INTERACTIVE PERCEPTION OF ARTICULATED OBJECTS FOR AUTONOMOUS MANIPULATION

A Dissertation Presented

by

DOV KATZ

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

DOCTOR OF PHILOSOPHY

September 2011

Computer Science

INTERACTIVE PERCEPTION OF ARTICULATED OBJECTS FOR AUTONOMOUS MANIPULATION

A Dissertation Presented

by

DOV KATZ

Approved as to style and content by:

Oliver Brock, Chair

Erik Learned Miller, Member

Andrew G. Barto, Member

Andrew Cohen, Member

Andrew G. Barto, Department Chair
Computer Science

"For the things we have to learn before we can do them, we learn by doing them."

— *Aristotle*

ACKNOWLEDGMENTS

Graduate school is a long and challenging process. In this journey, I have lived in two foreign countries, worked at two universities, and met dozens of friends and colleagues. I could never have done it alone. More people than I could possibly name have helped me along the way. But there are a few who deserve special recognition for their impact on my life.

First and foremost, my wonderful, beautiful and patient wife, Gili. For all those times you stood by me, for all the joy you brought to my life, and for every dream you made come true. Without your love, none of this would have been worthwhile.

Yahel and Maayan, you amaze me every second of every day. This journey may have been a little more challenging with the two of you around, but so much more fun. I love you.

My parents, Zafie and Jacob Katz, for their love and support. You laid the foundation for everything that I have accomplished in my life. Omer, who despite being second is always somewhat ahead. And Miri, for all your curiosity and care.

After family, my adviser Oliver Brock, who taught me a great deal about many things. Oliver's knowledge, wisdom, and creativity have been inspiring. Oliver, thank you for being my strongest critique, my biggest supporter, and a friend. Everything I do in the future will reflect the things I have learned from you.

To the members of my committee: I would like to express my gratitude to Professor Erik Learned Miller for many stimulating discussions over the years. His care for detail and passion towards research have been contagious. I am thankful to Professor Andrew G. Barto whose ground-breaking research is the reason I ended up at UMass

Amherst. And finally, I would like to express my gratitude to Professor Andrew Cohen for strengthening my curiosity towards human psychology—the study of the most amazing robot.

Many others have also contributed to my academic development. All of the professors who taught my classes. Special thanks go to Leeanne Leclerc for being the spirit behind the graduate program, and a friend. Janika Urig for being such a good listener. I would also like to thank my fellow graduate students, Jacqueline Feild, TJ Brunette, Yuandong Yang, Steve Hart, Brendan Burns, George Fagogenis, Ines Putz, Clemens Eppner, Sebastian Höfer, Florian Kamm, Ingo Kossyk, Nasir Mahmood, Roberto Martn, and Michael Schneider. And last but not least, Emily Horrell, whose contributions to this thesis are too many to list.

ABSTRACT

INTERACTIVE PERCEPTION OF ARTICULATED OBJECTS FOR AUTONOMOUS MANIPULATION

SEPTEMBER 2011

DOV KATZ

B.Sc., TEL AVIV UNIVERSITY, ISRAEL

M.Sc., UNIVERSITY OF MASSACHUSETTS AMHERST

Ph.D., UNIVERSITY OF MASSACHUSETTS AMHERST

Directed by: Professor Oliver Brock

This thesis develops robotic skills for manipulating novel articulated objects. The degrees of freedom of an articulated object describe the relationship among its rigid bodies, and are often relevant to the object's intended function. Examples of everyday articulated objects include scissors, pliers, doors, door handles, books, and drawers. Autonomous manipulation of articulated objects is therefore a prerequisite for many robotic applications in our everyday environments.

Already today, robots perform complex manipulation tasks, with impressive accuracy and speed, in controlled environments such as factory floors. An important characteristic of these environments is that they can be engineered to reduce or even eliminate perception. In contrast, in unstructured environments such as our homes and offices, perception is typically much more challenging. Indeed, manipulation in these unstructured environments remains largely unsolved. We therefore assume that

to enable autonomous manipulation of objects in our everyday environments, robots must be able to acquire information about these objects, making as few assumption about the environment as possible.

Acquiring information about the world from sensor data is a challenging problem. Because there is so much information that could be measured about the environment, considering all of it is impractical given current computational speeds. Instead, we propose to leverage our understanding of the task, in order to determine the relevant information. In our case, this information consists of the object's shape and kinematic structure. Perceiving this task-specific information is still challenging. This is because in order to understand the object's degrees of freedom, we must observe relative motion between its rigid bodies. And, as relative motion is not guaranteed to occur, this information may not be included in the sensor stream.

The main contribution of this thesis is the design and implementation of a robotic system capable of perceiving and manipulating articulated objects. This system relies on *Interactive Perception*, an approach which exploits the synergies that arise when crossing the boundary between action and perception. In interactive perception, the emphasis of perception shifts from object appearance to object function. To enable the perception and manipulation of articulated objects, this thesis develops algorithms for perceiving the kinematic structure and shape of objects. The resulting perceptual capabilities are used within a relational reinforcement learning framework, enabling a robot to obtain general domain knowledge for manipulation. This composition enables our robot to reliably and efficiently manipulate novel articulated objects. To verify the effectiveness of the proposed robotic system, simulated and real-world experiments were conducted with a variety of everyday objects.

TABLE OF CONTENTS

Page

ACKNOWLEDGMENTS . . . . . **v**

ABSTRACT . . . . . **vii**

LIST OF TABLES . . . . . **xiii**

LIST OF FIGURES . . . . . **xiv**

CHAPTER

1. INTRODUCTION **1**

1.1 Main Contributions 7
1.2 Thesis Outline 8

2. THE HUMAN ANGLE **9**

2.1 Explaining by Doing 10
2.2 Explaining by Forming Concepts 11
2.3 Explaining by Leveraging Social Cues 13
2.4 The Art of Forming Explanations 14
2.5 Summary 16

3. DOING THE RIGHT THING **18**

3.1 Exposing and Exploiting Task-Relevant Structure 22

4. RELATED WORK AND BACKGROUND **24**

4.1 Related Work 24

4.1.1 The Computational Approach 26
4.1.2 The Technological Approach 28
4.1.3 Leveraging Task Structure 32

4.2 Background . . . 35

4.2.1 Interactive Perception . . . 36
4.2.2 Structure from Motion . . . 38

4.2.2.1 Key-frame Methods . . . 40
4.2.2.2 Bayesian Filtering Methods . . . 41

4.2.3 Relational Reinforcement Learning . . . 43

4.2.3.1 Instance based Representation . . . 43
4.2.3.2 Extending the Representation . . . 45

5. THE PROPOSED APPROACH . . . 47

6. MODELING PLANAR ARTICULATED OBJECTS . . . 51

6.1 Interactive Perception . . . 52
6.2 Rigid Body Modeling . . . 53

6.2.1 Collecting Perceptual Evidence for Segmentation . . . 54
6.2.2 Obtaining Rigid Body Hypotheses . . . 55

6.3 Kinematic Structure Modeling . . . 58
6.4 Experimental Results . . . 62
6.5 Summary . . . 67

7. IDENTIFYING THREE-DIMENSIONAL RIGID BODIES . . . 69

7.1 Algorithm Overview . . . 70
7.2 Step I: Collecting Perceptual Evidence . . . 71
7.3 Step II: Obtaining Rigid Body Hypotheses . . . 72

7.3.1 Relative Motion Predictor . . . 73
7.3.2 Short Distance Predictor . . . 75
7.3.3 Long Distance Predictor . . . 76
7.3.4 Color Segmentation Predictor . . . 77
7.3.5 Triangulation Predictor . . . 80
7.3.6 Fundamental Matrix Predictor . . . 82

7.3.6.1 Random Sample Consensus . . . 82
7.3.6.2 The J-Linkage Algorithm . . . 85
7.3.6.3 Modeling Motion Using The Fundamental Matrix . . . 87
7.3.6.4 Assigning Features to Rigid Bodies . . . 89

7.4 Step III: Identifying Rigid Bodies . . . 90

8. MODELING THREE-DIMENSIONAL SHAPE AND MOTION 95

9. MODELING THREE-DIMENSIONAL KINEMATIC STRUCTURES 101

9.1 Computing Relative motion 102

9.1.1 Point-Cloud to Point-Cloud Registration 103
9.1.2 Removing Global Motion 106

9.2 Analyzing Kinematic Constraints 106

9.2.1 Modeling Prismatic Joints 107
9.2.2 Modeling Revolute Joints 109
9.2.3 Modeling Rigid Connections 111
9.2.4 Collecting The Evidence 113

9.3 The Scaling Problem 115

10. EXPERIMENTAL EVALUATION OF MODELING THREE-DIMENSIONAL OBJECTS 119

10.1 Experimental Setup 119
10.2 Experimental Results 121

10.2.1 Experiment I: Box 122
10.2.2 Experiment II: Door 123
10.2.3 Experiment III: Refrigerator 123
10.2.4 Experiment IV: Laptop 124
10.2.5 Experiment V: Cupboard 124
10.2.6 Experiment VI: Shelf on Wheels 125
10.2.7 Experiment VII: Toy Train 125
10.2.8 Experiment VIII: Tricycle 126
10.2.9 Experiment IX: Elevator 126
10.2.10 Experiment X: Set of Drawers 127

10.3 Limitations and Failures 138
10.4 Summary 139

11. GROUNDED RELATIONAL REINFORCEMENT LEARNING 141

11.1 Relational Representation 143
11.2 Grounding the Relational Representation 146
11.3 Learning Manipulation Expertise 147

11.3.1 Problem Definition 147
11.3.2 Q-Learning 148
11.3.3 Representation of Q-Value Function 149

11.3.3.1 Finding Similarity between Link Properties 150
11.3.3.2 Finding Similarity between Kinematic Structures 151

11.3.4 Selecting Optimal Actions 153

11.4 Experiments With Planar Objects 154

11.4.1 Experimental Setup 154
11.4.2 Gathering Manipulation Knowledge 156
11.4.3 Transferring Manipulation Knowledge 158

11.5 Future Work 161

11.5.1 Planning With Delayed Reward 162
11.5.2 Experiments With 3-D Objects 162

12.CONCLUSION 169

APPENDICES

A. THE NORMALIZED 8-POINT ALGORITHM 174
B. THE MAX-FLOW MIN-CUT ALGORITHM 177
C. BUNDLE ADJUSTMENT 180
D. THE EXTENDED KALMAN FILTER 182

BIBLIOGRAPHY 184

LIST OF TABLES

Table **Page**

9.1 Modeling the kinematic structure of a train toy 115

10.1 Detecting the kinematic structure of a box 128

10.2 Detecting the kinematic structure of a door 129

10.3 Detecting the kinematic structure of a refrigerator 130

10.4 Detecting the kinematic structure of a laptop 131

10.5 Detecting the kinematic structure of a cupboard 132

10.6 Detecting the kinematic structure of a shelf on wheels 133

10.7 Detecting the kinematic structure of a toy train 134

10.8 Detecting the kinematic structure of a tricycle 135

10.9 Detecting the kinematic structure of an elevator 136

10.10 Detecting the kinematic structure of a set of drawers 137

LIST OF FIGURES

Figure **Page**

3.1 The Sense-Plan-Act Paradigm 19

4.1 Examples of objects categorized by their functionality 33

5.1 The proposed approach (conceptual level) 50

6.1 Examples of planar kinematic structures 52

6.2 Interactive perception for planar objects 53

6.3 Pliers vs. picture of pliers 54

6.4 Degrees of freedom of planar objects 56

6.5 The highly connected subgraphs and rigid bodies analogy 57

6.6 Graph representation for an object with two revolute degrees of freedom 58

6.7 Identifying planar prismatic joints 59

6.8 Three steps of interactive perception 61

6.9 UMan (UMass Mobile Manipulator) 63

6.10 Experimental results for planar objects with a single revolute joint 64

6.11 Experimental results for planar wooden toy 66

7.1 A skill for modeling three-dimensional objects 70

7.2 Relative Motion predictor 74

7.3 Short Distance predictor 75

7.4 Short and Long Distance predictors 77
7.5 Color Segmentation predictor 79
7.6 Triangulation predictor 81
7.7 RANSAC: extracting 2D lines 83
7.8 J-Linkage: preference matrix 86
7.9 Fundamental Matrix predictor 89
7.10 The contribution of individual predictors to segmentation 93
7.11 The contribution of all predictors to segmentation 94
8.1 Quality of three-dimensional reconstruction 97
8.2 Three-dimensional reconstruction: laptop 99
8.3 Three-dimensional reconstruction: train 100
9.1 Relative motion between the parts of a tricycle 103
9.2 Relative motion between bodies connected by a prismatic joint 108
9.3 Aligning relative motion trajectories 108
9.4 Detecting prismatic joints 110
9.5 A revolute joint between two bodies 111
9.6 Detecting revolute joints 112
9.7 Interacting with a train toy 114
9.8 Modeling the kinematic structure of a train toy 114
9.9 Scale ambiguity 116
9.10 Relative motion at different scales (drawers) 117
9.11 Relative motion at different scales (laptop) 117
10.1 Interacting with a toy train 120

10.2 Ten real-world objects 121
10.3 Identifying rigid bodies: box 128
10.4 Detecting the kinematic structure of a box 128
10.5 Identifying rigid bodies: door 129
10.6 Detecting the kinematic structure of a door 129
10.7 Identifying rigid bodies: refrigerator 130
10.8 Detecting the kinematic structure of a refrigerator 130
10.9 Identifying rigid bodies: laptop 131
10.10 Detecting the kinematic structure of a laptop 131
10.11 Identifying rigid bodies: cupboard 132
10.12 Detecting the kinematic structure of a cupboard 132
10.13 Identifying rigid bodies: shelf on wheels 133
10.14 Detecting the kinematic structure of a shelf on wheels 133
10.15 Identifying rigid bodies: toy train 134
10.16 Detecting the kinematic structure of a toy train 134
10.17 Identifying rigid bodies: tricycle 135
10.18 Detecting the kinematic structure of a tricycle 135
10.19 Identifying rigid bodies: elevator 136
10.20 Detecting the kinematic structure of an elevator 136
10.21 Identifying rigid bodies: set of drawers 137
10.22 Detecting the kinematic structure of a set of drawers 137
11.1 Two Examples of Kinematic Structures 144
11.2 Experiments with a planar kinematic structure (PRPRPRP) 157

11.3 Experiments with a planar kinematic structure (RPRPRPR) 158

11.4 Experiments with a planar kinematic structure (RRRRRRRR) 159

11.5 Experiments with a planar kinematic structure (RRPRPRRPR) 160

11.6 Experimental validation of knowledge transfer (PRPRPRP to PRPRP) 164

11.7 Experimental validation of knowledge transfer (RRRRR to RRRR) 165

11.8 Experimental validation of knowledge transfer (RRRR to RRRRR) 166

11.9 Experimental validation of knowledge transfer (RPR to PRRP to RRPRPRRPR) 167

11.10Three-dimensional simulation environment 168

CHAPTER 1

INTRODUCTION

The realization of autonomous manipulation, the act of intelligently interacting with the environment to accomplish a task, remains one of the biggest challenges for roboticists. To accomplish a manipulation task, a robot must have some knowledge about the various objects in its environment and how its actions can affect them. When this knowledge is available, the robot can use it to determine a sequence of actions, a plan, for achieving its manipulation objective.

This thesis asks the following question: how can a robot acquire information about the state of the world, the objects within it, and the effect of its own actions on the state of the world, so as to improve its ability to achieve manipulation objectives?

Assessing the state of the world can be very challenging. The robot's environments may contain many objects, each having various properties. Some of these objects may be either partially or completely unobservable due to visual obstruction, lighting conditions, or sensor limitations. In addition, other agents may cause an object to move, disappear, reappear or change its configuration frequently and unexpectedly.

The designers of industrial robots address this problem of acquiring information about the state of the world by fitting the environment to the robot's needs. Factories are designed such that perception is simplified or even eliminated. For example, activities on a factory floor may be modeled *a priori*, and objects in the environment can be known in advance. As a result, knowledge about the state of the world and the objects it contains is either given or can be acquired robustly for a few special cases. The set of possible actions is also known beforehand, and their impact on

the environment is well understood. Automation engineers have been successful in leveraging this knowledge to design both the robot and the environment, enabling the automation of complex manipulation tasks. The resulting improvement in the quality, quantity, and safety of manufacturing has had a lasting impact on our economy and productivity.

Replicating the success of industrial robots in environments such as our homes and offices will have a significant impact on our daily lives. Autonomous robots, capable of manipulating objects in these unstructured environments, will enable a large variety of exciting and important application. These applications include, for example, elder care and flexible manufacturing. Manipulation in these unstructured environment is the focus of this thesis.

