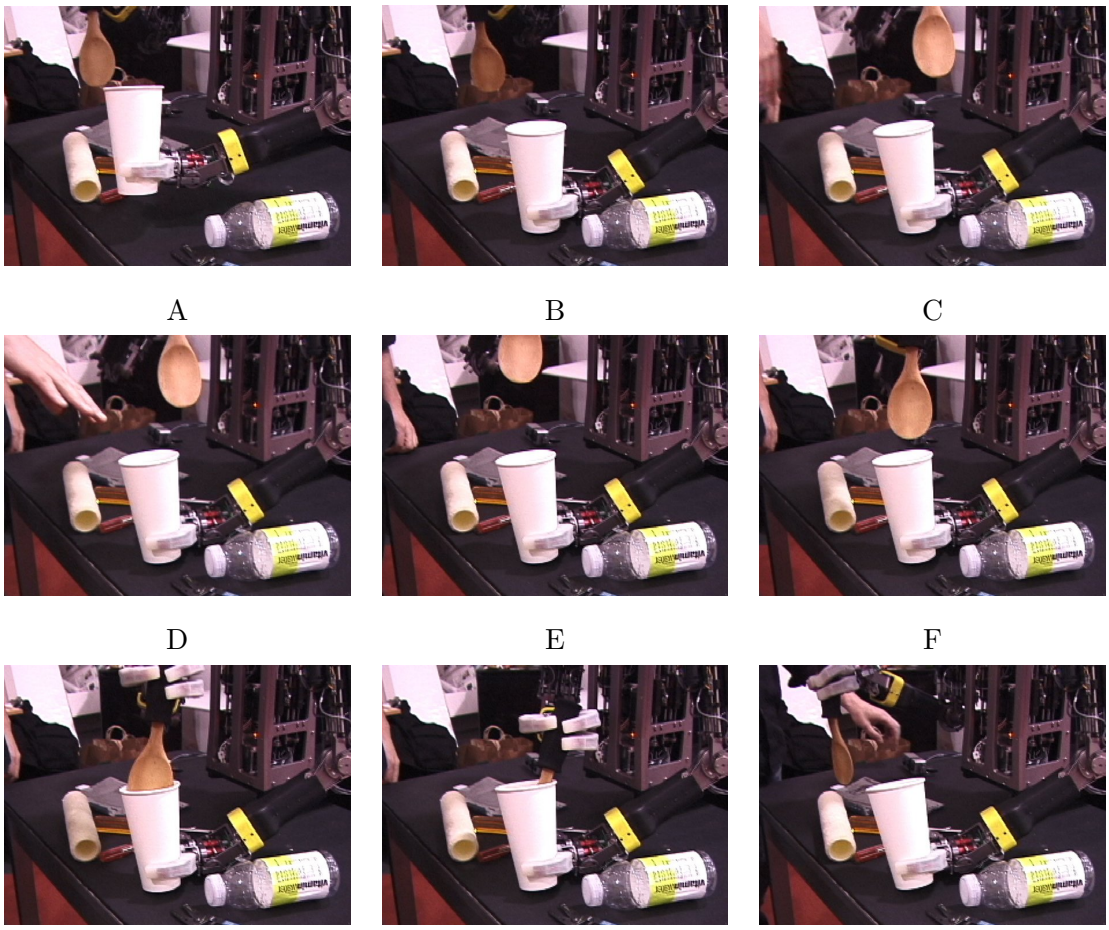


Figure 10-13: Execution of the *ContainerInsert* module with a cluttered background. (A-B) *ContainerPlace* aligns the cup to the table and detects its opening. (C) Using *TipPriors,* a grasped spoon is brought over the cup. (D) The wrist wiggles the spoon and *TipEstimate* estimates the true spoon tip (despite background motion from a person's hand). (E-G) Using the estimated tip, the spoon is brought over the cup by *TipServo* and compliantly lowered in. (H-I) Stirring motion is executed and the spoon is removed.

	Paper cup	Bowl	Box	Coffee mug	Jar
Mixing spoon	7/7	7/7	7/7	6/7	7/7
Water bottle	6/7				
Paint brush	6/7				
Paint roller	5/7				
Spoon (prior only)	1/7				

Figure 10-14: Task success for *ContainerInsert.* In a successful trial, Domo inserted the tool (rows) into the container (columns). For comparison, the last row shows results where the visual detection of the tip was disabled.

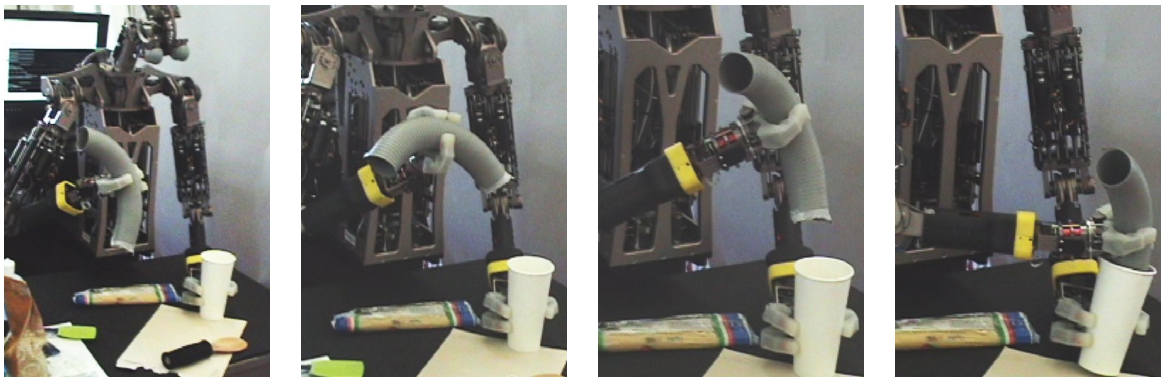


Figure 10-15: Execution of *ContainerInsert* using a flexible hose. The unknown bend in the hose requires the active perception of its distal tip and realignment prior to insertion.

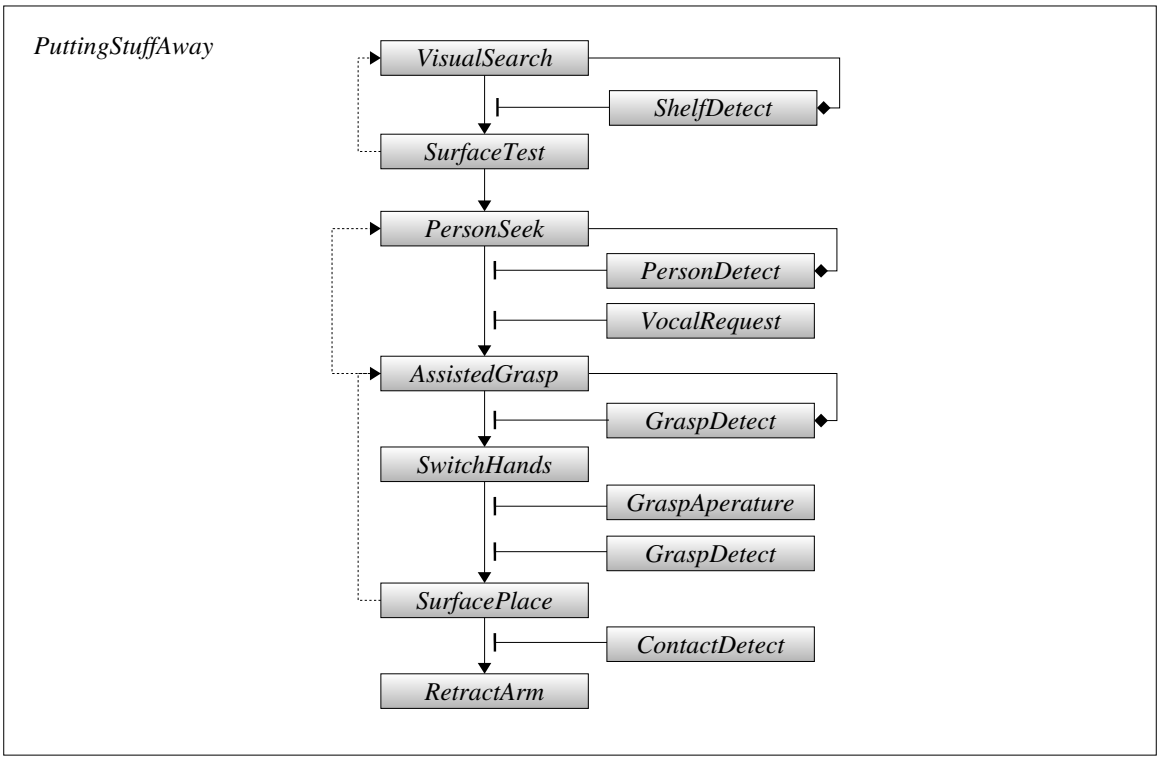
was deliberately varied between $\pm 20^\circ$ along the axis of the hand's power grasp. The grasp location on the object was varied by approximately $\pm$50mm along its length. Each trial took less than 20 seconds to complete and was performed over a visually cluttered table. A trial was successful if the object was fully inserted into the container. The success rates for both experiments are shown in Figure 10-14. As the results show, *ContainerInsert* was successful in roughly 90% of the trials. When the visual detection of the tip was disabled, the success rate fell to about 15%.

As a final example, we tested *ContainerInsert* using a flexible hose. The hose has an unknown bend, making it essential that Domo actively sense its distal tip in order to orient the hose prior to insertion. The execution of this test is shown in Figure 10-15. While *ContainerInsert* can handle the flexible hose in many cases, a single point representation doesn't provide sufficient information to reorient the hose when it has a large bend. In general, if the 3D orientation of the object tip were sensed using stereo or shape features, the object could be better aligned with the container prior to insertion.

10.4 *PuttingStuffAway*

A significant portion of domestic and workplace tasks involve putting stuff away. Examples include unloading a dishwasher, putting groceries in the refrigerator, clearing a desk, stocking items in a supermarket, and emptying a laundry basket. Although these tasks are beyond the abilities of today's robots, an intermediate step is for a robot to assist a person. For an individual with serious physical limitations, this help might allow the person to maintain autonomy in everyday activities that would otherwise require help from another person. For example, an elderly person in a wheelchair might use a robot to put a book back on a shelf.

To this end,the *PuttingStuffAway* module enables Domo to take items from a person and put them on a shelf. The *PuttingStuffAway* algorithm is described in Figure 10-16 in terms of the generic manual skill algorithm of Section 5.3.3. The module integrates the *AssistedGrasp*, *SurfaceTest*, *SwitchHands,* and *SurfacePlace* modules into a single, high-level task. To begin, *PuttingStuffAway* attempts to locate a useable shelf surface with *SurfaceTest*. If it has physically located a shelf but the shelf moves within the SES, then *SurfaceTest* will reestimate its location. Next, the robot awaits its collaborator's request for assistance.



Ready Wait until *ShelfDetect* signals the presence of a shelf.