Human environments, unlike factories, cannot be engineered for the robot. Therefore, the process of acquiring knowledge about the environment is significantly more challenging than in the industrial robots case. Because the environment cannot be modified for the robot, we must focus on designing the robot to be able to perceive its environment. Thus, there are two fundamental and strongly related challenges that a robot faces when trying to acquire knowledge in our everyday environments: what properties of the environment to measure, and how to interpret information that is only partially or indirectly observable.

The first challenge is choosing what properties of the environment to measure. To fully specify most unstructured environments, if at all possible, we require a high-dimensional state space. This complexity is due to the many objects contained in a typical environment; each characterized by a large variety of physical properties. It is impractical to acquire all of this information. Robots, therefore, must limit their measurements to aspects of the environment that best approximate the true state of the world, without compromising their ability to accomplish the given task.

The second challenge is understanding this information. A robot may only be able to obtain partial and noisy observations. There may be some properties that it cannot directly measure. Furthermore, the robot may be missing some necessary context or prior experience to correctly interpret its observations. This challenge raises questions such as what set of pixels constitutes an object? What information can be inferred by considering the relationship between observable objects? What interaction and with which object will reveal new, previously unobservable, information? And how will the robot's actions affect the state of the world? There may be no general answer to these questions. In many cases the information required to answer these questions does not exist in the respective sensor stream. Interpreting the information measured by the robot can be facilitated by providing it with concepts such as temporal relationship, cause and effect, and task-specific knowledge. Further understanding emerges with the use of experience.

Within the manipulation community, some researchers argue that a break-through in manipulation will be triggered by better sensing technologies. The development of better sensors is likely to make available more accurate models, even in unstructured environments. The problem of acquiring those models is therefore deferred to the developers of these novel sensors. This view is commonly taken, for example, in motion planning and grasp planning. In contrast, we believe that advances in sensing technology alone will not solve the problem of acquiring adequate models for manipulation. Such advances will result in more accurate observations, measured at higher frequencies. However, while better sensing is desirable, it does not resolve the challenges associated with deciding what properties of the world are most relevant to measure and how to interpret these measurements.

The recent successes of sampling-based motion planning and POMDP-based approaches to planning offer another perspective on addressing the challenges of manipulation in unstructured environments. These approaches benefit from the steady

increase in computational power. Their success may lead us to believe that perception can be solved by measuring everything, while relying on the ever-increasing computational power for processing large amounts of information. While advances in planning are essential to autonomous manipulation, we believe that better planning techniques alone cannot overcome the combinatorial explosion of information associated with manipulation in unstructured environments.

Acquiring the information necessary for manipulation in unstructured environments can also be addressed by perceptual techniques. Indeed, machine perception, in particular in the domain of computer vision [50], have recently made significant progress. Novel fundamental vision primitives, such as Lowe features [83], extract distinctive invariant features from images which can be used to reliably match different views of the same object. An increasingly powerful set of tools is being developed to address complex vision problems such as image segmentation [47, 51, 86, 113], semantic image retrieval [112] and semantic video search [95]. However, few of these advances in the realm of perception have had significant impact in autonomous manipulation.

Why have the advances in computer vision not had significant impact in autonomous manipulation? We believe that there are two closely related reasons. First and foremost, after initial and foundational work at the intersection of computer vision and robotics [66], both fields have progressed mostly independently. As a result, roboticists currently do not exploit the full potential of state-of-the-art computer vision techniques. There is a second important reason explaining why progress in computer vision does not impact autonomous manipulation. We believe that adequate perceptual capabilities have to be developed in the context of a specific robotic task. The perceptual information extracted from the sensor stream can then be tailored to the task to provide the appropriate feedback for ensuring robust and reliable task execution. In contrast, the majority of computer vision research is concerned

with general perception skills. The lack of impact that such skills have had in robotics is a result of the difficulty of developing general perception.

In this thesis, we hypothesize that perceptual skills should be considered within the context of a specific task. In this document, our focus will be on the task of manipulating rigid articulated objects. This is in contrast with the above "general" approaches towards perception. Considering a specific task provides the necessary structure for identifying what information needs to be measured (shape and motion), as well as how to interpret these measurements (kinematic model). Thus, we propose that developing task-specific perception will enable us to reduce the size of the state space associated with perception in unstructured environment, as well as the inherent uncertainty.

This thesis advances the state of the art in autonomous manipulation by proposing and implementing an approach for manipulating rigid articulated objects in unstructured environments. This approach is an example of our believe that progress in autonomous manipulation will be enabled by developing task-specific solutions. These solutions typically cannot be associated with a single academic subfield such as vision, machine learning or control. Instead, we expect each solution to lie on the intersection of these academic subfields, where the intersection is determined by the structure of the task. The approach we propose consists of two parts, each is an important contribution of this work. The **first contribution** is the development of perceptual skills which enable a robot to acquire and interpret the information necessary to model rigid articulated objects. The model computed by this perceptual skill provides our robot with the knowledge necessary for purposeful manipulation of the object.

The **second contribution** of this thesis is the development of a learning scheme, enabling a robot to obtain general domain knowledge for manipulation. Learning relies on the above perceptual skill. It provides our robot with the means to deter-

mine, through experience with the environment, what information is relevant for the manipulation of articulated objects. Furthermore, learning closes a loop around the robot's sensorimotor interactions with the environment, making it possible for the robot to make hypotheses about what information to measure and how to interpret these measurements. It also enables the robot to actively verify the correctness of these hypotheses.

We believe that the success of the proposed approach for manipulation in unstructured environments is due to our decomposition of the original difficult problem into simpler subproblems. Solving complex problems by decomposition is hardly a new idea. A common strategy for solving high-dimensional problems is decomposing them into sub-problems. However, not every decomposition is a good decomposition. For example, consider the expression $a^2 - 2ab + b^2$, which can be decomposed as $a^2(1 - \frac{2b}{a} + \frac{b^2}{a^2})$ or as $(a - b)^2$. Although both expressions are equivalent, the second decomposition is much simpler and requires less computation to evaluate. A good decomposition leads to simpler components (factors) that, when combined (multiplied), solve the original problem (equal the product).

Decomposition of hard problems is common practice in robotics. In fact, the dominant paradigm in robotics decomposes every problem into three subproblems: the sensing problem, the planning problem, and the acting problem. This division is matched by academic sub-fields such as vision, planning, machine learning, control, manipulation, and grasping. Because this decomposition is general, not taking into account any specific task, we believe that it should be reconsidered. We believe that a "good" decomposition does not naturally coincide with the boundaries imposed by traditional academic sub-fields. Instead, we hypothesize that a "good" decomposition typically exploits synergies that arise when these very boundaries are crossed.

The **third contribution** of this thesis is in proposing "interactive perception", a new category of perceptual skills, which exploits the synergies that arise when cross-

ing the boundary between action and perception. Interactive perception removes the traditional separation between action and perception by including interaction as part of the perceptual process. The robot generates strong visual signals during its interaction with the environment. This interaction can be used to expose information necessary for performing a task that is otherwise unavailable. Furthermore, the robot's knowledge of its embodiment and actions can be exploited to reduce the effects of uncertainty. We argue that interactive perception is a good decomposition, enabling the development of more reliable and robust perceptual skills.

In the following chapters, we develop a conceptual and algorithmic foundation for the problem of autonomous manipulation of rigid articulated objects. The ability to manipulate articulated objects is a prerequisite for a wide range of manipulation tasks (all prehensile manipulation tasks with rigid objects). The proposed solution addresses the general problem of mapping information from sensor space (what to measure) to task-relevant space (what and how to interpret our measurements). We believe that this work demonstrates that decomposing problems by considering the structure of the task is a promising approach, and will highlight our choice of decomposition and its advantages where appropriate.

1.1 Main Contributions

The main contribution of this thesis are:

1. A set of perceptual skills for acquiring and interpreting shape and kinematic models for rigid articulated objects

2. A learning framework, enabling a robot to obtain general domain knowledge for manipulations

3. An "enactive" approach to perception (interactive perception), which exploits the synergies at the boundary between action and perception

1.2 Thesis Outline

The remainder of the thesis is organized as follows: Chapter 2 motivates the work by reviewing selected works from within the psychology literature. Chapter 3 outlines the conceptual foundation of the thesis. In chapter 4 we review related work in robotics, learning, and computer vision. Chapter 5 provides an overview of the proposed approach for acquiring shape and kinematic models. In chapter 6 we provide the technical details of our solution for the simple case of planar objects. Then, in chapters 7, 8, 9, and 10, we present our solution to the general problem of modeling the shape, motion, and kinematic structure of 3-D objects. In chapter 11 we present a learning framework for gathering and generalizing manipulation expertise. Finally, chapter 12 concludes this dissertation.

CHAPTER 2

THE HUMAN ANGLE

Autonomous manipulation in human environments is very challenging. It requires making long term plans as well as reacting quickly to unexpected changes. It involves interactions with many different objects; some are well-known, other seen for the first time. To be successful, autonomous manipulation must allow for robust and reliable task execution in the presence of rich stimuli, and despite uncertainty. In this chapter, we draw inspiration from the only existing agents that can deal with these challenges—humans. The main objective of this chapter is to show, using a variety of examples from the psychology literature, that humans excel in manipulation in unstructured environments because of our ability to extract and leverage structure.

Autonomous manipulation in our everyday environments is hard. These environments are infinitely complex and dynamic, and human and robots alike have to perform manipulation tasks relying on bounded resources. And yet, we are able to grasp and manipulate dozens of objects every day, and can easily navigate through a previously unseen office or a crowded street. What is it, then, that allows us to deal with these complexities that seem so daunting for robots?

To answer this question, you may not have to go very far: walk upstairs, open the door gently, and look in the crib. What do you see? Most of us see a picture of innocence and helplessness. But, in fact, what we see in the crib is an extremely powerful learning machine. The tiny fingers and mouth are exploration devices that probe the alien world around them with high precision. The crumpled ears take a buzz of incomprehensible noise and flawlessly turn it into nouns, verbs and adjectives— a

language. The wide eyes constantly take in rich and complex light stimuli, and turn it into objects and people.

There is much we can learn from this robot-in-the-crib. We may be far from fully understanding how the human brain works, but we already know much about when it fails. It is through these failures that psychologists can learn about our brain's tricks and shortcuts. In the following, we will discuss four such "tricks". All of them have something in common: they reveal and exploit structure.

2.1 Explaining by Doing

Human children in the first three years of life are consumed by a desire to explore and experiment with objects. They are fascinated by causal relations between objects, and quite systematically explore the way one object can influence another object. They persistently explore the properties of objects using all their senses. A child might gently tap a new toy car against the floor, listening to the sounds it makes, then try banging it loudly, and then try banging it against the soft sofa. In fact, this kind of playing around with the world, while observing the outcome of their own actions, actually contributes to babies' ability to solve the big, deep problems of disappearance, causality, and categorization.

A child's explanatory drive tightly couples action and perception. This coupling was observed in the 80s by the psychologist Gibson [53, 54]. Gibson describes perception as an active process, highly coupled with motor activities. Motor activities are necessary to perform perception—and perception is geared towards detecting opportunities for motor activities. He called these opportunities "affordances". Gibson's theories continue to be relevant in psychology, cognitive science, and philosophy [100]. In a recent book, the philosopher and cognitive scientist Alva Noë describes an "enactive" approach to perception. He argues that perception is an embodied activity that cannot be separated from motor activities and that can only succeed if the perceiver

possesses an understanding of motor activities and their consequences [99]. Similar "enactive" theories have been proposed for the development of cognitive capabilities by Varela [137] and Gallagher [52].

Perceiving the world, making decisions, and acting to change the state of the world seem to be three independent processes. Thus, rejecting the functional separation of action and perception—at first—seems counterintuitive. An "enactive" approach to perception, however, may be essential for surviving in a high-dimensional and uncertain world. It provides a straightforward way to formulate theories about the state of the world and directly test these theories through interactions. "Enactive" perception imposes structure, which limits significantly what needs to be perceived and explained. An important contribution of this thesis is the realization of a robotic system which explains its environment by coupling action and perception. We refer to the robotic version of "enactive" perception as interactive perception.

2.2 Explaining by Forming Concepts

Cognitive and developmental psychologists distinguish between two types of knowledge: "Perception" and "Conception". The former is taken to be the structure of immediate experience, whereas the latter refers to knowledge that is (or can be made to be) explicit—that is, knowledge that is easily talked about and thought of.

Perception is the process of acquiring information from immediate sensor data. Conception, on the other hand, is the means to organize this information, to generalize it into categories of knowledge. These categories impose structure on the world, significantly reducing the complexity associated with perceiving and reasoning about unstructured environments.

Smith and Heise [118] argue that the dynamic nature of perceptual similarity is a causal force in the development of conceptual beliefs. In other words, forming concepts and categories, generalizing over our inherently ambiguous and high-dimensional per-

ceptual knowledge, is a strategy. Generalization reshapes the high-dimensional state space, thus reducing the complexity of the problem.

There may be various ways to form object categories from perception. Smith and Heise [118] show that 36-months-olds form different categories for different tasks. In their experiments, children were presented with an object. The experimenter gave the object a name, and then asked the children to decide whether other objects could be called by the same name (i.e. belong to the same category). The results show that the children were indifferent to variation in size or texture, but excluded objects that differed in shape. In a second experiment, toy eyes were affixed to each object. Now, children became sensitive to changes in texture as well as shape, and remained indifferent only to variation in size. Adding toy eyes change the abstract shapes into animated objects. The two experiments are now instances of two different tasks: sorting abstract shapes and sorting animated objects. Thus, the results demonstrate that the representation selected by children depends on the context of the task.

Early language introduces children to generalization. The first words that babies learn, around the age of 1 year, may be the result of grown-ups giving a sort of sportscaster's play-by-play of everything the baby does: "There, you picked up the car, oh, now you're putting it on the floor. Oh! you're pushing the car forward...". This kind of early language helps organize the world for babies. This early language is grounded in the baby's sensorimotor experiences with the real-world. However, rather than describing instances of perceptual experience, it is mostly used to describe categories, classes of perceptual experiences.

Language imposes structure. The label "car", for example, encapsulates much information. It abstracts away the complexities associated with acting and perceiving a specific instance of a car. As a result, knowledge about cars can be easily transferred and generalized. For language to be effective, it must be grounded in the child's sensorimotor interactions. Simply put, language requires that at some level, the label

"car" can be translated into perceiving and acting upon a specific car. Similarly, an important contribution of this thesis is in the development of a learning scheme that forms concepts which allow the robot to generalize its manipulation knowledge.

2.3 Explaining by Leveraging Social Cues

Social interactions represent a substantial portion of many daily activities in human populations. A common and well-described consequence of this interpersonal activity is that an individual's behavior, whether intentionally or not, is modified through interactions with others. Social interactions, particularly between children and adults, play a crucial role in the cognitive development of infants.

To exchange knowledge with others, a baby must acquire conversation skills. The pinnacle of social interactions is language. Language acquisition begins when an adult teaches a toddler the correct labels for objects [104]. Not surprisingly, a mother-toddler interaction consists of more than just teaching object labels. Toddlers also learn to identify the adult's body dynamics, and their implication on social interactions. For example, when a baby coos, its mother watches, then when the baby goes still, the mother coos [26]. What happens here is that the adult is providing structure; it shows the infant what matters and how to disambiguate the dynamics of a conversation and to relate different modalities: gesture, motion, appearance, and sound.

The importance of social interaction does not stop after childhood. Adults naturally adapt to each other's body dynamics. We spontaneously adopt, if only temporarily, a similar posture or tempo during a conversation with a peer. Another example is clapping of an audience where the applause occurs in unison with many individuals clapping as a single synchronized ensemble. The synchronization of body dynamics is important as it structures the exchange of information. It facilitates learning and even results in better memory of events [102].

An important characteristic of social interaction is the knowledge gap between participants. In many social interactions, one partner has more knowledge, and through social cues it enables the other partner to learn quickly and efficiently. The knowledgable partner does nothing more than providing task-structure, reducing the complexity and uncertainty associated with exploring the relevant domain.

The question of discovering the dimensions of the world that are relevant to a specific task is open. For humans, the process of discovering the dimensions of the world that are relevant to a specific task often requires the help of an experienced teacher. Robots may also one day receive important information through social interactions. Regardless of how task-relevant information is acquired, it seems that when it is available, hard problems can be made easy. Indeed, the main hypothesis in this thesis is that high-dimensional problems can be solved by exploiting the structure of the task, as this structure significantly restricts the associated state space.

2.4 The Art of Forming Explanations

In politics and sociology, divide and rule (derived from Latin divide et impera) is a combination of political, military and economic strategy of gaining and maintaining power by breaking up larger concentrations of power into chunks that individually have less power than the one implementing the strategy. Among scientists and engineers, this strategy is also known as divide and conquer [29]. The underlying principle remains the same; to solve a complex problem, one should search for sub-problems that can be solved easily and whose composition solves the original, more difficult problem.