Compensatory Using *SurfaceTest*, verify the location of the shelf surface.

Ready Activate *AssistedGrasp* to take an item from a person and wait for *GraspDetect*.

Precondition Use *SwitchHands* to transfer the object to the hand nearest the shelf.

Control Using *SurfacePlace*, deposit the item on the shelf.

Success Use *ContactDetect* to signal when the object makes contact with the shelf.

Figure 10-16: The *PuttingStuffAway* module assists a person by putting items on a shelf.

When *PersonDetect* and *VocalRequest* are signaled, *AssistedGrasp* takes an object from the person. *AssistedGrasp* uses whichever arm is closest to the person. *PuttingStuffAway* determines which hand is closest to the shelf. If necessary, it uses *SwitchHands* to transfer the object to the hand nearest the shelf. If *GraspDetect* signals that *SwitchHands* is successful, then *SurfacePlace* puts the item on the shelf. The item is placed upright or horizontally depending on the item's placement stability η.

It should be noted that *PuttingStuffAway* is not simply a temporal sequencing of the modules shown in Figure 10-16. Instead, it involves a carefully designed layering of these modules, allowing for partial recovery from failure and increasing the system's versatility. Throughout, perceptual features such as *GraspDetect* or *ShelfDetect* are constantly monitored to ensure that the state of the world has not changed unexpectedly on the robot. If this is the case, *PuttingStuffAway* can lower the priority of an active module and recovery is attempted without explicit planning. For example, if a person suddenly removes the grasped object from the Domo's hand during *SwitchHands,* then Domo will automatically direct its gaze back to the person and *AssistedGrasp* will wait to be handed another object. Or if the shelf is moved unexpectedly, *SurfaceTest* will automatically retest the location of the surface.

10.4.1 Results

In total, we have executed PuttingStuffAway in over one hundred trials. To demonstrate its performance, we conducted an experiment with *PuttingStuffAway* comprising 18 trials and two subjects, where each trial lasted approximately one minute. A trial consisted of the subject handing Domo a bottle, Domo transferring the bottle to its other hand and then placing it on the shelf. One trial is depicted in Figure 10-17. Each subject performed 3 trials on each of the 3 bottles shown in Figure 10-18. The bottles vary in diameter from 40-75mm and length from $100-200$mm. For each subject, the shelf remained stationary and the *SurfaceTest* module executed only once at the start of the experiment. We measured success using the following criteria:

1. Grasp: Stable grasp after transfer of the bottle from the person to the robot.
2. Switch: Stable grasp after transfer of the bottle between hands.
3. Place: Bottle X was left on the shelf.

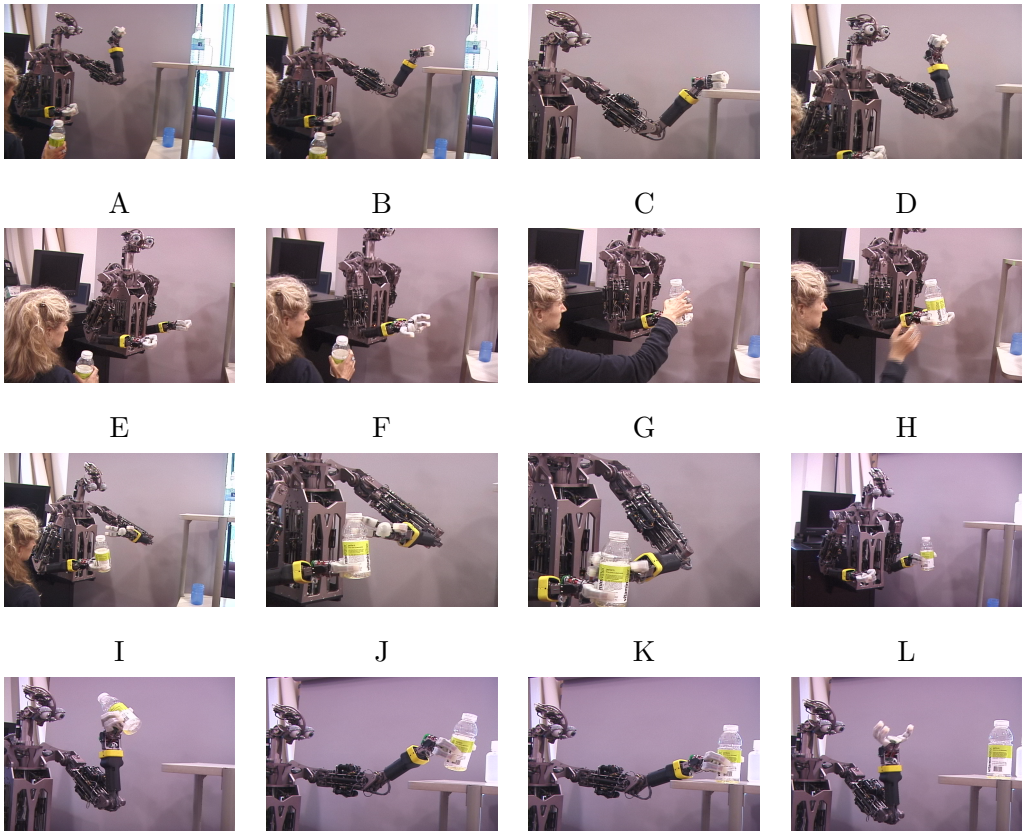


Figure 10-17: Execution of the *PuttingStuffAway* module. (A-D) Hypothesis testing: A shelf is rolled up to Domo. It is visually detected and *SurfaceTest* physically verifies its location. (E-H) Cooperative interaction: A person requests assistance from Domo. They are visually detected and *AssistedGrasp* cues the person to hand it the bottle. (I-L) Expanding the workspace: The shelf is out of the person's reachable workspace but within the workspace of the robot's left arm. *SwitchHands* transfers the bottle from the right to left hand. (M-P) Placement: Domo places the bottle on the shelf at the known location. The manipulator compliance and downward force allow the bottle to align with the shelf.

A B C

Figure 10-18: The three bottles used in the *PuttingStuffAway* experiments.

	Grasp	Transfer	PlaceA	StandA	PlaceB	StandB	PlaceC	StandC
Subject 1	9/9	9/9	3/3	3/3	3/3	3/3	3/3	2/3
Subject 2	9/9	8/9	2/3	2/3	3/3	2/3	2/3	1/3

Figure 10-19: Experiment results for *PuttingStuffAway*, with 3 trials for each of 3 bottles (A,B, and C) and two subjects.

4. Stand: Bottle X was left on the shelf standing upright.

As seen in Figure 10-19, Domo was largely successful at the task for the three objects. One subject was experienced in working with the robot at this task and consequently achieved a higher success rate. Failures were typically a result of insecure grasps being formed during the object transfer phase. Variability in the subject's placement of the object in the robot's hand tended to be amplified by the transfer operation.

We also successfully tested PuttingStuffAway with non-ideal objects such as a cracker box and a paper bag of coffee beans. As shown in Figure 10-20, Domo transfered the bag between its hands and placed it on a shelf. In doing so, Domo unknowingly exploited the intrinsic dynamics of the bag. When grasped at the top, the weight of the beans cause the bag to naturally align with gravity, and therefore the shelf. This behavior is a characteristic of deformable objects that would be difficult to predict using a model-based approach.

10.4.2 Discussion

PuttingStuffAway effectively extends the collaborator's reach, allowing them to place objects in locations that might be difficult or uncomfortable to access without assistance. If this

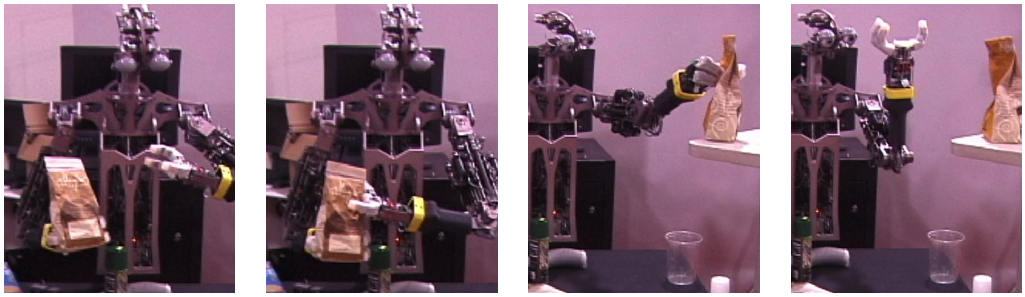


Figure 10-20: Execution of the *PuttingStuffAway* module on a deformable object. A paper bag of coffee beans is transfered between Domo's hands and placed on the shelf. In doing so, Domo unknowingly exploited the intrinsic dynamics of the bag. When grasped at the top, the weight of the beans cause the bag to naturally align with gravity, and therefore the shelf.

skill were combined with a mobile base, the person's effective reach could be dramatically extended. While the module demonstrates end-to-end execution of a useful, cooperative task, there are many ways in which it can be extended. The assumptions of *SurfacePlace* and *SwitchHands* restrict the types of items that can be placed. Richer sensing of the object's features could allow for better placement of non-cylindrical items. Also, all items are placed at the same location on the shelf. For real-world tasks, some planning of the object's placement will be required.

10.5 *HelpWithChores*

HelpWithChores is a final demonstration that integrates all of the modules described thus far. *HelpWithChores* enables Domo to assist a person in tasks that might be expected of a robot working in a domestic setting.