Addressing easy problems first is hardly a new idea. In his developmental stage theory [117], Piaget postulates that human cognitive development employs the same strategy. Piaget identified 6 developmental stages that occur in the first two years. For Piaget, object grasping can only be considered after a period in which the infant

learns to coordinate its own actions and perceptions. Similarly, experimenting with an object requires that grasping, hand-eye coordination and goal oriented planning have already been solved. The importance of these stages is in the structure they impose on the environment. While the infant is still busy coordinating her actions and senses, she virtually ignores everything else around her. The result is a much simpler problem. Once a problem is solved reliably, our infant is ready to solve a more complex problem.

Failures illustrate best how fundamental this process of divide and conquer is for human development. A famous series of experiments by Piaget (1896-1980) established the notion of conservation and demonstrated that children mostly lack it up to the age of 7. One of the classic conservation experiments involved liquid quantity. A child is first shown two short, fat beakers. They are filled with colored water as the child watches. The child is asked to say when the two beakers have the same amount of water in them. Next the adult takes a tall, thin beaker and pours colored water from one of the short, fat beakers into one of the tall, thin beakers. The child is asked to compare the tall thin beaker to the short fat one. Most children under the age of 6 will point to the tall beaker and say, "This one has more in it." The child is swayed by the perceptual cue of height. Such a child lacks a conception of the conservation of liquid. Not having this concept, the child has no way to realize that volume stays the same when liquid is poured from one vessel to another.

A similar result is obtained when experimenting with the conservation of area (blocks that are spread out are believed to cover more area than blocks that are clustered together), conservation of mass, and of number. And, more recently, psychologists have been able to show that the mechanism of inference changes quite dramatically in the first few years of life [14]. These results suggest that humans tackle easy problems first, by making strong assumptions about the environment.

Once a certain set of skills is in place, it can be composed to yield a more advanced behavior.

Failure to solve complex problems does not disappear in adulthood. In fact, various experiments expose how context dependent is our ability to make rational decisions [6] or detect visual changes [114]. These experiments demonstrate that when multiple decompositions are available, the selection of how to tackle a problem is context based. A wrong decomposition will often render a task unsolvable, whereas with another decomposition it becomes trivial. Context-based decomposition is the basis for one of the important contributions of this thesis—interactive perception.

2.5 Summary

Psychological studies suggest that humans are naturally driven to explore their environment, generalize their observations, learn from others, and tackle difficult problems by splitting them into manageable chunks. These studies tell us that to survive in complex environments, humans use a variety of tricks to discover task-relevant structure. This structure limits the state space that must be considered for successful interactions, and therefore renders it feasible. If robots are to succeed in similar environments, we believe that they too must be capable of discovering and exploiting task-relevant structure.

The examples discussed in this chapter provide us with a bag of tricks that could accelerate progress towards robotic autonomous manipulation. In this thesis, we explore and utilize some of these tricks. Our robotic manipulator leverages the powerful idea of explaining by doing. It interacts with the environment to better understand how to manipulate objects. Like humans, our robot too relies on conceptual categories to reduce the complexity of its environment. This allows it to categories continuous noisy visual information into simple concepts such as links and joints. Finally, an important theme throughout this thesis is our decomposition of the general manipu-

lation problem into simpler, manageable subproblems. We believe that this specific decomposition is key to the success of our solution.

CHAPTER 3

DOING THE RIGHT THING

Autonomous manipulation is a challenging problem because it is a very high-dimensional problem. What is it that makes autonomous manipulation high-dimensional? The answer is very straight forward: autonomous manipulation is the problem of deliberately changing the state of the environment, and real-world environments are extremely complex. They contain much information to measure and analyze. Measuring and analyzing all this information is impractical. Fortunately, we believe, it is also unnecessary.

In order to solve autonomous manipulation, the dimension of the associated state space must be reduced. Simply put, the process of choosing what aspects of the world to measure and interpret must be very selective. How can a robot know what information to measure? How should it interpret these measurements?

Inspired by the way humans address these challenges (see discussion in chapter 2), we propose to tackle high-dimensional problems by identifying the lower dimensional task-relevant manifold of the state space. To do so, humans continuously explore and try to explain their environment, generalize observations, learn from others, and divide hard problems into lower dimensional subproblems. In this chapter, we provide our own answer to these two questions, proposing a similar set of tricks towards autonomous manipulation.

Most approaches towards autonomous manipulation follow the Sense-Plan-Act (SPA, see Figure 3) methodology. SPA is a systematic view of robots as input-processing-output systems; to handle the high-dimensional state space of autonomous

manipulation, SPA divides every problem into these there separate components. Sensing is responsible for inferring the state of the world, and acting is responsible for exerting forces onto a continuous environment to perform a task. In contrast to sensing and acting, planning is an internal process. It receives information about the world via the sensing component, computes a plan, and then provides the acting component with instructions to execute that plan.

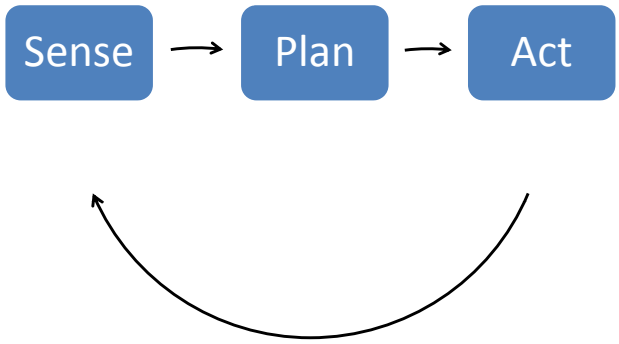


Figure 3.1. The Sense-Plan-Act Paradigm

Planners typically use symbols to make intelligent decisions. They expect to receive information about the continuous world in a symbolic representation, and are expected to provide discrete instructions for execution. This symbolic representation defines a basis that spans the state space accessible to the planner. Only those aspects of the world that can be represented using this basis, can be considered by the planner. Moreover, to perform a task, the planner must be able to represent the solution using the selected symbolic representation. The right symbolic representation, therefore, must be task-specific; it has to span exactly the state space required for solving the task.

How can we make sure that the solution computed by the planner achieves the task robustly and reliably? To do so, our symbols must be meaningful in the real-world. That is, the symbolic representation must refer to physical properties of the environment that the robot can perceive or act upon. This association between symbols and referents is called grounding [59]. To enable grounding, a robot must be capable of sensorimotor interactions with objects, events, actions, properties and states that its symbolic representation refers to.

The separation between sensing, planning, and acting that is encouraged by the SPA methodology was questioned by scientists from various fields. In chapter 2, we presented the ideas of psychologists, philosophers, and cognitive scientists such as Gibson, Noë, and others arguing against the separation between action and perception [52, 53, 54, 99, 100, 101, 137]. The trend towards eliminating the separation between action, perception and planning has also been present within the robotics community, for example in Brooks' behavior-based robotics [16, 17]. The view of perception and action as one integral process stands in contrast with the classical take on computer vision as inverse optics, proposed by David Marr in 1982 [87].

Indeed, the division between sensing, planning, and acting has created two significant difficulties in choosing the right symbolic representation. First, many researchers in perception have been trying to solve exclusively the problem of data acquisition. They design perceptual skills that are as general as possible, offering as much information as possible. Rather than reducing the dimension of the state space associated with manipulation, this approach generates representations that include a symbol for any physical property that can be measured. The resulting state space contains much information that is not required for solving the task, and is therefore rendering planning unnecessarily difficult.

The second difficulty that arises from this division is the result of separating sensing and acting. The symbols provided to the planner are acquired by the sensing

component. As such, these symbols only refer to sensor data, overlooking sensorimotor features and the manipulation task. This representation does not allow for action to be part of the perceptual process (i.e.: it prevents robots from taking an enactive approach to perception). The unnecessary constraint imposed on the space of referents limits the power of the symbolic representation. Although excluding sensorimotor symbols reduces that size of the state space, the excluded symbols may be the ones required for spanning the state space in which the solution to the task is to be found.

In order to do "the right thing" [116], we propose and examine three hypotheses for tackling high-dimensional autonomous manipulation problems:

1. **Representation:** The symbolic representation of the environment must be task-specific

2. **Grounding:** Symbols must be grounded in the robot's sensorimotor interactions with the environment

3. **Hierarchy:** Within a task-hierarchy, simple subtasks enable more complex tasks by "reshaping" the state space

Our first hypothesis suggests that the robot's symbolic representation should be defined by the task. In other words, the robot should represent only those aspects of the world that are relevant for its task. This approach is motivated by humans studies exposing task-induced bias in representation selection (see discussion in chapter 2). Limiting the richness of the representation restricts the size of the associated state space.

Our second hypothesis suggests that the robot's symbolic representation must be grounded in its sensorimotor interactions with the world. This approach is motivated by humans studies showing the benefits of an enactive approach towards perception

(see discussion in chapter 2). To enable grounding of symbols in the robot's sensorimotor interactions, we argue that the artificial division of research into perception, planning, and action has to be eliminated. In our work, we will show that by defining symbols that represent both action and perception, we provide planning with the appropriate abstraction of the physical world to enable a simple solution.

Our third hypothesis is that in order to solve a complex task, we should decompose it into simpler subtasks. The resulting hierarchy of tasks facilitates the discovery of task-relevant structure in two important ways. First, it reduces the complexity of the task for which the robot needs to find relevant structure. Second, and arguably more importantly, the solution for each subtask "reshapes" the space of sensorimotor features. As a result, it becomes easier to discover relevant structure for the more complex task.

We believe that a robot can learn task-relevant structure by relying on task-hierarchy. To verify that, we propose to use a solution to a sub-task to reshape the symbolic space in which a more complex task has to considered. We expect this hierarchy to enable the discovery, through learning, of high-level task-relevant structure. With this knowledge of the task-relevant structure, we believe that it becomes easier to achieve the given complex manipulation task.

3.1 Exposing and Exploiting Task-Relevant Structure

All three hypotheses stem from our belief that feasible tasks are tasks for which there exists task-related structure in the physical world. This structure is precisely what needs to be symbolically represented to enable the planner to solve the given task. Structure can be extracted by considering, for example, the laws of physics, temporal relationship between events, the notion of cause and effect, or any other type of logical or probabilistic relationship. Choosing a good symbolic representation, therefore, can be achieved by considering task-relevant structure in the physical world.

In this work, we propose several methods for exposing task-relevant structure, which in turn enables the robot to use the right, grounded, task-relevant sensorimotor symbols. One method simply relies on a human teacher: the robot is given *a priori* knowledge about the structure relevant to the given task. It then uses this knowledge to acquire task-relevant sensor information, and represents it with symbols that are meaningful for achieving the task. Another method learns task-relevant structure. Learning is realized as trial and error induction, guided by feedback from the consequences of the robot's sensorimotor interactions with its environment.

To verify the above hypotheses, this thesis will focus on the problem of manipulating articulated objects. In the next chapter, we will take a look at related work addressing the same problem. Of particular interest will be to consider how these works leverage task-structure, provide a useful decomposition for solving more complex problems, and whether they are grounded in the robot's sensorimotor interactions with the environment.

CHAPTER 4

RELATED WORK AND BACKGROUND

In this chapter we provide a review of relevant literature. We divide the discussion into two parts: In section 4.1, we analyze three categories of approaches towards manipulation in unstructured environments. This analysis will provide the context necessary to understand our own approach to manipulation in unstructured environments. Then, in section 4.2, we provide some necessary background in the robotic manipulation, computer vision and machine learning literature. This background is important because the subfields and methods we discuss are essential for understanding the core components of our system. The work we present in the remainder of this thesis relies on or further develops this literature.

4.1 Related Work

This section divides the literature in autonomous manipulation into three large categories of approaches. The first category relies on advances in computation to reduce the computation time necessary to solve hard problems. The second category relies on improvement in sensor technology to enable the acquisition of important information. And finally, the third category of approaches leverages task-structure to reduce the complexity of the state space associated with the problem. Our goal is to demonstrate that methods in the third category are more general, reliable and robust. This is important because the work presented in this thesis falls into this category of approaches.

To manipulate an articulated object, we need to acquire information about its kinematic structure. Kinematics is the study of motion alone, without considering the forces that cause it [31]. It is restricted to the position, velocity, acceleration, and other higher order derivatives of the position variables with respect to time. Hence, the study of the kinematics of objects refers to all the geometrical properties of an object, as well as its time-based properties of motion. Understanding the kinematic structure of an object is important for manipulation because it is closely related to the function of the object, and therefore its intended use.

We describe the kinematic structure of an object using two types of symbols: links and joints. A link is a rigid body where the distance between two arbitrary points remains constant. A joint connects together two links. The type of joint defines the motion constraints between the links. We describe the relationship between two links using as many as 6 parameters (three for position and three for orientation). The number of parameters needed to describe this relationship determines the degrees of freedom connecting the links. When two links are connected by 6 degrees-of-freedom, we refer to them as "disconnected".

With knowledge of the kinematic structure of an object, it becomes straight forward to plan a manipulation sequence for changing its configuration. It is therefore not surprising that the robotics and computer vision communities have recently begun investigating the problem of autonomously acquiring kinematic models from sensor data. In the following we present three categories of methods for acquiring object models. These categories should be seen not solely in the context of model acquisition, but more generally as approaches for addressing the challenges of autonomous manipulation in unstructured environments.

4.1.1 The Computational Approach

The computational approach for problem solving is best understood in the context of search problems. Any computable task, be it manipulating an object, grasping a tool, or moving to the end of the corridor, can be reduced to a search problem. And, every search problem can be solved using an approach known as "generate and test", a trivial yet general technique which consists of systematically enumerating all possible candidates for the solution and checking whether each candidate satisfies the problem statement.

For over 50 years now, AI researchers have been occupied with developing methods for reducing the complexity of such search problems. Indeed, for some domains, or under specific assumptions, the dimension of the state space that must be searched can be reduced enough to render search feasible. Unfortunately, this approach does not scale to most autonomous manipulation problems, and a general method for reducing the dimensionality of the state space is yet to be discovered—and may even not exist.

If the dimensionality of the state space cannot be reduced, how can we solve search problems? The computational approach suggests that we rely on the exponential growth in computation power. Proponents of the computational approach believe that the exponential increase in computational power guarantees that we will eventually be able to solve search problems of arbitrary dimensionality.

How can we leverage the increasing computational power to solve the problem of acquiring the kinematic structure of an object from sensor data? First, we cast this problem as a search problem. The input to our problem is a set of point features along with their position over time. These features are associated with different parts of the object. Our task becomes that of generating all possible feature-rigid body associations and all possible joints connecting every pair of rigid bodies. Then, we must

go over all generated hypotheses, and test them against the observed positions over time. Finally, search will identify the hypothesis that best explains the observations.

A popular approach towards modeling the motion of one or more rigid bodies, allowing for either a static or a moving camera, is known as structure from motion [30, 55, 62, 115, 144, 145]. We note that in the standard formulation of structure from motion, the input is two-dimensional (i.e. point features on the camera plane). When three-dimensional observations are available, the problem is somewhat easier, yet not fundamentally different.

Most existing methods for recovering structure from motion assume that the world is static, i.e. they can only handle a single object that is the entire world, while the camera is allowed to move freely. Structure from motion methods can be divided into global optimization methods (e.g. bundle adjustment [62, 82]) and recursive estimation methods (e.g. Extended Kalman Filter [21, 132]). Global optimization methods require that the entire input sequence is available a priori, whereas recursive estimation methods improve their shape and motion model over time. Some methods for recovering 3-D shape from motion remove the single rigid body assumption [55]. These algorithms, however, are computationally complex and require a long sequence of images.

Yan and Pollefeys [144] rely on structure from motion [21, 132] to obtain 3-D feature trajectories, and then use spectral clustering to identify rigid bodies and their kinematic relationships. This work assumes affine geometry[1] of the scene and only considers revolute joints. Ross et al. [115] follow the same principle, using maximum likelihood estimation instead of spectral clustering. The strength of the latter two

[1]Affine geometry is the study of geometric properties which remain unchanged by affine transformations Affine transformations relate between two vector spaces with a linear transformation followed by a translation

algorithms is that they can handle bodies that undergo slight deformation during the motion.

All of the aforementioned approaches only handle revolute joints and make strong assumptions to simplify the perception problem. To determine the kinematic model, these methods all rely on motion information. This motion, it is assumed, is generated by some external force. Thus, these methods are not suitable for unstructured environments, in which this motion is not guaranteed. Finally, because motion is the only source of information used by these methods, relevant task-structure encoded in color, texture, or spatial contingency is ignored. This additional task-structure, as we will see in later chapters, can significantly simplify the problem of acquiring the kinematic structure of an object.

In contrast, the work described in this thesis exploits the robot's embodiment to generate object motion. It leverages a rich set of visual and motion information, and applies to all types of 1-degree-of-freedom joints. Our work also solves a structure from motion problem. However, unlike the computationally intensive methods above, we determine the different rigid parts in the scene, and only then compute the three-dimensional shape and motion of each part—a much easier problem to compute.