As shown in Figure 10-21, *HelpWithChores* integrates the *BimanualFixture*, *Container-Insert*, and *PuttingStuffAway* modules, among others. These modules run concurrently, allowing a person to vocally request them at any time. The rich integration of these modules allows for a believable cooperative experience for the person. If the person notices that Domo is failing at a task, they can provide vocal (*VocalRequest*) or contact (*ContactDetect*) feedback to alert the robot. If Domo accidentally drops an object (*GraspDetect*), the person can pick it up and ask the robot to grasp it again (*AssistedGrasp*). Alternatively, at anytime

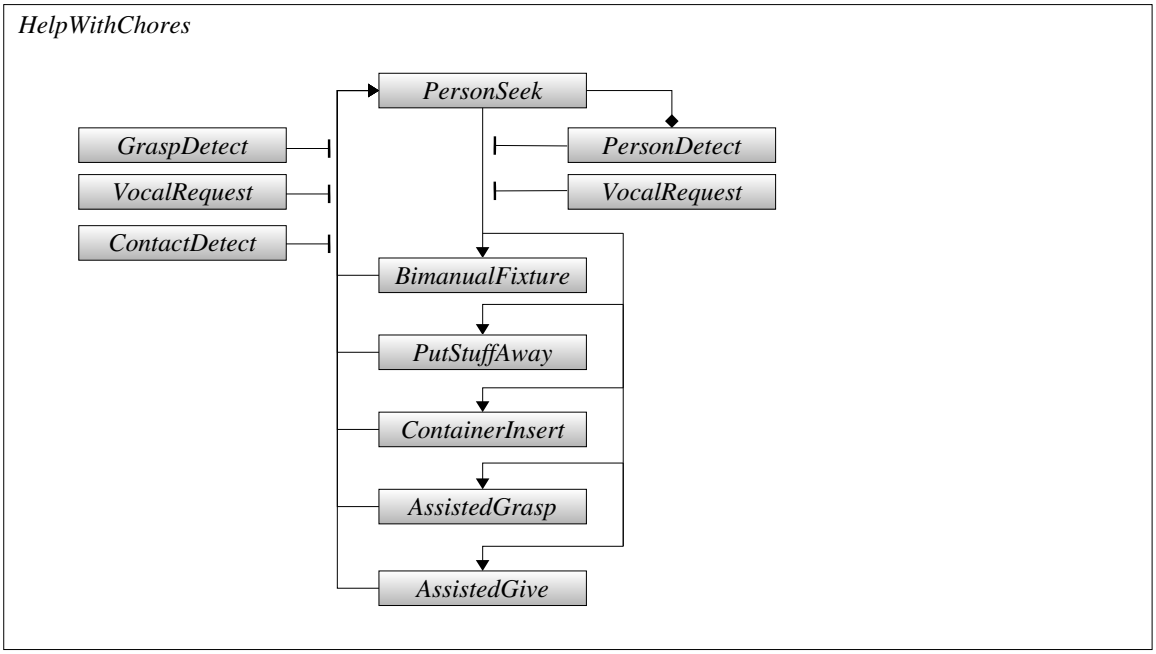


Figure 10-21: The *HelpWithChores* combines the task-level modules *BimanualFixture*, *ContainerInsert*, and *PuttingStuffAway* modules, among others, into a single integrated module. These modules run concurrently, allowing a person to vocally request them at any time. A task can be interrupted prematurely by taking the object away from the robot (*GraspDetect),* asking it to stop (*VocalRequest),* or grabbing the arm (*ContactDetect*).

the person can ask Domo to hand them a grasped object (*AssistedGive*).

In this way, the robot and person work as a team. The person intuitively provides task-level planning and guides the robot's action selection. In return, the robot accomplishes manual tasks for the person. This interactivity during a task is central to the creature robot approach. It requires that a module like *HelpWithChores* is not an algorithmic playback of a manipulation plan. The person should not act as a proxy to a keyboard, making API calls into the code. The robot should at all times be responsive to detected failures during the task and redirection by the collaborator.

With this in mind, one possible *HelpWithChores* scenario is as follows:

1. Domo is positioned at a table cluttered with objects and near a shelf. Domo first physically verifies the location of the shelf.

2. A person asks for help in preparing a drink. They hand Domo a cup and bottle of

juice. Domo "pours" the juice into the cup.

3. The person now hands Domo a spoon. Domo inserts the spoon into the cup and "stirs" the drink.

4. Domo hands the spoon back to the person and then places the prepared drink on the shelf.

5. Next, the person asks for help in putting away groceries. They hand Domo a box of crackers. Domo passes the box to the other hand and puts them upright on the shelf.

6. The person hands Domo a paper bag of coffee and Domo places it on the shelf as well.

7. Now, the person asks for help in clearing off the table. They hand Domo a box and Domo grasps it with both hands.

8. Domo keeps the box near the person as they go about clearing the table into it.

9. Finally, the task is done and Domo lowers the box onto the table.

This scenario was realized by Domo and the author. We accomplished it as one consecutive task, punctuated by vocal requests for the robot, over the course of 5 minutes. Images from the task execution are shown in Figure 10-22 and 10-23. Of course, other scenarios are possible using *HelpWithChores*. For example, Domo could assist a person working on an assembly line by holding a tool tray for the person, putting tools away, holding a tool and then handing it back when the person is ready, and performing the insertion of two parts during assembly.

In total, we have successfully demonstrated the *HelpWithChores* module on Domo dozens of times to visitors of our lab. In each demonstration, the task scenario is composed on the spot, and the demonstration lasts for around five minutes. As we can see, even with the limited perceptual and motor abilities of Domo, a rich behavioral integration allows it to extend beyond simple, experimental demonstrations. We begin to see the potential for Domo to be a truly useful, partner robot.

10.6 Discussion

Leading up to this chapter, we have described the implementation of many SLATE modules that form Domo's basic perceptual and motor skills. In this chapter, we have shown how the

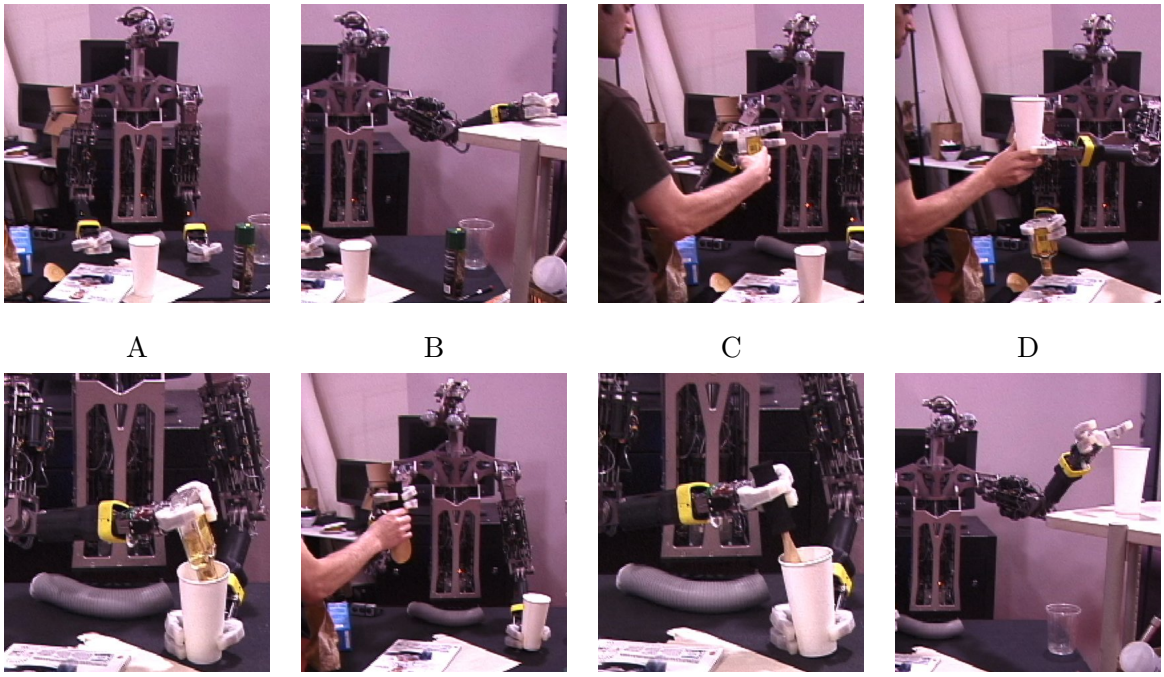


Figure 10-22: Execution of the *HelpWithChores* module during the first half of one consecutive run. In this sequence, Domo assists in preparing a drink. (A) Domo begins at a cluttered table. (B) A shelf appears and *SurfaceTest* verifies its location. (C-D) A juice bottle and cup are handed to Domo using *AssistedGrasp*. (E) *ContainerInsert* "pours" the drink into the cup. (F-G) Now, Domo is handed a spoon and *ContainerInsert* "stirs" the drink. (H) Finally, Domo puts the finished drink on the shelf.

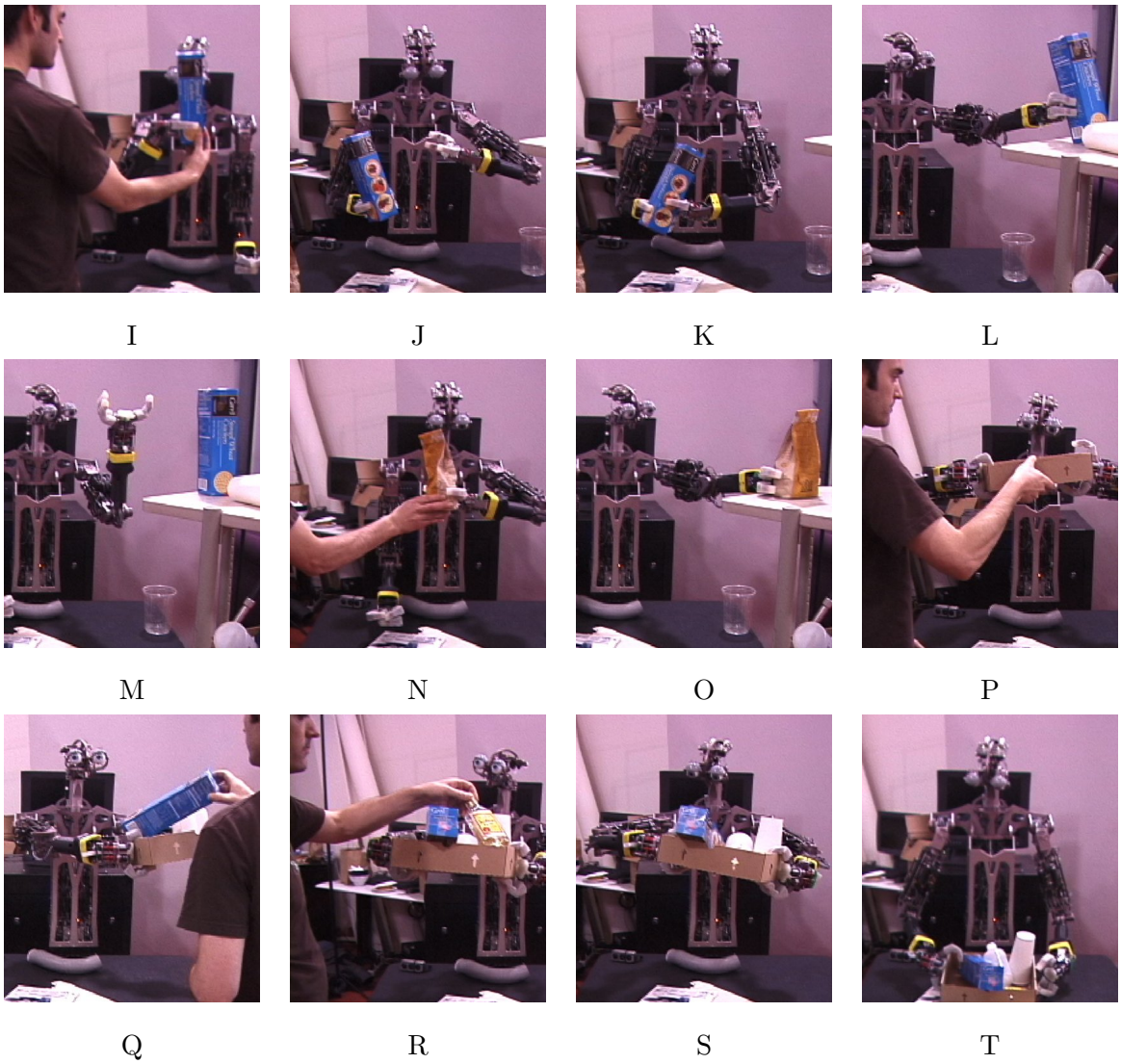


Figure 10-23: Execution of the *HelpWithChores* module during the second half of one consecutive run. (I-L) A box of crackers is handed to Domo's right hand. It transfers them to the left hand and places them upright on the shelf. (N-0) A bag of coffee beans is handed to Domo. This time, it uses its left hand because of the person's proximity. It then puts the bag on the shelf. (P) Domo forms a bimanual grasp on a box. (Q-R) Domo keeps the box near the person as they clean up the table and put items in the box. (S-T) Now that the task is done, Domo lowers the box onto the table.

integration of these modules allows Domo to achieve relatively complex manipulation tasks. While the *HelpWithChores* module allows Domo to accomplish tasks that are reasonably close to actual tasks a person might expect assistance with, there are several immediate extensions that would be useful. First, Domo depends on its collaborator to act appropriately. During *ContainerInsert,* it expects to be handed two objects that can be inserted together. If the collaborator hands Domo inappropriate objects, it will still attempt the task. Basic task knowledge about the expected objects could allow Domo to detect when a task is not going as expected. Second, the objects that *HelpWithChores* can work with are mostly limited to roughly cylindrical shapes. While the physical limitations of Domo's hands and wrists excludes many objects, Domo could work with more diverse objects by extending the perception of task relevant features to include simple 3D shape features.