4.1.2 The Technological Approach

We have argued that autonomous manipulation in unstructured environments is a challenging problem because of the inherent uncertainty and the associated high-dimensional state space. Is it possible that some problems are hard to solve not because the associated state space is too large, but rather because it is too small?

According to the technological approach, the answer is yes. Proponents of this approach argue that better sensing and actuation technology would reveal information that is necessary for solving certain problems. Information that is, perhaps, currently unavailable. Technological advances will enable us to sense new physical phenomena,

detect more information at higher frequencies and with remarkable accuracy. This view is motivated by scientific success stories such as the invention of the microscope, which dramatically changed our understanding of biology. Similarly, in robotics, the introduction of laser-based range sensing has enabled Simultaneous Localization And Mapping (SLAM).

We note that the technological approach is closely related, and even depends on, the computational approach. Better sensors provide more information, at a higher rate and with higher accuracy. The resulting state space is significantly larger. Thus, to take advantage of the increase in available information, we require faster processing.

The technological advance that is probably most relevant to the task of acquiring kinematic models of novel objects is the invention of various three-dimensional sensing technologies. The structured light camera is one such technology. It projects a known pattern of pixels (often grids or horizontal bars) onto a scene. The way that these deform when striking surfaces allows a vision system to calculate the depth and surface information of objects in the scene. Sturm et al. [123, 124], for example, use a structured light camera (see Willow Garage PR2's camera [134] and Microsoft's Kinect [140]) to acquire 3-D information about a single moving body. They then learn a model for the degree of freedom that constraints the motion of that body.

Structured light cameras suffer from a disadvantage due to the fact that the projected light is typically uniform across the image plane. As a result, much of the available three-dimensional information may be associated with non-object parts of the scene. This would result in much wasted computation, and lower resolution in relevant parts of the image.

Motion capture is the process of recording movement and translating that movement onto a digital model [58, 122]. Motion capture systems were originally used as a photogrammetric analysis tool in biomechanics research. As such, they seem particularly relevant to the task of identifying the kinematic structure of objects. Motion

capture systems typically consist of a number of synchronized cameras, an image acquisition system, a capturing area, and a special set of markers. The markers cover all relevant parts of the object (links and joints). The result of a motion capture session is a set of trajectories associated with the various features tracked by the system.

Ross et al. [115] capture two-dimensional trajectories to infer the kinematic structure of an object. This work begins by grouping the two-dimensional features into rigid body parts. An Expectation-Maximization (EM) algorithm is then used in order to learn the correct three-dimensional position of each rigid body from the 2D observations. At each new iteration, the algorithm further improves its model. Finally, it would converge to the correct kinematic structure. This work assumes that links have the shape of a stick, and that joints are positioned between the ends of two sticks. These assumptions are strong, and would fail in the case of more complex links, or when joints are attached to the middle of a link (e.g. scissors).

Kirk et al. [77] also rely on markers to capture three-dimensional human motion. Skeletal information is then inferred from the captured motion. To identify the rigid parts of the body, the standard deviation in distance between all markers is computed. The kinematic constraints between rigid parts are computed by solving a nonlinear optimization problem, where the location of a revolute joint minimizes the distance to all markers. We note that this method only applies to revolute joints.

Sturm et al. [125, 126] take a similar approach towards learning the kinematic model of a robotic manipulator arm. This work introduces a flexible kinematic model of a manipulator based on Bayesian networks. This model is able to simultaneously identify the number of degrees of freedom while learning the geometrical relationships between the links as a function of the joint angles. This work also only allows for revolute joints. Both Kirk et al. [77] and Sturm et al. [125, 126] rely on artificial markers to recover the three-dimensional trajectories of rigid parts.

Laser radar (LADAR) is another technology [4] which is frequently used to acquire precise 3-D models of rigid bodies [12, 18, 109]. In contrast with the previous vision-based sensors, LADARs offer very high precision and reliability. However, to the best of our knowledge, three-dimensional laser scans have not been used yet to model kinematic chains. This is probably due to the fact that laser radars do not naturally lend themselves to tracking changes in dynamic scenes.

More recently, researchers have proposed radio frequency based technology to facilitate object recognition. This technology uses a novel RFID 3-D tag which can identify not only an object, but also its position and orientation [56, 143]. RFID technology is cheap and reliable. Its usage is increasing rapidly, and already many manufactured goods are equipped with such tags. In contrast with the above technologies, the usage of RFID tags is an attempt to circumvent perception. As such, un-tagged objects, erroneous tags or a mistake in interpreting the content of a tag are all events from which a robot would not be able to recover.

All of the above methods, with the exception of the RFID tags, share the same view of modeling the kinematic structure of an object. First, they rely on new technology to acquire more accurate and complete 3-D information of the object and its motion. Then, they analyze this motion to determine the kinematics of the object. Similar to the computational approach, all methods in this category also ignore task-relevant structure such as color, texture and spatial contingency. These methods depend on motion observations alone, and are therefore unable to generalize across objects that, for example, appear to be visually identical. We believe that this indicates a wrong decomposition. Our approach towards kinematic modeling uses a different decomposition of the problem, as it relies on multiple sources of information. Finally, all of the aforementioned methods require that an external force generates object motion. In contrast, in this thesis we will propose to actively put objects

in motion. Furthermore, we will demonstrate a learning scheme for deciding what motion to generate based on the robot's experience.

4.1.3 Leveraging Task Structure

Unstructured environments are very rich in information and opportunities to act. At a first glance, it seems that manipulation tasks in these environments must be associated with a high-dimensional state space and much uncertainty. We hypothesize that, in practice, solvable tasks are embedded in a lower dimensional state space manifold, with limited uncertainty. Thus, a prerequisite for solving manipulation tasks in unstructured environments is identifying this task-relevant low-dimensional subspace. The following methods leverage knowledge of the manipulation task at hand to reduce the dimensionality of the state space and restrict uncertainty.

Green et al. [57] consider the recognition of articulated objects as a reasoning task. To identify scissors in a sequence of images, this work first discovers the various functional parts (handles, finger holes, blades, etc.) of the object. Then, assuming motion of the object across frames, it uses the sequence of images (in which the object is shown in varying configurations) to determine its kinematic structure. Finally, the system can determine whether the parts, the relationship between parts and the object's kinematic structure all agree with the functionality of scissors. By understanding the relationship between motion, appearance and functionality in the case of scissors, object recognition becomes much simpler. This understanding directs the attention of the perceptual process to the most task-relevant dimensions of the state space.

Sutton et al. further develop this idea of functional categorization (see Figure 4.1). In [127], a three-dimensional model of a target object is constructed from range and color information. This model is then used, in conjunction with knowledge of physics, shape-suggested functionality, causation, stability, etc., to label the object's potential

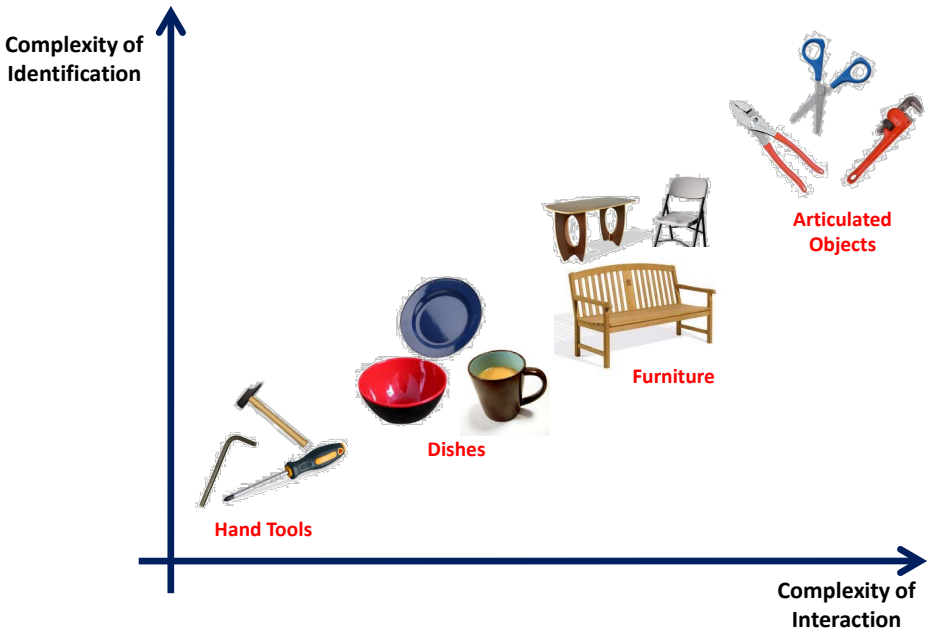


Figure 4.1. Examples of objects categorized by their functionality. The complexity of object recognition is considered here to follow the complexity of the necessary interaction to identify the functionality of an object.

functionality. Finally, the system computes and executes plans to confirm the object's functionality through interaction. This work proposes an interactive approach towards object recognition. Its experimental component is limited, however, and only includes chairs and cups. The introduction of deliberate actions to facilitate perception allows the robot to prove (or disprove) its hypotheses about the functionality of an object in an efficient and direct way. Similarly to [57], knowledge of the task (functional behavior) facilitates the identification of the lower dimensional task-relevant subspace, within which object recognition can be solved easily. We note, however, that in both cases knowledge of functional behavior is given to the robot, and not grounded in its sensorimotor skills.

Anguelov et al. [5] obtain kinematic models of doors by tailoring perception to the kinematics of doors. This work makes a strong assumption about the kinematic structures to be observed. However, by exploiting the structure imposed by this

strict task, the resulting perceptual skill can acquire reliable and accurate kinematic models.

Sturm et al. [123, 124] learn models of kinematic joints from three-dimensional feature trajectories, generated from deliberate interactions with the environment. This approach does not attempt to identify rigid bodies in the scene; it assumes that only one rigid body is moving at a time, and that this body is connected to a static object (or the background). This work fits a set of models to the observed motion of a rigid body. Each rigid body (or part) is considered as a vertex in a graph. Edges in the graph represent the type of joint between a pair of parts. A possible arrangement of the object parts is represented by a spanning tree. There are many possible spanning trees. The tree which maximizes the expected likelihood of a new observation is selected as the articulated object's model. This process is recursive. In every step, as a new observation becomes available, the model (spanning tree) is updated. The correct kinematic structure will be discovered after sufficient interaction with the object. The three dimensional structure and motion of the parts have been computed based on artificial markers (earlier versions of this work) or using a 3-D structured light camera (more recent). This work is limited to closed kinematic structures[2], and therefore cannot model many everyday objects, such as scissors, a stapler, or a rolling office chair. The interaction, together with the closed kinematic chain restriction, significantly constraint the state space. As a result, learning a joint label from 3-D motion becomes feasible, but only for this specific subcategory of articulated objects.

Taycher et al. [131] track human motion to learn an articulated model. They assume a known number of links, and determine the kinematic constraints between pairs of links using maximum likelihood. This work assumes the object can be mod-

[2]A closed kinematic chain consists of a series of links with the distal end fixed to the ground or some immovable point. Open kinematic chains consist of a series of links with the distal end free in space

eled as an articulated tree. It is therefore suitable for modeling the human body, but not for kinematic chains or loops. Similar methods were proposed by Hu et al. [67], Poppe et al. [108], and others.

All of the above methods exploit the structure of the task to identify the task-relevant lower dimensional state space. As a result, the original hard high-dimensional problem becomes manageable. Some of the above methods leverage the robot's embodiment to purposefully manipulate the environment, in order to reveal task-relevant information. The work that we present in this thesis further develops this interactive approach towards perception.

None of the above examples is suitable for manipulating novel articulated objects in unstructured environments. They all either assume much a priori knowledge or significantly constraint the type of target objects. Moreover, all of these methods rely solely on motion to infer the kinematic model. In contrast, our approach will leverage much more task-relevant information in the form of motion, but also color, texture, and spatial contingency, while making only very general assumptions.

4.2 Background

Our approach to the problem of perceiving the kinematic structure of articulated objects in unstructured environments decomposes the problem into three steps. First, we extract the different rigid parts of an object. Second, we determine the shape and motion of each part. And finally, we compute the kinematic relationship between pairs of parts.

To determine what are the different parts of an object, without assuming prior knowledge, we must observe relative motion between the object and its environment and between the parts of the object. In unstructured environment, we cannot assume that this relative motion will occur. Instead, our solution relies on the robot's embodiment to generate the necessary motion. We have already seen an instance of

this enactive approach to perception when discussing the human angle in chapter 2. In section 4.2.1, we further develop this discussion and provide a review of previous work in robotics taking a similar approach.

To determine the shape and motion of each rigid part, we rely on the well-studied field of structure from motion. In section 4.2.2, we discuss the state of the art in structure from motion. The discussion provides the necessary background to understand our own contribution to structure from motion, which is discussed in chapter 8.

Determining the kinematic structure of an object can be seen as an iterative process of interacting with it, modeling the shape of the newly discovered parts, and finally computing the kinematic relationship between every pair of parts. Computing the kinematic relationship between pairs of parts is relatively straight forward. We will review the necessary literature when we present our solution to this problem in chapter 9.

For most applications in unstructured environments, it is essential that this iterative process is efficient. In other words, the robot should interact with the environment such that the information gained as a result of each action is maximized. In chapter 11, we propose a relational reinforcement learning framework enabling a robot to gather and generalize manipulation expertise. The goal of this learning framework is leveraging the robot's experience in manipulation to improve the efficiency of perceiving and manipulating new objects. In section 4.2.3, we provide the background on (relational) reinforcement learning.

4.2.1 Interactive Perception

To acquire information about the kinematic structure of an object, without assuming prior knowledge, we must observe relative motion between the parts of the object. To guarantee robustness in unstructured environments, we cannot rely on this relative motion to occur. Instead, the robot must be able to generate that relative

motion. We refer to this idea of interacting with the environment in order to facilitate perception as interactive perception. Coupling perception and manipulation has two complementary advantages: perception can focus on signals relevant to the specific manipulation task, while physical interactions can reveal properties of the environment that would otherwise remain hidden. In our case, by interacting with objects, their kinematic and dynamic properties become perceivable.

The proposed coupling of perception and manipulation naturally extends the concept of active vision [2, 9]. Active vision allows an observer to change its vantage point so as to obtain information most relevant to a specific task [11]. This thesis goes one step further: it allows the observer to manipulate the environment to obtain task-relevant information. Due to this coupling of perception and physical interaction, we refer to the general approach as interactive perception.

In interactive perception, the emphasis of perception shifts from object appearance to object function and the relationship between cause and effect. Irrespective of the color, texture, and shape of an object, the robot has to determine whether the object has the functional affordances necessary to accomplish a given task. Interactive perception thus represents a shift from the dominant paradigm in computer vision and manipulation. This shift is necessary to enable the robust execution of manipulation tasks in unstructured environments.

Interactive perception is supported by psychological research, showing that humans use the functional affordances of an object for its categorization [10, 20]. Further, psychologists, philosophers, and cognitive scientists such as Gibson, Noë, Varela, Gallagher and others [52, 53, 54, 99, 100, 137] have proposed an "enactive" approach for perception (see the discussion in Chapter 2). We believe that these works lend support to our view that the simplest solution to manipulation in unstructured environments does not necessarily have to follow a strict separation between sensing, thinking and acting.

There is much work in robotics that is consistent with our view. Active vision [2] and visual servoing [69, 71], for example, tightly couple perception and action for a special set of skills. Compliance in embodiment [3, 38] can also be leveraged to replace aspects of traditional computation with morphological computation [105, 106], effectively crossing the boundary between embodiment and action. The work of Edsinger and Kemp in mobile manipulation [45] also follows similar ideas. They "let the body do the thinking", an idea similar to morphological computation [105, 106]. They too emphasize the importance of task-relevant perception, a consequence of close coupling between action and perception. Other prior work also enhance perception with deliberate physical interactions to acquire information about the functionality of objects [13, 57, 121, 127].

To the best of our knowledge, there are only two examples of interactive perception in the literature. Fitzpatrick and Metta [49, 89] facilitate object segmentation by pushing, and Christiansen et al. determine a model of an object's dynamics by observing its motion in response to deliberate interactions [23].

4.2.2 Structure from Motion

Throughout this thesis, the robot's visual information comes in the form of tracked points features. These point features are a noisy planar projection of the real three-dimensional world. Moreover, tracking is often noisy. To determine what are the different parts of the object, we have to cluster these point features into rigid bodies. Interactive perception is a key ingredient in our solution to this problem: it generates relative motion, which in turns facilitate segmenting an object into its rigid parts.

After we have identified which subset of the point features belongs to a single rigid body, we are ready to reconstruct its shape and motion. We can leverage the fact that there is a strong relationship between the motion of each point and the shape and motion of the corresponding rigid body. The ability to exploit this rigid body

constraint supports our belief that this is the right decomposition of the problem into subtasks in which the inherent task-structure is easy to exploit.

Recovering this relationship between motion and structure is the goal of the well-studied field of structure from motion. The origins of structure from motion can be traced back to the second half of the 19^{th} century. Researchers within the field of photogrammetry aimed to extract geometric information from images. Photogrammetry begins with a set of manually-selected features in a sequence or pair of images where the object or camera has moved. It then uses various techniques to extract depth information (three-dimensional structure).

Over the last two decades, we have seen remarkable progress in three aspects of structure from motion. First, the constraints imposed on the motion of features in two (or more) images, assuming rigid scenes and moving camera, have been formalized. These constraints are now well understood even in special cases such as degenerate motion or for observations produced by un-calibrated cameras [62]. Second, a variety of salient and robust feature descriptors and feature detection methods have been developed [19, 60, 83]. And finally, the introduction of machine learning into computer vision enabled robust and reliable feature matching methods [48].

This progress has led to a new generation of structure from motion methods. In these methods, an initial estimate of the three-dimensional structure of a scene is obtained using pairs of images. The initialization is followed by a bundle adjustment step [90, 135], which minimizes the re-projection error and renders the estimation globally consistent. These methods can recover the structure of the scene, as well as the camera locations, up to a projective transformation for un-calibrated cameras. With calibrated cameras, these methods can typically compute a solution up to scale. These methods, however, cannot be applied to mobile manipulation problems, where vision typically involves a continuous stream of images. The first step of these methods relates feature motion across pairs or triplets of images, ignoring any global

considerations. And the second step, bundle adjustment, suffers from a computational complexity that is cubic in the number of frames. This computational (in)efficiency renders these methods impractical for analyzing long sequences of images.

Recently, researchers began addressing the problem of computing structure from motion, in real-time, for a long sequence of images. The potential applications of real-time 3-D shape and motion estimation are many, ranging from augmented reality to grasping, and from 3-D navigation to object manipulation. These recent algorithms can be roughly divided into two categories. Methods in the first category select a sparse set of key-frames [1, 27, 60, 78, 79, 80, 94, 96, 97, 98]. They then apply pairwise structure from motion followed by bundle adjustment over the subset of the key-frames that is closest to the current frame. Methods in the second category apply Bayesian filtering techniques [8, 21, 22, 24, 25, 32, 33, 35, 37, 40, 43, 64, 70, 72, 73, 84, 92, 111, 119]. These methods iteratively integrate information from the current image into a multidimensional probability distribution. This probability distribution captures the information gathered along the sequence of images, until the current frame. We now discuss the state of the art in both categories.

4.2.2.1 Key-frame Methods

The first demonstrated real-time key-frame method was visual odometry [97, 98]. To initialize a set of visual features, this method relies on Harris corners [60]. The features are tracked along the sequence using normalized correlation. The three-dimensional structure of the scene and the camera's motion are estimated, in the key-frames, using the 5-point algorithm [96]. Experiments with visual odometry demonstrated errors as low as 1% relative to trajectories of hundreds of meters. Recent visual odometry methods achieve improved performance by leveraging stereo information [27] or combining stereo with inertial measurements [80]. In [94], visual

odometry was enhanced with local bundle adjustment, to compute 3-D reconstruction using trajectories of about 500 meters.

Visual odometry methods provide real-time performance, while demonstrating impressive accuracy in depth estimation over long sequences and for long camera trajectories. Because they employ bundle adjustment only locally, they cannot guarantee globally consistent estimation. As a result, these methods are subject to drift. This shortcoming renders them insufficient for most mobile manipulation problems, where it is expected that a system operates for extended periods of time. For such systems, even the slightest drift in the three-dimensional model will result in gross errors.

To address the shortcoming of visual odometry, Klein and Murray [78, 79] have proposed a key-frame method based on two parallel processing threads. The first thread performs camera tracking. It assumes that a map of natural features is available. The second thread constructs a consistent map by applying global bundle adjustment, over selected frames of the sequence. This method achieves real-time performance and reliable reconstruction when exploring room-sized environments. It is also robust against sudden camera motions. Similar in spirit is the work of Konolige and Agrawal [1]. Here, one thread performs visual odometry using stereo together with inertial measurements [80], whereas the second thread marginalizes out all features to produce a skeleton of frames in which the constraints of the whole trajectory are updated whenever a new key-frame is introduced.

4.2.2.2 Bayesian Filtering Methods

Bayesian filtering methods represent the state of a system with a multidimensional probability distribution. Measurements are processed and integrated into this probability distribution at every time step. These methods are ideal candidates for processing long sequences of visual information because their computational complex-

ity is proportional to the size of the state space. This is in contrast with key-frame based methods, where complexity is proportional to the number of frames.

Bayesian filtering has been extensively researched within the robotic navigation community. SLAM (Simultaneous Localization and Mapping), estimates a map of the environment, as well as the position of the robot inside that map. Estimation is computed in real-time, and relies on information gathered by sensors mounted onto the navigating robot. Most instances of SLAM rely on odometry or inertial measurements to determine relative changes in the robot's position, and laser scanners to determine the distance to obstacles.

Recently, due to the availability of cheap, reliable cameras, visual SLAM is gaining popularity. Graph-SLAM, for example, estimates an a posteriori solution for the entire trajectory of the robot. It generates a map from all measured data. Graph-SLAM addresses 3-D estimation as an online global optimization problem, much in-line with the global optimization *à la* bundle adjustment [35, 84]. Alternatively, visual SLAM can be formulated as an online filtering problem. Under this formulation, sequential processing of visual data is performed in order to achieve a globally consistent mapping [8, 21, 72, 111].

The extended Kalman filter (EKF) was the first filtering technique to tackle a navigation-in-the-plane SLAM task [37, 70, 119]. EKF was also the first filtering technique to demonstrate real-time performance in visual SLAM [32, 33]. Other Bayesian filters have been proposed to tackle classical SLAM: particle filters [92], sum of Gaussians filtering [40] and the unscented Kalman filter [73]. Their success resulted in attempts to apply the same filters to visual SLAM [43, 64].

Recent research focuses on both increasing the robustness and reducing the complexity of EKF-based visual SLAM. For example, the method in [22], reduces the search region by using prior information. This method also maintains multiple matching hypotheses, increasing the algorithm's robustness against outliers. Civera et

al. [24, 25] apply an efficient 1-Point RANSAC, exploiting knowledge from previous filtering iterations to remove outliers.

4.2.3 Relational Reinforcement Learning

One of the important contributions of this thesis is in developing a learning framework enabling a robot to generalize and transfer the manipulation expertise acquired by the proposed skill. This requires that the robot learns a policy for interacting with its environment in an efficient way in order to achieve its manipulation objective.

Determining an efficient sequence of actions under uncertainty is an important component of decision making problems. Many of these problems can be formalized as Markov decision processes (MDP) [110]. Many dynamic programming techniques have been developed to maximize the expected utility of an agent in an environment described by an MDP. The advantage of these techniques is that they can handle uncertain outcomes of stochastic actions.

One of the more popular dynamic programming techniques is reinforcement learning [128]. An important advantage of reinforcement learning is that it does not require a transition model of the environment in order to compute optimal behavior policies. Instead, it relies on online, sample-based techniques to compute (optimal) policies.

4.2.3.1 Instance based Representation

Reinforcement learning computes a solution to a problem represented as a Markov decision process (MDP) [136]. In this formulation, every state and action are represented explicitly. In the following we formally define an MDP, and discuss Q-learning—perhaps the most well-known reinforcement learning algorithm [138].

An MDP is a tuple $M =< S, A, T, R >$, where:

- $S \subseteq S'$ is the set of possible states
- A is the set of available actions

- $T : S \times A \to \Pi(S)$ is a probabilistic transition function
- $R : S \times A \times S \to \mathbb{R}$ is a reward function

A solution to this MDP is a policy π, telling the agent what action to take in a given state. Q-learning is a reinforcement learning technique [138]. It determines a policy $\pi : S \to A$ for selecting actions based on the current state. To determine this policy, the agent must learn an approximation Q^* to the optimal Q-value function $Q^* : S \times A \to \mathbb{R}$. The agent learns Q^* by performing a series of experiments, each of which reveals how much reward a particular action can obtain in a particular state. The Q-value function accumulates information about the total expected reward for an entire trial. The policy defined by the Q-value function is given by:

$$\pi(s) = \arg\max_a Q^*(s, a)$$

The Q^* function is continuously updated by the agent, as it interacts with the environment and receives rewards for performing specific actions in specific states. Updating is done as follows:

$$Q^*(s_t, a_t) := (1 - \alpha)\, Q^*(s_t, a_t) + \alpha \left(r_{r+1} + \gamma \max_a Q^*(s_{t+1}, a) \right),$$

where α is the learning rate, γ is the discount factor, and r_t is the reward received at time t.

The **learning rate** α determines the extent to which newly acquired information overrides old information. A factor of $\alpha = 0$ prevents the agent from learning anything new, whereas $\alpha = 1$ makes the agent "forgetful" (i.e. it considers only the most recent information).

The **discount factor** γ determines the importance of future rewards. A factor of $\gamma = 0$ results in an "opportunistic" agent, only considering current rewards. A

discount factor approaching $\gamma = 1$ results in an agent that strives for a long-term high reward. If γ exceeds 1, the Q-value will diverge.

We refer the curious reader to [128], for a detailed review of reinforcement learning algorithms and applications.

4.2.3.2 Extending the Representation

Reinforcement learning requires that all possible states of the environment are represented explicitly. The current state of the art in MDP representation is based on propositional languages [15]. This requirement limits the applicability of reinforcement learning to real-world environments, as these environments are characterized by high-dimensional state spaces. Thus, to apply reinforcement learning techniques to real-world problems we must choose a representation which reduces the dimensionality of the state space.

In recent years many methods have been proposed to abstract or approximate parts of the problem. Some methods focus on filtering out dimensions of the state space that are not relevant. Other methods use approximations to balance the necessary computational efforts to compute a policy and the quality of that policy. Another set of methods proposes a more powerful representation of the state space which collapses large parts of the state space onto a single state. Of particular interest to us are methods that use first-order logic languages to model and solve MDPs.

The motivation for using first-order logic is that many environments can be described more naturally in terms of objects and relations among these objects [74, 136]. The first example of using first-order logic to model and solve an MDP is the work of Džeroski et al. [41]. The use of relational representations in reinforcement learning offers many advantages. One important advantage is generalization across objects and the ability to transfer learned knowledge to different tasks in similar environments. Moreover, the use of relational representations facilitates the use of back-

ground knowledge. Partial policies, for example, are naturally provided in a logical form. Also complex predicates are naturally defined using first order logic; these predicates can induce abstractions over states and value functions, effectively reducing the dimensionality of the associated state space.

In chapter 11 we develop a relational reinforcement learning algorithm. This algorithm enables a robot to efficiently manipulate novel articulated objects. We will define this learning problem using relational representation. To that end, we will extend our definition above of an MDP into an RMDP (Relational MDP), and modify the Q-learning algorithm above to handle relational state and action representation. We will see that the problem of efficiently manipulating articulated objects yields itself naturally to a relational representation, as joints are relations between links, and links are an abstraction of the large underlying visual state space. Because this relational representation is task-specific, it reduces the dimensionality of the state space. This supports our hypotheses that the right representation has to be task-specific and that the right decomposition significantly reduces the complexity of the problem. In our case, the decomposition is into the interactive perceptual skill and the learning algorithm, which relies on the representation provided to it by the interactive perceptual skill.

CHAPTER 5

THE PROPOSED APPROACH

In chapter 3 we established that the complexity of autonomous manipulation in unstructured environments is due to the high-dimensional state space associated with it. To enable autonomous manipulation, we proposed the following set of hypotheses that, we believe, provide robots with the means to reveal task-relevant structure, which in turns renders the complex state space more manageable.

1. The symbolic representation of the environment must be task-specific
2. Symbols must be grounded in the robot's sensorimotor interactions with the environment
3. Within a task-hierarchy, simple subtasks enable more complex tasks by "reshaping" the state space

Our first hypothesis suggests a representation that is task-specific. This representation reduces the state space to only those aspects that are relevant to the task. Simply put, we represent everything we need to consider in order to accomplish the task, and nothing more. The second hypothesis requires that the symbolic representation relates to the real-world through the robot's sensors and actuators. This guarantees that the result of symbolic manipulation (reasoning, planning, etc.) remains relevant in the real-world. Finally, our third hypothesis advocates a task-hierarchy. The decomposition of a difficult problem into subproblems results in smaller problems that are easier to solve (i.e. span a smaller state space than the original problem).

This thesis proposes to use the above three hypotheses as the foundation of our approach towards autonomous manipulation. Its main focus is the important problem of manipulating articulated objects. To autonomously manipulate an articulated object, a robot must be able to deliberately change the object's extrinsic and intrinsic degrees of freedom. To that end, the robot must be able to acquire information about the object's shape and its kinematic structure. With that information, it becomes possible to plan a sequence of interactions to achieve a given manipulation objective.

The manipulation of articulated objects is a subproblem of autonomous manipulation. It is an important prerequisite for most manipulation tasks, as most real-world objects are rigid and possess degrees of freedom. And, similarly to the general problem of manipulating any object, it is associated with a very high-dimensional state space. An approach for manipulating articulated objects will advance the state of the art in autonomous manipulation. More importantly, a successful application of our hypotheses to this problem will reinforce our belief that these hypotheses also apply to the general problem of autonomous manipulation.

In the remainder of the thesis, we consider manipulation to be the problem of deliberately changing the configuration of articulated objects. We propose an approach that leverages our three hypotheses to enable robots to autonomously manipulate such objects. Our approach provides robots with the ability to model the shape and kinematic structure of rigid articulated objects. Moreover, it provides robots with the means to acquire expertise in interacting with the environment. As a result, robots could learn what aspects of the state space are most relevant for modeling, what action is likely to provide the most information about an object, and what is the most efficient way to acquire the necessary information about an object in order to manipulate it into a desired configuration.

The proposed approach enables a robot to interact with a novel object in its environment, while continuously improving its understanding of the object's shape and

kinematic structure. The success of this approach will be measured by the quality of the acquired models. A good object model is one that enables the robot to predict the outcome of its interactions. With such a model, the robot can deliberately change the intrinsic and extrinsic degrees of freedom of an object, thus facilitating autonomous manipulation.

What does a robot need to know in order to manipulate an articulated object? First, it needs to know what are the parts (or: links, rigid bodies) of the object. Then, it needs to determine the kinematic relationship between these parts. To determine this information, our robot will interact with the environment. It will cause relative motion that reveals that structure of the object. Thus, our robot also requires the ability to determine how to interact with its environment in an efficient way. Indeed, our approach for manipulating articulated objects decomposes the problem into three subproblems: rigid body modeling, kinematic structure modeling, and planning.

The three components of our solution are illustrated in figure 5.1. The first component is rigid body modeling. This component divides the world into rigid bodies, and provides the robot with the outline of each body. The second component is kinematic structure modeling. This component computes the kinematic relationship between the various rigid bodies. Rigid body modeling and kinematic structure modeling are closely tied together. The shape and motion of the parts of an object effect the way its kinematic structure is modeled, and vice versa. Finally, the third component is a planner. The planner determines how the robot should interact with the environment in order to acquire information about the shape and structure of objects. It also determines the interactions that will lead to changing the configuration of a given object as to achieve a given task.

In the following chapters we will present in more detail the three components of our proposed approach for autonomous manipulation. In chapter 6 we present our solution to the rigid body and kinematic structure modeling components, for

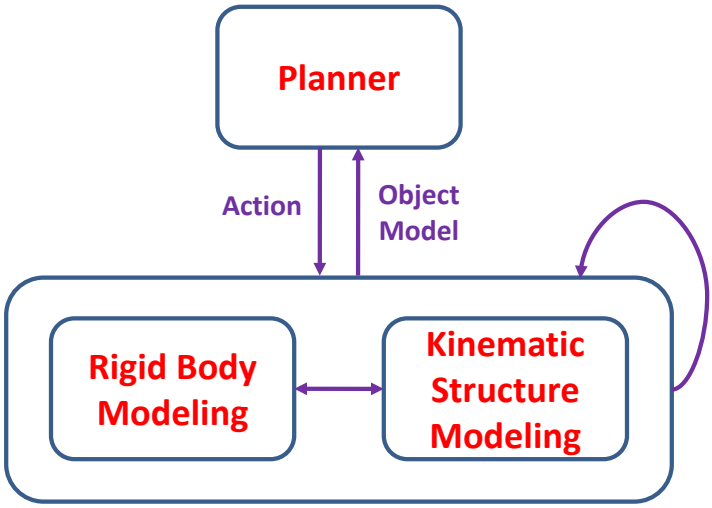


Figure 5.1. The proposed perceptual approach (conceptual level). Our approach for manipulating rigid articulated objects is composed of 3 components: Rigid Body Modeling and Kinematic Structure Modeling components that continuously improve the model of each object, and a Planning component that directs the robot's interaction with the environment.

the simple case of planar objects. Then, in chapters 7, 8, 9, and 10, we present our solution to the general problem of modeling the shape, motion, and kinematic structure of 3-D objects. Finally, in chapter 11 we present the third component of the proposed approach—the planner.