CHAPTER 11

Conclusions and Future Directions

This thesis suggests one path forward for the emerging area of robot manipulation in human environments. Throughout, we have emphasized the importance for a robot to constantly sense its environment instead of referring to an internal model. Sparse perceptual features can be used to capture just the aspects of the world that are relevant to a task, and many manipulation tasks can be designed through the perception and control of these features. As a consequence, a robot's perceptual system can be specialized and more robust, and its controllers can generalize across particular objects. Although human environments are characterized by uncertainty, we have described strategies for a robot to use its body to reduce this uncertainty. Human environments also exhibit important structure that can be used to a robot's advantage. The people within these settings can be induced to help a robot, and the everyday objects that a robot might work with are designed with common features that can simplify their use. For example, we have shown that just the ability to detect and control the distal tip of a grasped object can afford a robot with a variety of task opportunities.

Although we have demonstrated only the first steps towards robots working in human environments, hopefully we have shown the potential of our approach.

11.1 Contributions

- **Compliant manipulators** Robot manipulators are traditionally stiff and dangerous. We have designed light-weight arms and hands that use Series Elastic Actuators to provide intrinsic safety, force control, and compliance.

- **Creature robo**t Unlike most of today's manipulation robots, robots that work with people will need to exhibit a diverse set of responsive behaviors. We have implemented a behavior-based architecture, SLATE, and have integrated many real-time behaviors into a single system, allowing Domo to seamlessly switch between tasks.

- **Let the body do the thinking** Often, a robot can avoid difficult perceptual problems through clever physical design and action. We have demonstrated several compensatory actions, such as using a flat surface to align an object, that allow Domo to take actions that reduce its uncertainty about the world. We have also demonstrated the use of passive and active compliance in a manipulator to increase a task's robustness.

- **Cooperative manipulation** Manipulation is usually thought of as the isolated activity of just the robot. We have shown that a robot can intuitively leverage the advanced perceptual, motor, and planning skills of a collaborative partner to assist in accomplishing a task.

- **Task relevant features** Current approaches to manipulation often assume, or require the recovery of, a 3D model. We have demonstrated an alternative approach, where the the robot views the world as a sparse collection of simple perceptual features relevant to a task.

- **Everyday environment** Roboticists do not have the luxury of engineering human environments, so robots will have to function in the world as it is. The results in this thesis were obtained in an unmodified office environment characterized by a large variability in lighting, cluttered backgrounds, and largely unrestricted interactions with people. To this end, we have developed a visual attention system that relies on robust motion features.

- **Everyday objects** Many techniques in robot manipulation depend on prior models

of an object or fiducial markers. Consequently, they do not always extend to the variation found in everyday objects. Our methods were demonstrated on a variety of everyday objects without prior models. Although some restrictions are required on the size and shape of manipulable objects, we do not require the use of special markers or backgrounds (with the exception of the shelf edge detection).

- **Everyday tasks** Ultimately, we want robots to do work that is useful to people. We have demonstrated a single end-to-end task where Domo assisted a person with domestic chores, including preparing a drink by "pouring" into and stirring the contents of a cup, putting grocery items away on a shelf, and holding a box for the person as they clean up.

The spirit of this work, as well as the contributions related to the *TipEstimate* and *InterestRegions* modules, was developed in collaboration with Charles Kemp, and we refer the reader to Kemp and Edsinger (2005, 2006a) for the results of this collaboration.

11.2 How Far Does This Extend?

It is important to consider the generality of our work, where it works best, and where it breaks. In this thesis, we have addressed the following aspects of human environments:

- Generalization across objects within an object class
- Variability in lighting
- Unpredictable dynamics of human interaction
- Cluttered backgrounds
- No prior 3D models of objects or the environment
- No environment engineering

However, we have not addressed several important aspects of the domain, including:

- Naive collaborators
- Multiple environments

- Autonomous grasping
- Autonomous task-planning
- Generalization across objects outside of an object class

The unaddressed areas of our work point to three categories of limitations: the robot itself, the reasonable scope of work, and intrinsic limitations of the approach. In the first category, the robot's design and sensory system restricted certain research directions. For example, the lack of dense tactile sensing, visual depth perception, fine resolution force sensing, as well as physical limitations in the manipulator prohibited exploration of haptics guided control and autonomous grasping. In the second category, many areas of work are simply beyond the realistic scope of a single dissertation. With more time, multiple environments could be tested, richer perception of social cues could be implemented, and naive collaborators could work with the robot. In the third category of limitations, we find research problems that are difficult to address given our framework. Let's consider what these limits might be.

One potential limitation is in the manipulation of out-of-class objects. We have demonstrated our approach on objects within a class. For example, the *SwitchHands* module works with objects that are roughly cylindrical, such as a box of crackers or a bag of coffee. However, if the collaborator were to hand the robot a bath towel, the algorithm would fail. Although we assume the collaborator will hand objects within a class, it is a reasonable extension to detect when out of class objects are handed. This detector could be learned, as *SwitchHands* automatically categorizes objects in to those that can be passed between hands, and those that can't. While this would improve the system robustness, we would prefer if *SwitchHands* would succeed on out-of-class objects. This is a difficult task as it requires generalization of the robot's manual skills beyond their currently specialized form. This could possibly be achieved using a suite of multi-modal, task-relevant features that extend beyond a single category.

Another potential limitation involves coordinating and planning complex tasks. Domo relies on a collaborator to coordinate the robot's manual skills over time. In this way, the collaborator performs the task planning. However, we would like to incrementally expand the robot's autonomy, and this will require that it assume some of the task planning responsibility. Local task planning is possible within our framework. For example, when *SurfacePlace* activates *AssistedGrasp* but remains in a wait state until *GraspDetect* is sig-

naled, it appears that Domo plans to first grasp an object before placing it on the shelf. This behavior emerges locally from the control system and is not part of a global plan. However, this form of planning may be difficult to extend to more complex situations. Consider the task of loading a dishwasher. To optimize the use of space, a person will first load the smaller items and then place the large pots and pans in last.

In summary, our approach provides opportunities to expand the variety of objects, environments, tasks, and collaborators that it can work with. Although many of its components are hand designed, it also provides opportunities to integrate online learning. For example, the system could learn the significant perceptual features of a task, learn the class of objects that a manual skill is appropriate for, or learn to coordinate the interaction of behavior modules within a controller.

11.3 Lessons Learned

In developing this work, we have learned a number of implicit lessons that are worth making explicit. These include:

Assumptions about the world

Assumptions about the state of the world are usually imprecise, and often incorrect. This is a theme taken up by Brooks (1991b) and particularly true for our work. For example, after *SwitchHands* executes, we assume that the regrasped object is appropriately aligned in the hand. Misalignment can cause *SurfacePlace* to subsequently fail. Of course, it is often difficult to avoid these assumptions, as the alternative requires constant sensing and adaptation.

Communication through the world

Communication through the world, as described in Section 3.1, can limit the implicit assumptions a module makes about the world's state. For example, *SurfacePlace* monitors *GraspDetect* to determine when to execute. If the object is dropped in transit to the surface, *SurfacePlace* is automatically deactivated and *RelaxArm* takes control of the arm. Communication through the world can also enforce modularity within the design, as it avoids the complexity of internal wires between modules.

Testing on real hardware

It is important to test algorithms on real robot hardware as much as possible, particularly in manipulation. Even with ideal objects, the contact geometry and forces between the object and manipulator will be far from the ideal. A grasp on an object is rarely a true rigid grasp, and slippage can introduce unpredictable effects. In some cases, the sensing resolution required by an algorithm will be unrealistic, particularly when derivatives, such as Jacobians, are being used.

Dealing with failures

Detection and recovery from failure is perhaps more difficult, and as important, as the manual skill itself. It is usually possible to achieve a manual skill that works within a limited range of environment variability, but outside of this range we would like the system to at least detect failure. In our approach, we can detect basic failures through detectors such as *ContactDetect* and *GraspDetect*. Rather than explicitly consider all failure modes of a manual skill, we use these detectors as checkpoints during execution. If the checkpoint fails, then the controller can revert to a previous stage and try again, or relinquish control. This allows SLATE to handle the exception by activating the next-highest priority module.

Feedforward versus feedback

Within a manual skill, we often switch between feedforward and feedback control. Perceptual errors are often amplified by a feedback controller and often feedforward control is more reliable. Of course, feedback control provides greater precision but it requires a stable percept. In the generic manual skill algorithm of Section 5.3.3, we first use a feedforward precondition action, localize the feature, and then use either feedforward or feedback control based on this localization. With *SurfacePlace* we use feedforward control during the reach phase, while with *ContainerInsert* we use visual feedback to provide precise control during insertion.

Leveraging human environments

We have shown that an object's tip is a useful feature for robot manipulation. Consider that we defined the tip as the furthest point from the center of rotation of a grasped object,

and we found that this corresponds to the functional feature of a wide class of objects. This relationship is due to the way everyday objects are designed, and because of the social conventions surrounding how people hand objects to each other. It suggests that we might view much of everyday, human manipulation as exploiting such affordances. We can design robots to leverage these same affordances.

11.4 Going Forward

Rather than pushing for highly complex manipulation tasks, we envision developing foundational manipulation skills that may be coarse, but are capable within human environments. As advances are made in sensing, machine vision, machine learning, and actuator technologies, the robustness and generality of these skills can be augmented and expanded. This provides a smooth path towards full autonomy, where the robot's dependence on a human collaborator is incrementally reduced. We now describe a few directions that deserve immediate attention.