CHAPTER 6

MODELING PLANAR ARTICULATED OBJECTS

Articulated objects are abundant in our everyday environments. This category of objects includes many tools such as scissors, pliers, but also door handles, drawers, light switches, boxes etc. A robot able to manipulate these objects in an unstructured environment would enable a large variety of applications. To that end, we develop a perceptual skill, enabling a robot to acquire the necessary information for manipulating articulated objects. In this chapter, we restrict ourselves to the case of planar articulated objects, as the ones shown in Figure 6.1. In later chapters, we will remove this restriction and develop a perceptual skill for modeling three-dimensional articulated objects.

To reflect the fact that the robot operates in an unstructured environment, our perceptual skill is initialized with no specific knowledge about the object to be manipulated. The robot has to autonomously acquire information about the object's kinematic structure. Based on this information, the robot can then manipulate the object into a given configuration.

The ability to manipulate articulated objects is essential for a wide range of manipulation tasks (all prehensile manipulation tasks with rigid objects). We therefore believe that the perceptual skill presented in this chapter will serve as a sensorimotor foundation for more complex manipulation tasks. In other words, this skill represents a good decomposition of manipulation in unstructured environments.

Figure 6.1. Examples of planar kinematic structures. Left: scissors with a single revolute joint. Right: a wooden toy with a prismatic joint and two revolute joints.

6.1 Interactive Perception

Successful manipulation of articulated objects must be informed by knowledge of the object's kinematics. At a first glance, it seems plausible that this knowledge can be acquired by visual inspection of the object of interest. However, the most fundamental information for determining the kinematic structure of an object is the relative motion between the object and its environment, as well as the motion between the parts of the object. Without observing this motion—assuming no prior knowledge—the kinematic structure of the object cannot be determined. Thus, to perceive the kinematic structure of an object, we must see it in motion.

We leverage this simple insight about the importance of motion: rather than passively observing a scene, our robot manipulates the environment specifically to assist the perceptual process (see Figure 6.2). Perception, in turn, provides information necessary to manipulate successfully. The robot is watching itself as it manipulates the environment and interpreting the sensor stream in the context of this deliberate interaction.

Coupling perception and manipulation has two important advantages: First, perception can be directed at those aspects of the sensor stream that are relevant in the context of a specific manipulation task. Second, physical interactions can reveal properties of the environment that would otherwise remain hidden to sensors (see

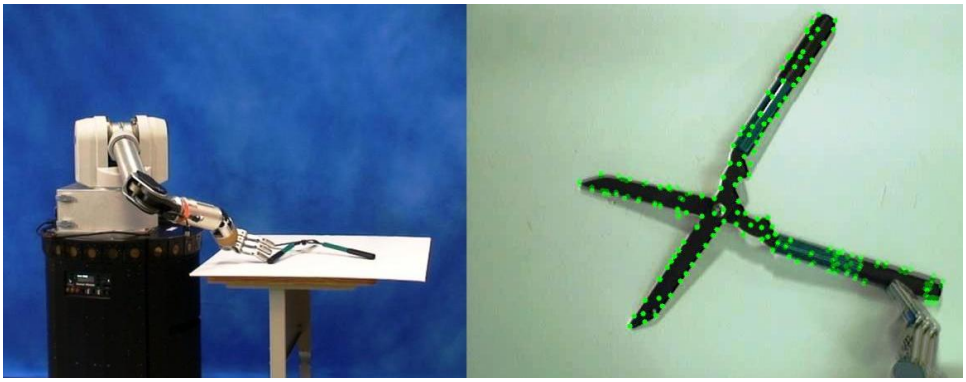


Figure 6.2. Interactive perception for planar objects. UMan (UMass Mobile Manipulator) performs a manipulation task without prior knowledge about the manipulated object. The right image shows the scene as seen by the robot through an overhead camera; dots mark tracked visual features.

Figure 6.3). In our case, by interacting with objects, their kinematic and dynamic properties become perceivable. This approach is based on our hypotheses that the robot's symbolic representation must be task-specific and grounded in the robot's sensorimotor interactions with the environment. Furthermore, interactive perception offers a new perspective on how to decompose solutions to perception problems. Instead of separating the solution into a vision step and a manipulation step (i.e. following the classical sense-plan-act paradigm discussed in chapter 3) it mixes the two according to the task.

6.2 Rigid Body Modeling

Our goal is to support manipulation in unstructured environments with a perceptual capability to continuously detect, segment, track, and model the kinematic structure of objects. To achieve this, our method relies on the insight that manipulation itself can facilitate the acquisition of perceptual information (*interactive perception*). By physically causing objects to move, the robot can generate a perceptual signal that enables object detection, segmentation, tracking, and modeling.

Figure 6.3. Pliers vs. picture of pliers. Left: a pair of pliers as seen through the robot's camera Right: a printed photo of the pliers. Interaction is necessary to distinguish between the two objects.

Our algorithm is composed of three components. The first component collects perceptual information that provides the input to our perceptual skill. This component tracks visual point features throughout the robot's interactions with the environment. The second component analyzes the trajectories of these features to formulate hypotheses about the presence of rigid bodies in the scene. The result is a clustering of features and their trajectories, where each cluster corresponds to a presumed rigid body. The third component of our algorithm uses the feature trajectories associated with a pair of rigid bodies to determine their kinematic relationship.

6.2.1 Collecting Perceptual Evidence for Segmentation

The first component of the algorithm collects perceptual information. The robot observes its interactions with the environment by tracking a large number of point features using the optical flow-based tracker proposed by Lucas and Kanade [85, 139]. During its interaction, the robot records the features' image coordinates (u, v) for each time t in feature observations $f_i(t) = \{u, v\}$.

Feature tracking is a simple and computationally efficient operation. It only requires that the scene contains sufficient texture to support visual tracking. It makes

no assumption about the shape, size, or color of objects, about their motion, or the motion of the camera. Unfortunately, feature tracking in unstructured scenes is highly unreliable. Features can jump between image regions, are lost, swapped, or drift along edges in the image. The remainder of the algorithm will automatically eliminate this noisy data, rendering the algorithm suitable for manipulation in unstructured environments.

To cause motion of objects in the scene, the robot must interact with the environment. In this chapter, we will assume that this initial interaction is given or generated by random, force-controlled motion. In later chapters, we will eliminate this assumption, and show that a robot can learn to generate such goal-directed interactions autonomously. We note that our algorithm does not differentiate between objects that move by themselves, objects moved by the robot, or objects moved by a human or robot teacher.

6.2.2 Obtaining Rigid Body Hypotheses

The second part of the algorithm uses the obtained feature trajectories to determine hypotheses about groups of features that belong to the same rigid body. To formulate these hypotheses, the algorithm leverages the simple insight that the relative distance between two points on a rigid body does not change as the body is pushed, rotated and translated. However, the distance between points on different rigid bodies does change as the bodies rotate and translate relative to each other (see Figure 6.4). Consequently, interacting with an object while observing changes in relative distance between points on the object will uncover clusters of points, where each cluster represents a different rigid body.

The first step of our algorithm serves to identify all rigid bodies observed in the scene. To achieve this, we build a graph $G(V, E)$ from the feature trajectories $f_i(t)$ obtained throughout the interaction. Every vertex $v \in V$ in the graph represents a

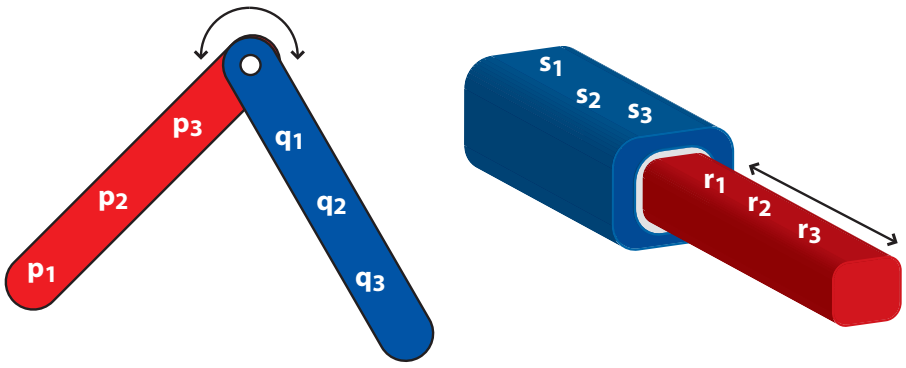


Figure 6.4. Degrees of freedom of planar objects. Left: revolute. Right: prismatic. Points p_i, q_i, r_i, s_i on each rigid link do not change relative distances.

tracked image feature. An edge $e \in E$ connects vertices (v_i, v_j) if and only if the distance between the corresponding features remains constant within some threshold throughout the observed interaction. Features on the same rigid body are expected to maintain approximately constant distance between them. In the resulting graph, all features on a single rigid body form a highly connected sub-graph. Identifying the highly connected sub-graphs is analogous to identifying the scene's different rigid bodies (see Figure 6.5).

To separate the graph into highly connected sub-graphs we use the min-cut algorithm, which separates a graph into two sub-graphs by removing as few edges as possible. Appendix B elaborates on the min-cut max-flow algorithm. Min-cut can be invoked recursively to handle graphs with more than two highly connected sub-graphs [63]. The recursion terminates when breaking a graph into two sub-graphs requires removing more than half of its edges.

Our min-cut algorithm has worst case complexity of $\mathcal{O}(nm)$, where n represents the number of nodes in the graph and m represents the number of clusters [88]. Most objects possess only few joints, making $m \ll n$. We can therefore conclude that, for practical purposes, clustering is linear in the number of tracked features.

Figure 6.5. The highly connected subgraphs and rigid bodies analogy. The scissors is composed of two rigid parts. Identifying the highly connected sub-graphs (yellow and black) is analogous to identifying the object's different rigid bodies. The intersection between the two clusters indicates the location of the revolute joint (red dot).

This procedure of identifying rigid bodies is robust to the noise present in the feature trajectories. Unreliable features randomly change their relative distance to other features. This behavior places such features in small clusters, most often of size one. In our algorithm, we discard connected components with three of fewer features. The remaining connected components consist of features that were tracked reliably throughout the entire interaction. Each of these components corresponds to either the background or to a rigid body in the scene. It now remains to understand the kinematic constraints between the different rigid bodies.

6.3 Kinematic Structure Modeling

We consider rigid bodies that are connected via a single degree of freedom joint: revolute, prismatic, rigidly connected or disconnected. Observing the relative motion that two rigid bodies undergo reveals the type of joint that connects them. Two bodies that are connected by a revolute joint share an axis of rotation. Two bodies that are connected by a prismatic joint can only translate with respect to one another. We examine all pairs of rigid bodies identified in the previous step of our algorithm to identify by which of the two joint types they are connected. If two rigid bodies experience relative motion that cannot be explained by either joint type, we infer that the bodies are not connected and belong to different articulated objects. This for example, is the case of the background, which we regard as another object.

For an object composed of k rigid bodies, there are $\binom{k}{2}$ pairs to analyze. In practice k is very small, as most objects possess only few joints.

To find revolute joints, we exploit the information captured in the graph G. We leverage the fact that vertices that belong to two different highly connected subgraphs represent features that must have maintained a constant distance to two different rigid bodies. These vertices (features) must be on or near the revolute joint connecting the two bodies. Thus, in order to find all revolute joints in a scene, we search the entire graph for vertices that belong to two clusters (see Figure 6.6).

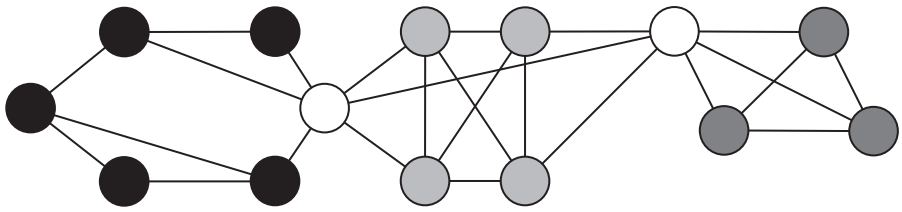
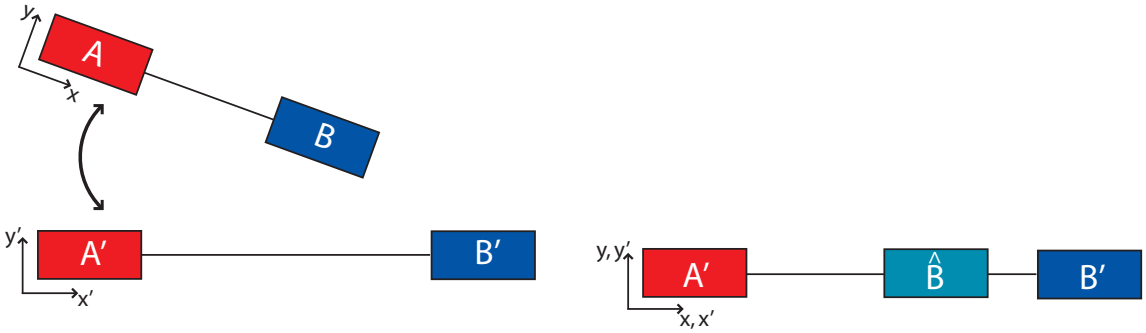


Figure 6.6. Graph representation for an object with two revolute degrees of freedom. Highly-connected components (shades of gray) represent the rigid bodies that make up the links of the object. Vertices of the graph that are part of two highly connected components represent feature that lie on or near a revolute joint (white).

To determine if two rigid bodies are connected by a prismatic joint, we exploit information from two different time instances. For every pair of bodies we examine the feature trajectories to determine the time steps when the two bodies experience their minimal and maximal relative displacement. Those instances will provide the strongest evidence and result in maximum robustness. We denote the corresponding positions of the two bodies by A, B and by A', B' at the beginning and the end of the motion, respectively (see Figure 6.7(a)). If the two bodies are connected by a prismatic joint, they may have rotated and translated together, as well as translated with respect to each other.

To confirm the presence of a prismatic joint, we compute the transformation $\mathbf{T}$ that maps features from A to A': $A' = \mathbf{T} \cdot A$. We then apply the same transformation to the second body to get its expected position, $\hat{B}$, at the second time instance (see Figure 6.7(b)). If $\hat{B} = B'$ we have no evidence for a prismatic joint and in fact A and B at this point appear to be the same rigid body. If $\hat{B}$ is different than the observed position of B', and the displacement between B' and $\hat{B}$ is a pure translation, we conclude that the two bodies are connected by a prismatic joint. If neither prismatic nor revolute joint was detected, the two bodies must be disconnected.



(a) Motion of two bodies connected by a prismatic joint

(b) Predicted pose $\hat{B}$ of body B vs. its actual pose B'

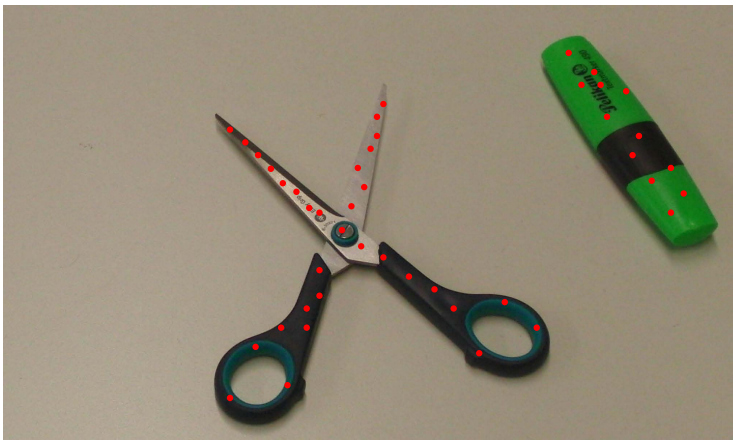
Figure 6.7. Identifying planar prismatic joints. Based on the transformation between A and A', we anticipate B's new position: $\hat{B}$. The translation between B' and $\hat{B}$ can be explained by a prismatic joint.

After all pairs of rigid bodies represented in the graph have been considered, our algorithm has categorized their observed relative motions into prismatic joints and revolute joints, or it has determined that two bodies are not connected. The background, for example, should fall into the latter category. Using this information we build a kinematic model of the object, represented by a set of Denavit-Hartenberg [31] parameters.

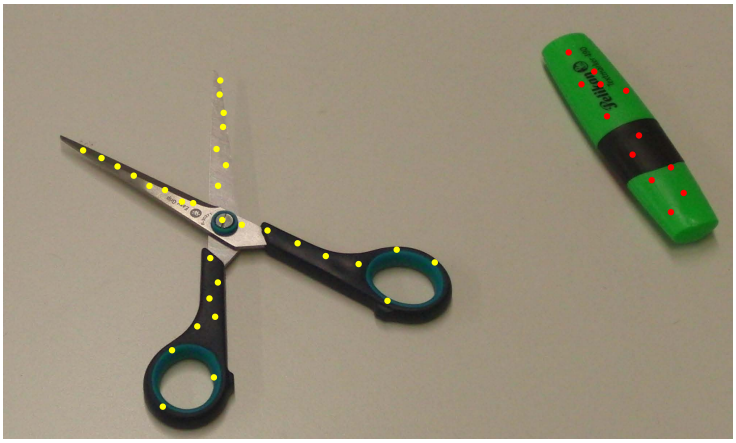
It is possible that two objects coincidentally perform a relative motion that would be indicative of a prismatic or revolute joint between them. Our algorithm would detect a joint between those bodies, as it has no evidence to the contrary. However, additional interactions with the objects will eventually provide evidence for removing such joints. Figure 6.8 shows a sequence of interactions. The robot's understanding of the scene changes with each interaction.

Note that our algorithm makes no assumptions about the kinematic structure of objects in the scene. It simply examines the relative motion of pairs of bodies. As a result, the algorithm naturally applies to planar serial chains as well as to planar branching mechanisms. Furthermore, our algorithm does not make any assumptions about the number of articulated or rigid bodies in the scene. As long as a sufficient number of features are tracked and as long as the interaction has resulted in sufficient relative motion, our algorithm will identify the correct kinematic model.

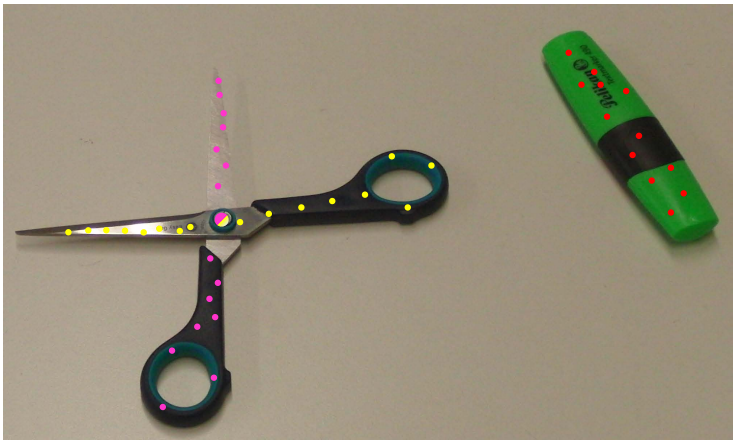
Once the kinematic structure of an articulated object has been identified, planning its manipulation becomes a trivial task. We define the manipulation task as moving the articulated body to a particular configuration q. Based on the kinematic model, we can determine the displacements of the joints required to achieve this configuration. During the interaction, we can track and update the kinematic model until the desired configuration is attained.



(a) No relative motion observed. All visual features are clustered as one rigid body (red dots).



(b) Relative motion observed between scissors and marker. Visual features are now clustered into two rigid bodies (yellow and red dots).



(c) Relative motion observed between the scissors's parts. Features are finally clustered into three rigid bodies (red, yellow and pink dots). The revolute joint is indicated by a larger dot.

Figure 6.8. Three steps of interactive perception. The three rigid bodies in the scene (scissors and marker) are identified through a sequence of interactions.

6.4 Experimental Results

We validate the proposed perceptual skill in real-world experiments. In these experiments, a robot interacts with various planar articulated objects in order to extract their kinematic structure. The resulting kinematic model is then used to perform purposeful manipulation.

Experiments were conducted with our mobile manipulator UMan (see Figure 6.9). UMan consists of a holonomic mobile base with three planar degrees of freedom, a seven degree-of-freedom Whole Arm Manipulator (WAM) by Barrett Technologies, and a three-fingered (4-DOF) Barrett hand. The robot interacts with various planar articulated objects placed on a table in front of it (see Figure 6.2). The tabletop has a wood-like texture. In some of our experiments, we have placed a white table top to provide a plain background. However, we later determined that this is not necessary and does not affect the robustness of the proposed algorithm. An overhead off-the-shelf web camera with a resolution of 640 by 480 pixels provides a video stream of the scene. The camera could have also been mounted directly on the robot, as long as it provides a good, roughly downwards view of the table. The camera mount is uncalibrated, but we ensure that the table was within the field of view, and approximately perpendicular to the camera frame. The experiments were performed next to a window, thus lighting conditions vary significantly between experiments.

In the beginning of each experiment, the robot detects 500 LK features in the scene. It then tracks the position of these features throughout the experiment. An experiment consists of a sequence of interactions. Those interactions were performed using a pre-recorded motion of the manipulator. During the interaction, about half of the tracked features are lost. The lost features are discarded even before the graph representation is constructed. Among the remaining ones, about half are very noisy and unreliable; they are discarded by our algorithm (see explanation in section 6.2.2). Lost or noisy features are usually the consequence of the manipulator's own motion

Figure 6.9. UMan (UMass Mobile Manipulator)

in the image during the interaction with the object. Shadows, reflections and lens distortion also result in unreliable features.

The results of interacting with 4 objects: scissors, shears, pliers, and a stapler are shown in Figure 6.10. All four objects possess one revolute joint. The objects were placed on top of a white table top. Experiments were repeated at least 30 times with each object. In each experiment, the target object was placed in a different initial pose. In every single one of the experiments, the algorithm was able to successfully identify the kinematic structure of the object. The positions of the joints were also accurately detected (revolute joints are marked by green dots).

Once the kinematic model of an object has been acquired, it can be used to plan and execute purposeful interactions with the object. In our experiments, we tasked UMan with forming a 90° angle between the links of the four objects. This task simulates tool use, since using any of the four objects would require to achieve, or alternate between, specific configurations. The bottom two rows of Figure 6.10

Figure 6.10. Experimental results for planar objects with a single revolute joint. We use interactive perception to extract the kinematic properties of different objects. The first row of images shows the four objects (scissors, shears, plier, and stapler) in their initial pose. The second row shows the final pose of the four objects after the robot has interacted with them. The third row shows the detected revolute joint (marked by a green disc). The fourth row of images shows the links of the obtained kinematic model and the manipulation plan to form a right angle between the two links of the tools. Putting the two links into a 90° angle here serves as an example of tool use. The links of the tools are shown as green lines, and the orientation of one of the links to achieve the goal configuration of the tool is marked by a red line. The last row of images shows the results of executing the manipulation plan as presented in the previous row: the two links of the tools have been arranged in a 90° angle.

illustrate the results of executing a manipulation plan based on the detected kinematic structure for the four objects. The results of all four experiments were very accurate, indicating that the extracted kinematic models can be applied to tool use.

Figure 6.11 shows the results of interacting with a wooden toy. In this set of experiments, the object was placed on top of a wooden table with a very similar texture to that of the object. The presence of an "interesting" background resulted in a more or less uniform distribution of features across the entire image. For this object, as well, we repeated experiments at least 30 times. The object's initial pose was changed between experiments. In all of our experiments, the proposed perceptual skill successfully identified the kinematic structure of the object, consisting of two revolute joints and one prismatic joint. The object was correctly separated from the background and no erroneous joints were detected. The positions of the joints were detected with accuracy (prismatic joints are marked by a green line and a green dot marks revolute joints). We note that to reveal complex kinematic structure, such as that of the wooden toy, the robot has to interact with it several times. The results of such a sequence of interactions should generate the perceptual evidence necessary for the identification of the different joints of the object.

In all of our experiments, the proposed algorithm was able to extract the kinematic structure of the target object. This robustness is achieved using a low-quality, low-resolution web-cam. The algorithm only requires small displacement of the object to reliably detect its kinematic structure. The algorithm does not require parameter tuning. The experiments were performed under uncontrolled lighting conditions, using different camera positions and orientations, and for different initial poses of the object.

The robustness and effectiveness of the proposed algorithm provides strong evidence that interactive perception is an important approach for manipulation and perception in unstructured environments. By combining very fundamental, but nor-

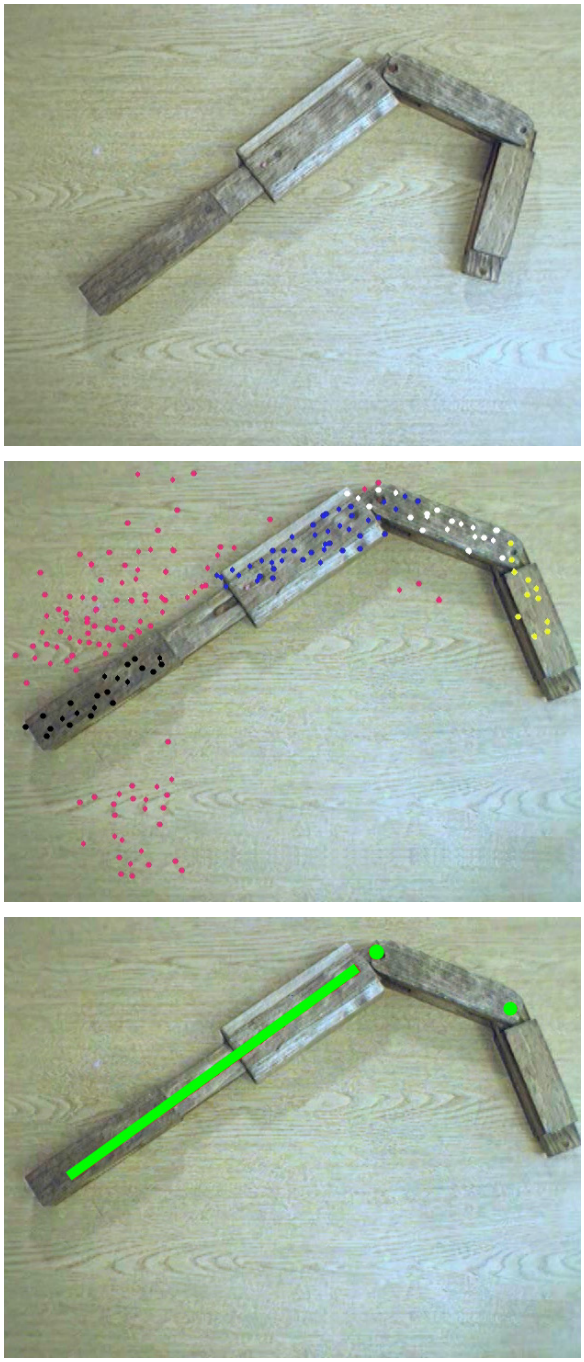


Figure 6.11. Experimental results for planar wooden toy. We use interactive perception to extract the kinematic properties of a wooden toy (length: 90cm). The top image shows the object in its initial pose. The middle image shows the object after the interaction. The detected clusters corresponding to rigid bodies are displayed. The bottom image shows the detected kinematic structure (green line marks the prismatic joint, green dots mark the revolute joints).

mally separate, capabilities from perception and manipulation, the proposed perceptual skill demonstrates a level of robustness that could not have been achieved by manipulation or vision alone.

6.5 Summary

In dynamic and unstructured environments, robots cannot rely on accurate a priori models and are generally unable to acquire such models on the fly. As a result, the reliance on these models renders manipulation and perception skills brittle and unreliable in these environments, even if they work well under highly controlled conditions.

The work presented in this chapter shows that it is possible to achieve robustness in perception and manipulation, even in unstructured environments, when perception and manipulation are coupled in a task-specific manner. By monitoring its own deliberate interaction with the environment, a robot is able to improve perception by revealing information about the environment that would otherwise remain hidden. Coupling manipulation and perception also enables the robot to interpret its sensor stream, taking into account the specific task and the known, deliberate interaction. This significantly improves the robustness of perception, as the task and the interaction constrain the space of consistent interpretations of the perceptual data. We refer to this approach of coupling perception and manipulation as interactive perception.

Interactive perception follows our third hypothesis: it is a task-specific decomposition of the general problem of autonomous manipulation in unstructured environments. It does not follow the traditional sense-plan-act paradigm. Instead, informed by the task, it crosses the boundaries between these components. The result is an elementary sensorimotor skill; a skill that can serve as a building block for more advanced manipulation capabilities.

Furthermore, this decomposition directly addresses our second hypothesis that the robot's representation must be grounded in its sensorimotor interactions with the environment. The resulting solution also follows our first hypothesis: it relies on a task-specific symbolic representation of the environment as a set of links, joints, and the kinematic relationship connecting them to each other. This is the information necessary for manipulating articulated objects. In chapter 11, we will show that this representation is well-suited also for the acquisition and generalization of manipulation knowledge.

The proposed interactive perceptual algorithm extracts **planar** kinematic models from unknown objects. The robot pushes on an object in its field of view while observing the objects motion. From this observation, the robot can compute the objects kinematic model. This model is then used to perform purposeful manipulation. This algorithm does not require prior knowledge of the object, is insensitive to lighting, texture, color, specularities, and is computationally highly efficient. It is thus ideally suited as a perception and manipulation skill for unstructured environments. We consider interactive perception, the conceptual basis behind this algorithm, as one of the important contributions in this thesis. In the next chapter, drawing on the same principles presented in this chapter, we extend the proposed interactive perceptual skill to address the manipulation of **three-dimensional** objects.

CHAPTER 7

IDENTIFYING THREE-DIMENSIONAL RIGID BODIES

The ability to identify, track, and model the shape and kinematic structure of articulated objects is a prerequisite for autonomous manipulation, as all rigid objects possess either intrinsic or extrinsic degrees of freedom. Our goal is to develop a perceptual skill that enables a robot to acquire the necessary information for manipulating three-dimensional rigid articulated objects. To realize this perceptual skill, we assume that a robot has the following capabilities: it is able to interact with the world and it is able to visually perceive its interactions. We will exclusively rely on visual feedback. Using a video stream, the perceptual skill is to infer a three-dimensional kinematic model of the object and a geometric description of its links.

In chapter 6 we limited our discussion to planar objects. To distinguish between the rigid parts of a planar object, we leveraged the fact that the real motion of an object and its observed motion (projected onto the camera plane) are the same, up to a scaling factor. The general three-dimensional case, which we tackle in the following three chapters, is significantly more challenging. The difficulty arises because we need to extract three-dimensional shape information and the kinematic structure of an object purely based on the trajectories of visual features in the image plane of the camera. We restrict the discussion to objects consisting of rigid components connected by revolute and prismatic joints.

The perceptual skill for three-dimensional objects renders the one we proposed for planar objects obsolete. Nevertheless, many of the principles that were discovered

during its development also apply to the general three-dimensional case. We will highlight the similarities and differences throughout our discussion.

7.1 Algorithm Overview

We believe that the concurrent estimation of object segmentation, 3-D structure, and kinematic structure of a scene with multiple moving bodies is too difficult to solve robustly in a single step. Therefore, we decompose the problem of perceiving kinematic models into three components—we call them factors (see Figure 7.1). The decomposition is chosen so that each factor can make extensive use of the structure inherent to the respective subproblem.

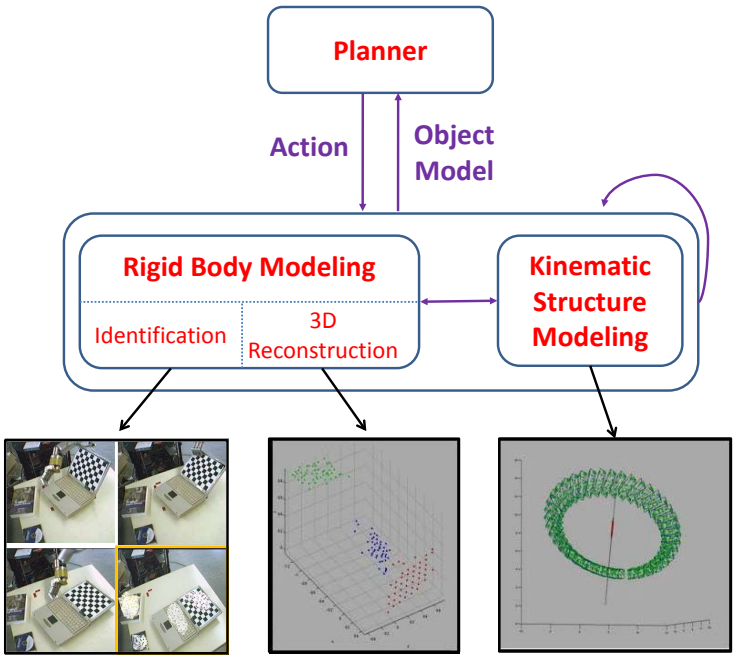


Figure 7.1. A skill for modeling three-dimensional objects. The proposed skill has 3 components (factors); identifying rigid bodies (this chapter), determining the 3-D shape and motion of rigid bodies (chapter 8), and modeling the kinematic structure of objects (chapter 9). The planner will be described in chapter 11.