11.4.1 A Suite of Task Relevant Features

In the course of this thesis, we have introduced several task relevant features such as the edge of a shelf, the opening of a container, and the tip of a tool. There are many other features that characterize human environments and would be worth pursuing, including flat surfaces, handles, alignment with gravity, interior cavities, as well as alignment features for stacking objects. A modular suite of perceptual detectors and controllers for these types of features would be of great use, both for Domo, and for other robotics researchers. We would also like to pursue a task relevant approach to tactile and auditory perception. One example in this direction is the work of Torres-Jara et al. (2005), where the tapping sound of a hand making contact with an object was used to discriminate between objects.

11.4.2 More Dynamics, More Adaptation

Robots in human environments require significant adaptation to unexpected dynamics. While Domo can adapt to some unexpected events such as dropping an object, or a person grabbing its arm, there are many circumstances that it does not adapt to. For example, during execution of the *SurfaceTest* module, the hand will sometimes get trapped beneath

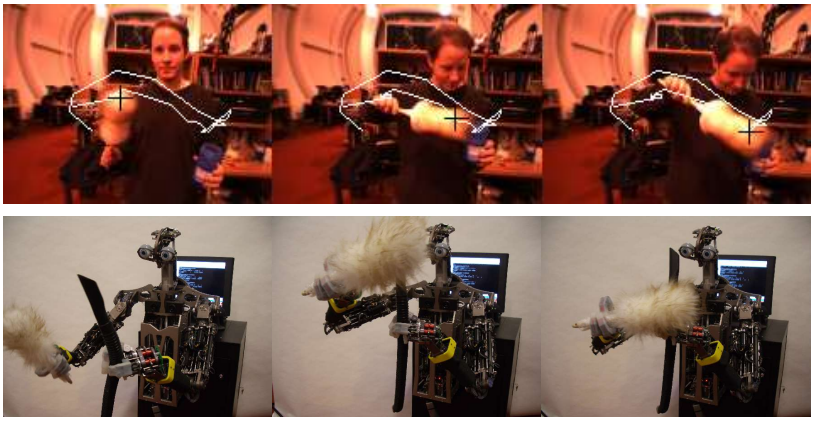


Figure 11-1: Preliminary results in task learning from demonstration. (Top) Domo tracks the relative location of the tip of a duster and a bottle as a person demonstrates a cleaning task. (Bottom) Because the relative tip trajectory is largely independent of the objects used, Domo uses the trajectory to reproduce the cleaning task using the duster and a flexible hose.

the surface as it reaches out. Although this event could be detected and explicitly considered, it would be preferable for a behavior to automatically subsume control of the arm and respond appropriately. This type of adaptivity is difficult. It requires richer perception and a better understanding of how to build integrated, behavior-based systems.

Ideally, these adaptive behaviors would be learned. For example, the perceptual features indicative of failure within a manual skill could be learned through experience. Currently, these features are hand selected. As the suite of perceptual features is extended, task failure could be learned as a complex function of the proprioceptive state and the state of the perceptual detectors. Along these lines, recent work by Jenkins et al. (2006) on Robonaut has demonstrated manifold learning of a task-success detector for teleoperation tasks. Learning an adaptive control policy within our framework is an important extension. Although our controllers are not composeable as in the work of Platt et al. (2003c), we could learn a policy for the sequential activation of motor modules based on feedback from perceptual detectors. This could improve the adaptivity of the manual skills beyond the performance of our hand designs.

11.4.3 Learning from Demonstration

The development of a task such as *PuttingStuffAway* requires a lot of experimentation and takes a lot of time. It would be preferable for task level behaviors to be acquired autonomously, perhaps through human demonstration. In the long run, learning from demonstration could serve as an intuitive way for people to program the robots they work with.

Our approach to manipulation is well suited for learning from demonstration. By focusing on task relevant features during both the demonstration and the execution of a task, a robot could more robustly emulate the important characteristics of the task, generalize what it has learned, and ignore irrelevant features such as the particular kinematic configuration of the demonstrator. This type of approach has already been investigated by Pollard and Hodgins (2002). As shown in Figure 11-1, we have begun to pursue learning from demonstration in preliminary work (Edsinger and Kemp, 2007).

11.5 Final Remarks

This thesis is guided by a desire to see robots help people with the work of everyday life. Robots that can work alongside us in our homes and workplaces could extend the time an elderly person can live at home, provide physical assistance to a worker on an assembly line, or help with household chores. In many ways, robot manipulation, and in particular, robot manipulation in human environments, is the next great challenge for robotics.

APPENDIX A

Module Summary

1. *AssistedGrasp:* Grasps an object by cueing a person to hand it.
2. *AssistedGive:* Gives an object to a person by reaching towards them.
3. *BimanualCue:* Reaches with both arms to a person.
4. *BimanualFixture:* Assists a person in putting items away by holding a box near them.
5. *BimanualPlace:* Lowers an object grasped between two hands onto a table.
6. *BimanualServo:* Grasps a box with two hands and positions it near a person.
7. *CameraReach:* Reaches along the optical axis of a camera.
8. *CompliantLower:* Uses force control to drop the arm in the direction of gravity.
9. *ContactDetect:* Detects external contact with the arm.
10. *ContainerPlace:* Places a grasped container on a table and detects its opening.
11. *ContainerInsert:* Visually servos the tip of a grasped object into a grasped container.
12. *GraspAperture:* Estimates the size of the opening of the hand.
13. *GraspDetect:* Detects when the hand is grasping an object.

14. *GraspRelease:* Opens the fingers on a hand.

15. *HelpWithChores:* Integrates all of the tasks into a single, assistive behavior.

16. *InterestRegions:* Detects fast moving, convex edges in the visual foreground.

17. *PalmServo:* Visually servos the contact surface of the palm.

18. *PersonDetect:* Detects the face or hand waving of a person.

19. *PersonReach:* Reaches towards a person.

20. *PersonSeek:* Scans the room in search for a person.

21. *PowerGrasp:* Forms a power grasp with the hand.

22. *PuttingStuffAway:* Assists a person by putting items on a shelf.

23. *RetractArm:* Brings an arm to the robot's side.

24. *StiffnessAdapt:* Sets the stiffness of the arm.

25. *ShelfDetect:* Detects a shelf edge using fiducials.

26. *SurfaceTest:* Reaches out to a surface to confirm its existence.

27. *SurfacePlace:* Places a grasped object on a surface.

28. *SurfaceReach:* Reaches to just above a known surface.

29. *SwitchHands:* Transfer a grasped object from one hand to another.

30. *TipEstimate:* Estimates the location of the distal tip of a grasped object.

31. *TipPose:* Controls the position and orientation of the tip of a grasped object.

32. *TipPriors:* Predicts the location of the tip of a grasped object for a given task.

33. *TipServo:* Uses visual feedback to control the tip of a grasped object.

34. *TipUse:* Grasps an object, finds its tip, and the controls the tip for a task.

35. *VisualFixate:* Servos a visual feature to the center of the image.

36. *VisualSearch:* Randomly scans the gaze around a room.

37. *VocalRequest:* Detects a person's vocal commands to perform a task.

38. *WatchHand :* Servo the eye gaze to the hand.

39. *WristWiggle:* Rotate a grasped object at the wrist while keeping the arm stationary.

Bibliography

Agre, P. and Chapman, D. (1987). Pengi: An implementation of a theory of activity. In *Proceedings of the Sixth National Conference on Artificial Intelligence (AAAI-87)*, pages 268–272.

Alami, R., Albu-Schaeffer, A., Bicchi, A., Bischoff, R., Chatila, R., Luca, A. D., Santis, A. D., Giralt, G., Guiochet, J., Hirzinger, G., Ingrand, F., Lippiello, V., Mattone, R., Powell, D., Sen, S., Siciliano, B., Tonietti, G., and Villani, L. (2006). Safe and dependable physical human-robot interaction in anthropic domains: State of the art and challenges. In *IROS'06 Workshop on Physical Human-Robot Interaction in Anthropic Domains*.

Albus, J. (1991). Outline for a theory of intelligence. *IEEE Transactions on Systems, Man and Cybernetics*, 21:473–509.

Albus, J., Lumia, R., Fiala, J., and Wavering, A. (1989). NASREM– The NASA/NBS standard reference model for telerobot control system architecture. In *Proceedings of the 20th International Symposium on Industrial Robots*.

Aldridge, H., Bluethmann, B., Ambrose, R., and Diftler, M. (2000). Control architecture for the Robonaut space humanoid. In *Proceedings of the First IEEE-RAS International Conference on Humanoid Robots*, Cambridge, Massachusetts.

Ambrose, R. O., Aldridge, H., Askew, R. S., Burridge, R., Bluethman, W., Diftler, M., Lovchik, C., Magruder, D., and Rehnmark, F. (2000). Robonaut: NASA's space humanoid. *IEEE Intelligent Systems Journal*, 15(4):57–63.

Arkin, R. (1998). *Behavior Based Robotics*. MIT Press, Cambridge, Ma.

Arkin, R. C. (1989). Homeostatic control for a mobile robot: Dynamic replanning in hazardous environments. In Wolfe, W. J., editor, *SPIE Proceedings, Mobile Robots III*, volume 1007, pages 407–413.

Arsenio, A. and Fitzpatrick, P. (2003). Exploiting cross-modal rhythm for robot perception of objects. In *Proceedings of the Second International Conference on Computational Intelligence, Robotics, and Autonomous Systems*.

Aryananda, L. (2006). Attending to learn and learning to attend for a social robot. In *Proceedings of the IEEE-RAS/RSJ International Conference on Humanoid Robots*, Genova, Italy.

Aryananda, L. and Weber, J. (2004). Mertz: A quest for a robust and scalable active vision humanoid head robot. In *Proceedings of the IEEE-RAS International Conference on Humanoid Robotics*, Santa Monica, Los Angeles, CA, USA. IEEE Press.

Biagiotti, L., Tiezzi, P., Vassura, G., and Melchiorri, C. (2005). Modelling and controlling the compliance of a robotic hand with soft finger-pads. In *Springer Tracts in Advanced Robotics*, volume 18, pages 55–75. Springer.

Bicchi, A. (2000). Hands for dexterous manipulation and robust grasping: a difficult road towards simplicity. *IEEE Transactions on Robotics and Automation*, 16(6):652–662.

Bicchi, A. and Tonietti, G. (2004). Fast and soft arm tactics: Dealing with the safety-performance trade-off in robot arms design and control. *IEEE Robotics and Automation Magazine*, 11(2):22–33.

Bicchi, A., Tonietti, G., Bavaro, M., and Piccigallo, M. (2003). Variable stiffness actuators for fast and safe motion control. In Siciliano, B., Khatib, O., and Groen, F., editors, *Proceedings of ISRR 2003*, Springer Tracts in Advanced Robotics (STAR). Springer Verlag.

Bischoff, R. and Graefe, V. (2005). Design principles for dependable robotics assistants. *International Journal of Humanoid Robotics*, 1(1):95–125.

Blackmore, L. and Block, S. (2006). Control and estimation for cooperative manipulator

tasks. Technical Report MIT-CSAIL-TR-2006-011, Massachusetts Institute of Technology, Cambridge, Ma.

Bluethmann, W., Ambrose, R., Fagg, A., Rosenstein, M., Platt, R., Grupen, R., Brezeal, C., Brooks, A., Lockerd, A., Peters, R., Jenkins, O., Mataric, M., and Bugajska, M. (2004). Building an autonomous humanoid tool user. In *Proceedings of the 2004 IEEE International Conference on Humanoid Robots*, Santa Monica, Los Angeles, CA, USA. IEEE Press.

Bogoni, L. (1998). Functional features for chopping extracted from observations and interactions. *Image and Vision Computing*, 16:765–783.

Braitenberg, V. (1984). *Vehicles: Experiments in Synthetic Psychology*. The MIT Press, Cambridge, Ma.

Breazeal, C. (2000). *Sociable Machines: Expressive Social Exchange Between Humans and Robots*. PhD thesis, MIT, Cambridge, Ma.

Breazeal, C. (2004). Social interactions in HRI: The robot view. *IEEE Transactions on Man, Cybernetics and Systems: Part C*, 34(2):181–186.

Breazeal, C., Brooks, A., Chilongo, D., Gray, J., Hoffman, G., Kidd, C., Lee, H., Lieberman, J., and Lockerd, A. (2004). Working collaboratively with humanoid robots. In *Proceedings of IEEE-RAS/RSJ International Conference on Humanoid Robots*, Santa Monica, CA.

Breazeal, C., Edsinger, A., Fitzpatrick, P., and Scassellati, B. (2001). Active vision for sociable robots. *IEEE Transactions on Systems, Man, and Cybernetics*, 31(5):443–453.

Brock, O. and Grupen, R., editors (2005). *Proceedings of the NASA/NSF Workshop on Autonomous Mobile Manipulation (AMM)*, Houston, Texas.

Brooks, R. A. (1986). A robust layered control system for a mobile robot. RA-2:14–23.

Brooks, R. A. (1990). Challenges for complete creature architectures. In *Proceedings of Simulation of Adaptive Behavior (SAB90)*.

Brooks, R. A. (1991a). Intelligence without reason. In *Proceedings of the 1991 International Joint Conference on Artificial Intelligence*, pages 569–595.

Brooks, R. A. (1991b). Intelligence without representation. *Artificial Intelligence Journal*, 47:139–160. originally appeared as MIT AI Memo 899 in May 1986.

Brooks, R. A. (1999). *Cambrian Intelligence*. MIT Press, Cambridge, MA.

Brooks, R. A., Aryananda, L., Edsinger, A., Fitzpatrick, P., Kemp, C., O'Reilly, U.-M., Torres-Jara, E., Varshavskaya, P., and Weber, J. (2004). Sensing and manipulating built-for-human environments. *International Journal of Humanoid Robotics*, 1(1).

Brooks, R. A., Breazeal, C., Marjanovic, M., Scassellati, B., and Williamson, M. (1999). The Cog project: Building a humanoid robot. In Nehaniv, C. L., editor, *Computation for Metaphors, Analogy and Agents*, volume 1562 of *Springer Lecture Notes in Artificial Intelligence*. Springer-Verlag.

Brooks, R. A. and Rosenberg, C. (1995). L - a common lisp for embedded systems. In *Lisp Users and Vendors Conference*.

Burke, J., Murphy, R., Rogers, E., Lumelsky, V., and Scholtz, J. (2004). Final report for the DARPA/NSF interdisciplinary study on human-robot interaction. *IEEE Transactions on Systems, Man and Cybernetics, Special Issue on Human-Robot Interaction*, 34(2):103–112.

Buss, S. (2004). Introduction to inverse kinematics with jacobian transpose, pseudoinverse, and damped least squares methods. Unpublished, available at http://math.ucsd.edu/ sbuss/ResearchWeb/ikmethods/iksurvey.pdf.

Buss, S. and Kim, J.-S. (2005). Selectively damped least squares for inverse kinematics. *Journal of Graphics Tools*, 10(3):37–49.

Calinon, S. and Billard, A. (2006). Teaching a humanoid robot to recognize and reproduce social cues. In *Proceedings of the 19th International Symposium on Robot and Human Interactive Communication (RO-MAN06)*.

Canny, J. (1986). A computational approach to edge detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 8:679–698.

Cham, J. G., Bailey, S. A., Clark, J. E., Full, R. J., and Cutkosky, M. R. (2002). Fast and robust: Hexapedal robots via shape deposition manufacturing. *The International Journal of Robotics Research*, 21(10):869–883.

Chang, C.-C. and Lin, C.-J. (2001). LIBSVM: a library for support vector machines.

Cheng, G., Nagakubo, A., and Kuniyoshi, Y. (2001). Continuous humanoid interaction: An integrated perspective- Gaining adaptivity, redundancy, flexibility- In one. *Robotics and Autonomous Systems*, 37(2-3):161–183.

Cisek, P. (2005). Neural representations of motor plans, desired trajectories, and controlled objects. *Cognitive Processing*, 6:15–24.

Coelho, J., Piater, J., and Grupen, R. (2000). Developing haptic and visual perceptual categories for reaching and grasping with a humanoid robot. In *Proceedings of the First IEEE-RAS International Conference on Humanoid Robots*, Cambridge, MA, USA.

Connell, J. (1989). A behavior-based arm controller. *IEEE Transactions on Robotics and Automation*, 5(5):784–791.

Craig, J. (1989). *Introduction to Robotics*. Addison Wesley, 2 edition.

Cutkosky, M. and Kao, I. (1989). Computing and controlling the compliance of a robotic hand. *IEEE Transactions on Robotics and Automation*, 5(2):151–165.

Diftler, M., Ambrose, R., Tyree, K., Goza, S., and Huber, E. (2004). A mobile autonomous humanoid assistant. In *Proceedings of the 4th IEEE/RAS International Conference on Humanoid Robots*.

Dollar, A. M. and Howe, R. D. (2005). Towards grasping in unstructured environments: Grasper compliance and configuration optimization. *Advanced Robotics*, 19(5):523–543.

Duda, R. O., Hart, P. E., and Stork, D. G. (2000). *Pattern Classification (2nd Edition)*. Wiley-Interscience.

Duric, Z., Fayman, J. A., and Rivlin, E. (1996). Function from motion. *IEEE Trans. Pattern Anal. Mach. Intell.*, 18(6):579–591.

Edsinger, A. and Kemp, C. (2007). Toward robot learning of tool manipulation from human demonstration. Technical Report To Appear, MIT Computer Science and Artificial Intelligence Laboratory.

Endo, Y. and Arkin, R. C. (2003). Anticipatory robot navigation by simultaneously localizing and building a cognitive map. In *Proceedings of the 2003 IEEE/RSJ International Conference on Intelligent Robots and System (IROS)*, pages 460–466, Las Vegas, NV.

Erdmann, M. and Mason, M. (1988). An exploration of sensorless manipulation. *IEEE Journal of Robotics and Automation*, 4(4):369–379.

ExactDynamics (2006). *The Manus Assistive Robot Manipulator (ARM)*. Available at http://www.exactdynamics.nl.

Fitzgibbon, A. W., Cross, G., and Zisserman, A. (1998). Automatic 3D model construction for turn-table sequences. In Koch, R. and VanGool, L., editors, *Proceedings of SMILE Workshop on Structure from Multiple Images in Large Scale Environments*, volume 1506 of *Lecture Notes in Computer Science*, pages 154–170. Springer Verlag.

Fitzpatrick, P. (2003). *From First Contact to Close Encounters: A developmentally deep perceptual system for a humanoid robot*. PhD thesis, Massachusetts Institute of Technology.

Fitzpatrick, P. and Metta, G. (2003). Grounding vision through experimental manipulation. *Philosophical Transactions of the Royal Society: Mathematical, Physical, and Engineering Sciences*, 361:2165–2185.

Fitzpatrick, P., Metta, G., Natale, L., Rao, S., and Sandini, G. (2003). Learning about objects through action - initial steps towards artificial cognition. In *Proceedings of the 2003 IEEE International Conference on Robotics and Automation (ICRA-03)*, volume 3. IEEE Press.

Fitzpatrick, P. and Torres-Jara, E. (2004). The power of the dark side: using cast shadows for visually-guided reaching. In *Proceedings of the International Conference on Humanoid Robotics*.

Fong, T., Nourbakhsh, I., and Dautenhahn, K. (2002). A survey of socially interactive robots. *Robotics and Autonomous Systems*, 42:143–166.

Forsyth, D. A. and Ponce, J. (2002). *Computer Vision: A modern approach*. Prentice Hall Professional Technical Reference.

Gaskett, C. and Cheng, G. (2003). Online learning of a motor map for humanoid robot reaching. In *Proceedings of the 2nd International Conference on Computational Intelligence, Robotics and Autonomous Systems (CIRAS 2003)*, Singapore.

Gentilucci, M., Roy, A., and Stefanini, S. (2004). Grasping an object naturally or with a tool: Are these tasks guided by a common motor representation? *Experimental Brain Research*, 157(4):496–506.

Gibson, J. J. (1977). The theory of affordances. In Shaw, R. and Bransford, J., editors, *Perceiving, Acting, and Knowing*. Lawrence Erlbaum Associates, Hillsdale, NJ.

Gibson, J. J. (1979). *The ecological approach to visual perception*. Houghton Mifflin, Boston, Ma.

Glassmire, J., O'Malley, M., Bluethmann, W., and Ambrose, R. (2004). Cooperative manipulation between humans and teleoperated agents. In *Proceedings of the 12th International Symposium on Haptic Interfaces for Virtual Environments and Teleoperator Systems*.

Gourdeau, R. (2005). *ROBOOP - A robotics object oriented package in C+*. Available at http://www.cours.polymtl.ca/roboop/.

Grupen, R. A., Henderson, T. C., and McCammon., I. D. (1989). A survey of general-purpose manipulation. *International Journal of Robotics Research*, 8(1):38–61.

Guertin, J. A. and Townsend, W. T. (1999). Teleoperator slave - WAM design methodology. *Industrial Robot*, 26:167–177.

Haigh, K. and Yanco, H. A. (2002). Automation as caregiver: A survey of issues and technologies. In *Proceedings of the AAAI-2002 Workshop on Automation as Caregiver: The Role of Intelligent Technology in Elder Care*, Edmonton, Alberta.

Hart, S., Ou, S., Sweeney, J., , and Grupen, R. (2006). A framework for learning declarative structure. In *Proceedings of the Robotics, Science & Systems Workshop on Manipulation for Human Environments*, Philedelphia, Pennsylvania.

Hartley, R. I. and Zisserman, A. (2004). *Multiple View Geometry in Computer Vision*. Cambridge University Press, 2nd edition.

Hillman, M., Hagan, K., Hagan, S., Jepson, J., and Orpwood, R. (1999). A wheelchair mounted assistive robot. In *Proceedings of ICORR '99: International Conference on Rehabilitation Robotics*.

Hirzinger, G., Sporer, N., Albu-Schaffer, A., Hahnle, M., and Pascucci, A. (2002). DLR's torque-controlled light weight robot III? Are we reaching the technological limits now? In *Proceedings of the International Conference on Robotics Automation*, pages 1710–1716.

Hollerbach, J. and Wampler, C. (1996). The calibration index and taxonomy for robot kinematic calibration methods. *International Journal of Robotics Research*, 15(6):573–591.

Horswill, I. (1993). Polly: A vision-based artificial agent. In *Proceedings of the 11th National Conference on Artificial Intelligence*, pages 824–829, Menlo Park, CA, USA. AAAI Press.

Huber, E. and Baker, K. (2004). Using a hybrid of silhouette and range templates for real-time pose estimation. In *Proceedings of ICRA 2004 IEEE International Conference on Robotics and Automation*, volume 2, pages 1652–1657.

Huber, M. and Grupen, R. (1994). 2-d contact detection and localization using proprioceptive information. *IEEE Transactions on Robotics and Automation*, 10(1):1–11.

Huber, M. and Grupen, R. (2002). Robust finger gaits from closed-loop controllers. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1578–1584, Lausanne, Switzerland.

Iida, F. and Pfeifer, R. (2006). Sensing through body dynamics. *Robotics and Autonomous Systems*, 54(8):631–640.

Inoue, H. (1979). Force feedback in precise assembly tasks. In Winston, P. and Brown, R., editors, *Artificial Intelligence: An MIT Perspective*. The MIT Press.

Itti, L., Koch, C., and Niebur, E. (1998). A model of saliency-based visual attention for rapid scene analysis. *IEEE Trans. Pattern Anal. Mach. Intell.*, 20(11):1254–1259.

J. Pratt, M. Chew, P. D. and Pratt, G. (2001). Virtual Model Control: An intuitive approach for bipedal locomotion. *International Journal of Robotics Research*, 20(2):129–143.

Jagersand, M. and Nelson, R. (1995). Visual space task specification, planning and control. In *Proceedings of the IEEE International Symposium on Computer Vision*, pages 521–526.

Jenkins, O., Peter, R., and Bodenheimer, R. (2006). Uncovering success in manipulation. In *Proceedings of the Robotics: Science and Systems Workshop on Manipulation in Human Environments.*

Jones, E., Oliphant, T., Peterson, P., et al. (2001). SciPy: Open source scientific tools for Python. Available at http://www.scipy.org/.

Kandel, E. R., Schwartz, J. H., and Jessell, T. (2000). *Principles of Neural Science.* McGraw-Hill and Appleton and Lange, New York, NY, 4th edition.

Kawasaki, H., Komatsu, T., and Uchiyama, K. (2002). Dexterous anthropomorphic robot hand with distributed tactile sensor: Gifu Hand II. *IEEE Transactions on Mechatronics*, 7(3):296–303.

Kemp, C. C. (2005). *A Wearable System that Learns a Kinematic Model and Finds Structure in Everyday Manipulation by using Absolute Orientation Sensors and a Camera.* PhD thesis, Massachusetts Institute of Technology.

Kemp, C. C. (2006). *pysense: A python based robotics package.* Available at http://robotmanipulation.org/pysense/.

Kemp, C. C. and Edsinger, A. (2005). Visual tool tip detection and position estimation for robotic manipulation of unknown human tools. Technical Report AIM-2005-037, MIT Computer Science and Artificial Intelligence Laboratory.

Kemp, C. C. and Edsinger, A. (2006a). Robot manipulation of human tools: Autonomous detection and control of task relevant features. In *Proceedings of the 5th IEEE International Conference on Development and Learning (ICDL-06)*, Bloomington, Indiana.

Kemp, C. C. and Edsinger, A. (2006b). What can I control?: The development of visual categories for a robot's body and the world that it influences. In *Proceedings of the Fifth International Conference on Development and Learning, Special Session on Autonomous Mental Development.*

Khatib, O. (1987). A unified approach to motion and force control of robot manipulators: The operational space formulation. *International Journal of Robotics and Automation*, 3(1):43–53.

Khatib, O., Yokoi, K., Brock, O., Chang, K., and Casal, A. (1999). Robots in human environments: Basic autonomous capabilities. *International Journal of Robotics Research*, 18(684).

Kikuchi, H., Yokoyama, M., Hoashi, K., Hidaki, Y., Kobayashi, T., and Shirai, K. (1998). Controlling gaze of humanoid in communication with human. In *Proceedings of the 1998 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Victoria, B.C., Canada.

Kim, B., Yun, S., Kang, S., Hwang, C., Kim, M., and Song, J. (2005). Development of a joint torque sensor fully integrated with an actuator. In *Proceedings of the 2005 International Conference on Control, Automation, and Systems*, pages 1679–1683, Gyeonggi-Do, Korea.

Kozima, H. and Yano, H. (2001). A robot that learns to communicate with human caregivers. In *Proceedings of the International Workshop on Epigentic Robotics*.

Kragic, D. and Chrisensen, H. I. (2002). Survey on visual servoing for manipulation. Technical report, Computational Vision and Active Perception Laboratory.

Kragic, D. and Christensen, H. (2001). Cue integration for visual servoing. *IEEE Transactions on Robotics and Automation*, 17(1):18–28.

Kyong, K., Freedman, S., Mataric, M., Cunningham, M., and Lopez, B. (2005). A hands-off physical therapy assistance robot for cardiac patients. In *9th International Conference on Rehabilitation Robotics*, pages 337– 340.

Lawrence, D. A. (1989). Actuator limitations on achievable manipulator impedance. In *Proceedings of the 1989 IEEE International Conference on Robotics and Automation*.

Littman, E., Drees, A., and Ritter, H. (1996). Robot guidance by human pointing gestures. In *Proceedings of th International Workshop on Neural Networks for Identification, Control, Robotics, and Signal/Image Processing*.

Lovchik, C. and Diftler, M. (1999). The Robonaut hand: A dexterous robot hand for space. In *Proceedings of the IEEE International Conference on Automation and Robotics*, volume 2, pages 907–912, Detroit, Michigan.

Lozano-Perez, T., Mason, M., and Taylor, R. H. (1984). Automatic synthesis of fine-motion strategies for robots. *International Journal of Robotics Research*, 3(1).

Ly, D., Regenstein, K., Asfour, T., and Dillmann, R. (2004). A modular and distributed embedded control architecture for humanoid robots. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, Japan.

Maeno, I. Y. T. (2005). Five-fingered robot hand using ultrasonic motors elastic elements. In *Proceedings of the 2005 IEEE International Conference on Robotics and Automation*, Barcelona, Spain.

Mason, M. (1985). The mechanics of manipulation. In *Proceedings of the IEEE International Conference on Robotics and Automation*.

Mason, M. (2001). *Mechanics of Robotic Manipulation*. MIT Press, Cambridge, Ma. Intelligent Robotics and Autonomous Agents Series.

Mataric, M. (1992). Behavior-based control: Main properties and implications. In *Proceedings of Workshop on Intelligent Control Systems, International Conference on Robotics and Automation*, Nice, France.

Metta, G. (2000). *Babybot: a study into sensorimotor development*. PhD thesis, LIRA-Lab, DIST, University of Genoa.

Metta, G. and Fitzpatrick, P. (2003a). Better vision through manipulation. *Adaptive Behavior*, 11:109–128.

Metta, G. and Fitzpatrick, P. (2003b). Early integration of vision and manipulation. *Adaptive Behavior*, 11(2):109–128.

Metta, G., Fitzpatrick, P., and Natale, L. (2006). YARP: yet another robot platform. *International Journal on Advanced Robotics Systems. Special Issue on Software Development and Integration in Robotics*, 3(1):43–48.

Michel, Gold, and Scassellati (2004). Motion-based robotic self-recognition. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, Sendai, Japan.

Mon-Williams, M. and Tresilian, J. (2001). A simple rule of thumb for elegant prehension. *Current Biology*, 11(13):1058–1061.

Nagai, Y. (2005). Learning to comprehend deictic gestures in robots and human infants. In *Proceedings of the IEEE International Workshop on Robot and Human Interactive Communication, ROMAN 2005*, pages 217–212.

Natale, L. (2004). *Linking Action to Perception in a Humanoid Robot: A Developmental Approach to Grasping*. PhD thesis, LIRA-Lab, DIST, University of Genoa.

Natale, L., Metta, G., and Sandini, G. (2004). Learning haptic representations of objects. In *Proceedings of the International Conference on Intelligent Manipulation and Grasping*, Genoa, Italy.

Norman, D. A. (1990). *The Design of Everyday Things*. Doubleday, New York, New York.

Ogawa, N. (2003). Rapid population aging in Japan: Simulated results based on the nupri economic-demographic-social security model. In *Proceedings of the 21st Population Census Conference*, Kyoto, Japan.

Pan, P., Lynch, K., and Peshkin, M. A. (2005). Human interaction with passive assistive robots. In *IEEE 9th International Conference on Rehabilitation Robotics*.

Peters, R. A., Hambuchen, K. E., Kawamura, K., and Wilkes, D. M. (2001). The Sensory Ego-Sphere as a short-term memory for humanoids. In *IEEE-RAS International Conference on Humanoid Robots*, pages 451–459, Tokyo, Japan.

Pfeifer, R. and Iida, F. (2005). Morphological computation: Connecting body, brain and environment. *Japanese Scientific Monthly*, 58(2):48–54.

Piater, J. and Grupen, R. (2002). Learning appearance features to support robotic manipulation. In *Proceedings of the Cognitive Vision Workshop*, Zurich, Switzerland.

Pineau, J., Montemerlo, M., Pollack, M., Roy, N., and Thrun, S. (2003). Towards robotic assistants in nursing homes: Challenges and results. *Robotics and Autonomous Systems*, 42(3-4):271–281.

Platt, R. (2006). *Learning and Generalizing Control-Based Grasping and Manipulation Skills*. PhD thesis, University Massachusetts.

Platt, R., Brock, O., Fagg, A., Karuppiah, D., Rosenstein, M., Coelho, J., Huber, M., Piater, J., Wheeler, D., and Grupen, R. (2003a). A framework for humanoid control and intelligence. In *Proceedings of the 2003 IEEE International Conference on Humanoid Robots*.

Platt, R., Fagg, A., and Grupen, R. (2003b). Extending fingertip grasping to whole body grasping. In *Proceedings of the 2003 IEEE International Conference on Robotics and Automation*, Taipei, Taiwan.

Platt, R., Fagg, A., and Grupen, R. (2003c). Nullspace composition of control laws for grasping. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*.

Platt, R., Fagg, A., and Grupen, R. (2004). Manipulation gaits: Sequences of grasp control tasks. In *Proceedings of the 2004 IEEE Conference on Robotics and Automation (ICRA)*.

Pollack, M., Engberg, S., Matthews, J., Thrun, S., Brown, L., Colbry, D., Orosz, C., Peintner, B., Ramakrishnan, S., Dunbar-Jacob, J., McCarthy, C., Montemerlo, M., Pineau, J., and Roy, N. (2002). Pearl: A mobile robotic assistant for the elderly. In *Workshop on Automation as Caregiver: the Role of Intelligent Technology in Elder Care (AAAI)*.

Pollard, N. and Hodgins, J. (2002). Generalizing Demonstrated Manipulation Tasks. In *Proceedings of the Workshop on the Algorithmic Foundations of Robotics (WAFR '02)*.

Pratt, G. and Williamson, M. (1995). Series Elastic Actuators. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS-95)*, volume 1, pages 399–406, Pittsburg, PA.

Radwin, R. and Haney, J. (1996). An ergonomics guide to hand tools. Technical report, American Institutional Hygiene Association. Available at http://ergo.engr.wisc.edu/pubs.htm.

Raibert, M. H. (1986). *Legged robots that balance*. Massachusetts Institute of Technology, Cambridge, MA, USA.

Robinson, D. (2000). *Design and Analysis of Series Elasticity in Closed-loop Actuator Force Control*. PhD thesis, Massachusetts Institute of Technology, Cambridge, MA.

Rome, E., Hertzberg, J., Dorffner, G., and Doherty, P. (2006). Executive summary – towards affordance-based robot control. In Rome, E., Doherty, P., Dorffner, G., and Hertzberg, J., editors, *Towards Affordance-Based Robot Control*, number 06231 in Dagstuhl Seminar Proceedings.

Rosenstein, M., Fagg, A., Ou, S., and Grupen, R. (2005). User intentions funneled through a human-robot interface. In *Proceedings of the 10th International Conference on Intelligent User Interfaces*, pages 257–259.

Rosier, J., van Woerden, J., van der Kolk, L., Driessen, B., Kwee, H., Duimel, J., Smits, J., de Moed, A. T., Honderd, G., and Bruyn, P. (1991). Rehabilitation robotics: The Manus concept. In *Proceedings of the Fifth International Conference on Advanced Robotics: Robots in Unstructured Environments*, volume 1.

Rucci, M. and Desbordes, G. (2003). Contributions of fixational eye movements to the discrimination of briefly presented stimuli. *Journal of Vision*, 3(11):852–864.

Saha, M. and Isto, P. (2006). Motion planning for robotic manipulation of deformable linear objects. In *Proc. IEEE Int. Conf. on Robotics and Automation (ICRA)*.

Salisbury, J. K. (1980). Active stiffness control of a manipulator in cartesian coordinates. In *Proceedings of the IEEE Conference on Decision and Control*.

Salisbury, J. K., Townsend, W. T., Eberman, B. S., and DiPietro, D. M. (1988). Preliminary design of a WholeArm Manipulation System (WAMS). In *Proc 1988 IEEE Intl Conf on Robotics and Automation*.

Santini, F. and Rucci, M. (2006). Active estimation of distance in a robotic system that replicates human eye movements. *Journal of Robotics and Autonomous Systems*, In Press.

Saxena, A., Driemeyer, J., Kearns, J., Osondu, C., and Ng, A. Y. (2006). Learning to grasp novel objects using vision. In *Proceedings of the International Symposium on Experimental Robotics (ISER)*.

Shimoga, K. (1996). Robot grasp synthesis algorithms: a survey. *International Journal of Robotics Research*, 15(3):230–266.

Sian, N., Yoki, K., Kawai, Y., and Muruyama, K. (2006). Operating humanoid robots in human environments. In *Proceedings of the Robotics, Science & Systems Workshop on Manipulation for Human Environments*, Philedelphia, Pennsylvania.

Simeon, T., Laumond, J. P., Cortes, J., and Sahbani, A. (2004). Manipulation planning with probabilistic roadmaps. *IJRR*, 23(7-8):729–746.

Song, W., Lee, H., Kim, J., Yoon, Y., and Bien, Z. (1998). KARES: Intelligent robotics system for the disabled and the elderly. In *Proceedings of the 20th Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, volume 20.

St. Amant, R. and Wood, A. (2005). Tool use for autonomous agents. In *Proceedings of the National Conference on Artificial Intelligence (AAAI)*, pages 184–189.

Stanger, C., Anglin, C., Harwin, W., and Romilly, D. (1994). Devices for assisting manipulation: A summary of user task priorities. *IEEE Transactions on Rehabilitation Engineering*, 2(4):256–265.

Stiefelhagen, R., Fugen, C., Gieselmann, P., Holzapfel, H., Nickel, K., and Waibel, A. (2004). Natural human-robot interaction using speech, head pose and gestures. In *Proceedings of the 2005 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Sendai, Japan.

Stiller, C. and Konrad, J. (1999). Estimating motion in image sequences, a tutorial on modeling and computation of 2d motion. *IEEE Signal Process. Mag., vol. 16, pp. 70–91.*

Stoytchev, A. (2005). Behavior-grounded representation of tool affordances. In *Proceedings of the IEEE International Conference on Robotics and Automation.*

Tapus, A. and Mataric, M. J. (2006). Towards socially assistive robotics. *International Journal of the Robotics Society of Japan*, 24(5).

Taylor, G. (2004). *Robust Perception and Control for Humanoid Robots in Unstructured Environments Using Vision.* PhD thesis, Monash University, Clayton, Victoria 3800, Australia.

Tedrake, R. L. (2004). *Applied Optimal Control for Dynamically Stable Legged Locomotion*. PhD thesis, Massachusetts Institute of Technology.

Tegin, J. and Wikander, J. (2005). Tactile sensing in intelligent robotic manipulation - a review. *Industrial Robot*, 34(1):64–70.

Thrun, S., Burgard, W., and Fox, D. (2005). *Probabilistic Robotics*. MIT Press.

Todorov, E. and Jordan, M. (2002). Optimal feedback control as as theory of motor coordination. *Nature Neuroscience*, 5(11):1226–1235.

Tolani, D., Goswami, A., and Badler, N. I. (2000). Real-time inverse kinematics techniques for anthropomorphic limbs. *Graphical models*, 62(5):353–388.

Torralba, A. (2003). Modeling global scene factors in attention. *Journal of Optical Society of America A: Special Issue on Bayesian and Statistical Approaches to Vision*, 20(7):1407–1418.

Torres-Jara, E. (2005). Obrero: a platform for sensitive manipulation. In *Proceedings of the 5th IEEE-RAS International Conference on Humanoid Robots*, pages 327 – 332.

Torres-Jara, E., Natale, L., and Fitzpatrick, P. (2005). Tapping into touch. In Berthouze, L., Kaplan, F., Kozima, H., Yano, H., Konczak, J., Metta, G., Nadel, J., Sandini, G., and Stojanov, G., editors, *Proceedings of the Fifth International Workshop on Epigenetic Robotics*, pages 79–86, Nara, Japan.

Townsend, W. T. and Salisbury, J. K. (1993). Mechanical design for WholeArm Manipulation. In *Robots and biological systems : towards a new bionics?* Springer-Verlag.

Ulrich, K. T., Tuttle, T. D., Donoghue, J. P., and Townsend, W. T. (1995). Intrinsically safer robots. Technical report, Barrett Technologies and NASA Kennedy Space Center, Cambridge, MA. USA.

Versace, J. (1971). A review of the severity index. In *Proceedings of the 15th Stapp Car Crash Conference*, pages 771–796.

Viola, P. and Jones, M. (2004). Robust real-time object detection. 57(2):137–154.

Vygotsky, L. (1962). *Thinking and Speaking*. The M.I.T. Press.

Waarsing, B., Nuttin, M., and Brussel, H. V. (2001). Introducing robots into a human-centred environment - the behaviour-based approach. In *Proceedings of the 4th International Conference on CLAWAR*, pages 465–470.

Wasik, Z. and Saffiotti, A. (2003). A hierarchical behavior-based approach to manipulation tasks. In *Proceedings of the 2003 IEEE International Conference on Robotics and Automation (ICRA-03)*. IEEE Press.

Whitney, D. and Nevins, J. (1979). What is the remote centre compliance (RCC) and what can it do? In *Proceedings of the 9th International Symposium on Robots*, pages 135–152, Washington, D.C.

Williams, D. and Khatib, O. (1993). The virtual linkage: A model for internal forces in multi-grasp manipulation. In *Proceedings of the International Conference on Robotics and Automation*, volume 1, pages 1025–30.

Williamson, M. M. (1998a). Exploiting natural dynamics in robot control. In *Fourteenth European Meeting on Cybernetics and Systems Research (EMCSR '98)*, Vienna, Austria.

Williamson, M. M. (1998b). Rhythmic robot control using oscillators. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 77–83, Victoria, Canada.

Williamson, M. M. (1999). *Robot Arm Control Exploiting Natural Dynamics*. PhD thesis, Massachusetts Institute of Technology.

Winston, P. H., Katz, B., Binford, T. O., and Lowry, M. R. (1983). Learning physical descriptions from functional definitions, examples, and precedents. In *National Conference on Artificial Intelligence*, pages 433–439.

Wolfe, J. (1994). Guided search 2.0: a revised model of visual search. *Psychonomic Bulletin and Review*, 1(3):202–238.

Worsnopp, T., Peshkin, M. A., Lynch, K., and Colgate, J. E. (2006). Controlling the apparent inertia of passive human-interactive robots. *Journal of Dynamic Systems Measurement and Control*, 128(1).

Yigit, S., Burghart, C., and Worn, H. (2003a). Specific combined control mechanisms for human robot cooperation. In *Proceedings of Intelligent Systems and Control*, San Diego, California.

Yigit, S., Burghart, C., and Wörn, H. (2003b). Concept of combined control mechanisms for human-robot-co-operation. In *Proceedings of the International Conference on Computer, Communication and Control Technologies (CCCT)*, Orlando, Florida.

Yokohama, K., Handa, H., Isozumi, T., Fukase, Y., Kaneko, K., Kanehiro, F., Kawai, Y., Tomita, F., and Hirukawa, H. (2003). Cooperative works by a human and a humanoid robot. In *Proceedings of the 2003 IEEE International Conference on Robotics and Automation*, Taipei, Taiwan.

Zinn, M., Khatib, O., Roth, B., and Salisbury, J. (2002). A new actuation approach for human friendly robot design. In *International Symposium on Experimental Robotics*, Italy.

Zinn, M., Khatib, O., Roth, B., and Salisbury, J. (2004). Playing it safe: human-friendly robots. *IEEE Robotics & Automation Magazine*, 11(2):12–21.

Zöllner, R., Asfour, T., and Dillmann, R. (2004). Programming by demonstration: Dual-arm manipulation tasks for humanoid robots. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2004)*, Sendai, Japan.