Hence, this chapter is the first in a sequence of three chapters describing the proposed skill for modeling three-dimensional rigid articulated objects. In this chapter,

we discuss the first factor of our skill: the identification of rigid bodies in a scene. Chapter 8 discusses our second factor: determining the three-dimensional shape and motion of each individual rigid body. Then, in chapter 9, we present the third factor, which solves the problem of determining the kinematic constraints between these rigid bodies. Finally, following these three chapters, we conclude the discussion in chapter 10, which is dedicated to the evaluation of the proposed skill with real-world experiments.

In the following sections we describe our first factor. The task of this factor is identifying the rigid bodies in a scene. This factor is composed of three steps. The first step collects perceptual information. In this step, we track visual point features throughout the robot's interactions with the environment. The second step considers various types of information about these features to formulate hypotheses about the presence of rigid bodies in the scene. Finally, in the third step, we use the set of hypotheses to compute an association between point features and rigid bodies.

7.2 Step I: Collecting Perceptual Evidence

To identify the rigid bodies in a scene, we begin by collecting perceptual information. This process is similar to the one described for the planar skill (see section 6.2.1). Although the type of target objects changes from planar to general 3-D, the input to both algorithms is identical. The robot observes its interactions with the environment by tracking a large number of point features using an LK optical flow feature tracker [85, 139]. During its interaction, the robot records the features' image coordinates (u, v) and their color values c for each time t in feature observations $f_i(t) = \{u_i, v_i, c_i\}$.

Feature tracking is simple and computationally efficient. It requires only sufficient texture in the scene. It makes no assumption about the shape, size, or color of objects, about their motion, or the motion of the camera. Feature tracking is, however, highly

unreliable in unstructured scenes; features frequently jump across image regions, are lost, swapped, or drift along edges in the image. To be suitable for manipulation in unstructured environments, our algorithm must automatically reject this noisy data.

Similar to the process described in section 6.2.1, we rely on the robot's interaction with the environment to cause the motion of objects. Furthermore, here too we assume that this interaction is either given or randomly generated. In chapter 11 we will remove this limitation by proposing a learning framework that enables a robot to gather manipulation knowledge in order to generate goal-directed interactions.

7.3 Step II: Obtaining Rigid Body Hypotheses

The second step of identifying the rigid bodies in a scene is to analyze the obtained feature trajectories $f_i(t)$. The goal of this step is to determine hypotheses about groups of features that could lie on the same rigid body. It formulates these hypotheses by leveraging the simple insight that features associated with a single rigid body share a set of spatial, temporal, and appearance properties, some of which will remain constant or evolve consistently as the object moves.

We use an undirected graph data structure $G = (V, E)$ to represent these hypotheses. A vertex $v_i \in V$ corresponds to a feature f_i and contains the feature observations $f_i(t)$. The weight $w_{i,j} \in [0, 1]$ of an edge $e_{i,j} \in E$ connecting vertices v_i and v_j indicates the belief that the associated features, v_i and v_j, belong to the same rigid body.

To compute this belief, we employ a set of predictors, each leveraging a different source of information. A predictor $P(f_i, f_j)$ estimates the belief that two features f_i and f_j belong to the same rigid body using a specific property, such as color or change in relative distance. The weight of the edge $e_{i,j}$ is simply the product of the belief values computed by all predictors: $w_{i,j} = \prod_k P_k(f_i, f_j)$.

We now describe in detail $k = 6$ such predictors. Note, however, that it is trivial to extend our algorithms by adding more predictors.

7.3.1 Relative Motion Predictor

The defining characteristic of a rigid body is that any two three-dimensional points on the body maintain a constant distance. When the motion of objects is limited to planar motion, the constant distance between three-dimensional points is also maintained by the corresponding projection onto the camera plane. Indeed, this principle was the main insight behind our skill for modeling planar objects. If, however, we allow arbitrary three-dimensional object motions, we can no longer exploit this principle so easily. For example, when a rigid body translates away from the camera, the distance between two points on this body decreases. When it translates towards the camera, the distance between the same two points increases. Nevertheless, it does hold in specific circumstances, i.e. when a body remains static, or moves approximately in parallel to the camera plane. Our first predictor is designed to consider exactly these cases.

The **Relative Motion** predictor determines the probability of two features f_i and f_j being on the same rigid body by evaluating their relative distance over time. If the relative distance varies little over time, i.e., if $|\max_t \delta(f_i(t), f_j(t)) - \min_t \delta(f_i(t), f_j(t))|$, where $\delta(\cdot, \cdot)$ is the distance between two features in pixels, is below a noise threshold ϵ_{RM}, we conclude that f_i and f_j are likely to belong to the same rigid body ($P_{RM}(f_i, f_j) = 1$). Otherwise, the predictor sets $P_{RM}(f_i, f_j)$ to $\frac{1}{2}$, indicating neutral belief.

$$P_{RM}(f_i, f_j) = \begin{cases} 1 & \text{if } |\max\limits_t \delta(f_i(t), f_j(t)) - \min\limits_t \delta(f_i(t), f_j(t))| \leq \epsilon_{RM} \\ \frac{1}{2} & \text{otherwise} \end{cases}$$

Figure 7.2 illustrates the behavior of the **Relative Motion** predictor. The figure shows a cabinet door in two configurations: open and closed. The motion of the door is approximately parallel to the image plane. As a result, the relative distance between pairs of tracked features (marked by pink and yellow dots) changes little over time. The predictor would therefore assign a belief value of 1 to the edges connecting most pairs of pink features as well as most pairs of yellow features. We expect it, however, to assign a belief value of $\frac{1}{2}$ to edges connecting a pink and a yellow vertex. In our experiments, we set ϵ_{RM} to 5 pixels.

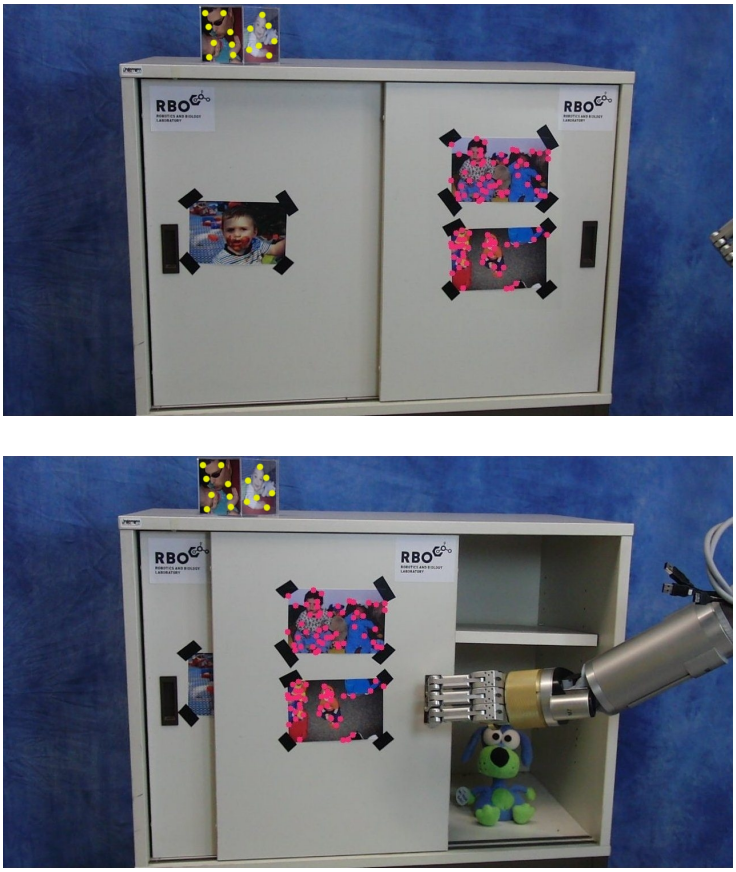


Figure 7.2. Relative Motion predictor. Features (pink and yellow dots) that experience little change in relative distance over time are clustered together.

The example in Figure 7.2 shows that the **Relative Motion** predictor can, in some cases, be enough to identify different rigid bodies in the scene due to their

different motion. In this case, it separates the door features (pink) from the picture cube features (yellow). We note, however, that this is *only* true for static features, or features undergoing motions parallel to the image plane; the existence of relative motion among features therefore does not necessarily imply that they are on different rigid bodies.

7.3.2 Short Distance Predictor

A single rigid body typically occupies a spatially contiguous region. Thus, we expect that two points on the same rigid body will be spatially co-located. This property is not generally true. The blades of a fan (Figure 7.3), for example, are part of a single rigid body. And yet, points on different blades are separated by a large distance (red dots). Nevertheless, assuming sufficient texture, there would be a sequence of features (blue dots) connecting any pair of features on the rigid body. We expect that this sequence will be composed of features that are relatively close to each other.

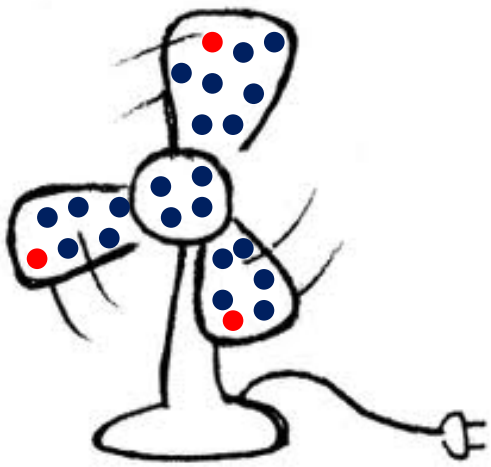


Figure 7.3. Short Distance predictor. The blades of the fan form a single rigid body. The predictor would assign a high belief value to features that are close to each other (blue dots). The propagation of connectivity would eventually also connect distant points (e.g. the two red dots).

The **Short Distance** predictor leverages this insight. It considers the position of all features in a single frame τ (typically τ is the first or last frame of the sequence). Two features are believed to be on the same rigid body if they are close to each other in this frame. The predictor computes a belief value as a function of the feature distance $\delta(f_i(\tau), f_j(\tau))$. If the distance is smaller than ϵ_{SD} pixels, it sets $P_{SD}(f_i, f_j) = 1$, indicating the belief that the two features belong to the same rigid body. Otherwise, the predictor sets $P_{SD}(f_i, f_j)$ to $\frac{1}{2}$, indicating non-conclusive evidence. In our experiments, we set ϵ_{SD} to 10 pixels:

$$P_{SD}(f_i, f_j) = \begin{cases} 1 & \text{if } \delta(f_i(\tau), f_j(\tau)) \leq \epsilon_{SD} \\ \frac{1}{2} & \text{otherwise} \end{cases}$$

7.3.3 Long Distance Predictor

The **Long Distance** predictor is similar to the **Short Distance** predictor. It leverages the same basic insight that points on the same rigid body should be spatially clustered. The **Short Distance** predictor increases the belief value of features that are close to each other. In contrast, here we take large distances evidence as an indication that features i and j belong to different rigid bodies. If $\delta(f_i(\tau), f_j(\tau))$ is larger than ϵ_{LD}^{max} pixels, we set $P_{LD}(f_i, f_j)$ to 0. If it is smaller than ϵ_{LD}^{min} pixels, we set $P_{LD}(f_i, f_j)$ to $\frac{1}{2}$, indicating no decision. Otherwise, the belief is a linear function of the distance:

$$P_{LD}(f_i, f_j) = \begin{cases} 0 & \text{if } \delta(f_i(\tau), f_j(\tau)) \geq \epsilon_{LD}^{max} \\ \frac{1}{2} - \frac{\delta(f_i(\tau), f_j(\tau)) - \epsilon_{LD}^{min}}{2 \cdot (\epsilon_{LD}^{max} - \epsilon_{LD}^{min})} & \text{if } \epsilon_{LD}^{min} \leq \delta(f_i(\tau), f_j(\tau)) \leq \epsilon_{LD}^{max} \\ \frac{1}{2} & \text{if } \delta(f_i(\tau), f_j(\tau)) \leq \epsilon_{LD}^{min} \end{cases}$$

Figure 7.4 illustrates the result of applying the short and long distance predictors. The features associated with objects on the top shelf are clustered together because

of the short distance between features. This is also the case for features on the bottom shelf. Because of the long distance predictor, features on the top shelf (pink dots) are associated with a different rigid body than features on the bottom shelf (blue dots). This clustering happens even though there has been no relative motion between between the top and bottom shelves. In fact, this clustering requires no motion at all. Here, and in all of our experiments, ϵ_{LD}^{max} is set to 160 pixels and ϵ_{LD}^{min} to 30 pixels.

Figure 7.4. Short and Long Distance predictors. Objects on the top and bottom shelves are clustered as two different rigid bodies (pink and blue dots). This assignment does not require any motion, it is solely the result of the observed distance between features.

7.3.4 Color Segmentation Predictor

Since the time of the Gestalt movement in psychology (early 20^{th} century, [46]), it has been known that perceptual grouping plays an important role in human visual perception. The above three predictors compute perceptual grouping based on distance or relative motion. The **Color Segmentation** predictor computes perceptual grouping based on color or texture, as proposed originally by proponents of the Gestalt theory. It postulates that the visual appearance of a rigid body is uniform.

The predictor uses color and texture information to segment an image into color-consistent regions (see Figure 7.5). Its implementation is based on the "Efficient Graph-Based Image Segmentation" algorithm [47] described below.

"Efficient Graph-Based Image Segmentation" [47] is an image segmentation algorithm that is useful for a wide range of computer vision tasks. To identify a region, it incrementally builds subgraphs composed of a set of pixel (vertices) interconnected by undirected edges with large weights. These weight measure the similarity between pixels (vertices). The algorithm begins by assigning each vertex (pixel) to its own region. Then, in every step of the algorithm, it merges regions based on a similarity measure. To compute the similarity between two regions, "Efficient Graph-Based Image Segmentation" relies on the following two insights: First, widely varying intensities do not, by themselves, provide evidence for multiple regions. Second, adaptive or non-local criterion must be used in order to consider the relationship between the intensity differences across the boundary of two regions and the intensity differences inside the regions.

The resulting algorithm is computationally very efficient. Its runtime is $\mathcal{O}(n \log n)$, for n image pixels and with low constant factors. Its performance can be tuned by setting 2 parameters: k and σ. Varying k changes the scale of observation: larger k creates preference for larger components. In our experiments, k was set to 100. To compensate for digitization artifacts and other noises, we adjust the value of σ. This smoothes the image with a Gaussian filter before computing the edge weights. Gaussian filtering is simply a convolution of the image with the following filter: $g(x) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{x^2}{2\sigma^2}}$. We always set the parameter of the Gaussian filter to be $\sigma = 0.8$, which does not produce any visible change to the image but helps remove artifacts.

The **Color Segmentation** predictor assumes that point features that are in the same color region are more likely to belong to the same body than points that are in neighboring regions. The more color regions separating between a pair of features

Figure 7.5. Color Segmentation predictor. The top image shows an unstructured scene. The bottom image shows its segmentation using color and texture information [47].

f_i and f_j, the weaker is the belief that they are on the same rigid body. If a pair of features are separated by $n \geq \epsilon_{CS}^{max}$ regions, we set $P_{CS}(f_i, f_j) = \frac{1}{2}$, indicating neutral belief. Otherwise, the predictor sets $P_{CS}(f_i, f_j) = 1 - \frac{n - \epsilon_{CS}^{min}}{\epsilon_{CS}^{max}}$. In our experiments, we set ϵ_{CS}^{max} to 4 and ϵ_{CS}^{min} to 2:

$$P_{CS}(f_i, f_j) = \begin{cases} \frac{1}{2} & \text{if } n \geq \epsilon_{CS}^{max} \\ 1 - \frac{n - \epsilon_{CS}^{min}}{\epsilon_{CS}^{max}} & \text{if } \epsilon_{CS}^{min} < n < \epsilon_{CS}^{max} \\ 1 & \text{if } n \leq \epsilon_{CS}^{min} \end{cases}$$

where n =number of separating color/texture regions.

7.3.5 Triangulation Predictor

The three-dimensional distance between a set of points that belong to the same rigid body remains constant regardless of the body's motion. The observation f_i of a point p_i is a projection of p_i onto the camera plane. Because motion is not necessarily in the plane, the distance between a set of features that belong to the same rigid body only rarely remains constant. Let us assume, for the purpose of this discussion, that none of the features under consideration disappear due to the body's motion (e.g. obstruction or self-obstruction). We can visualize the features as a spider-web; the changes in feature positions due to motion would compress or decompress the web in various places. However, we expect the neighborhood relationships to remain unchanged.

To capture the notion of neighborhood in a spider-web, we compute a Delaunay triangulation over the features for a single image frame. A Delaunay triangulation for a set of points in the plane is a triangulation $Tri(\Omega)$ such that no point in Ω is inside the circum-circle of any triangle in $Tri(\Omega)$. Delaunay triangulations maximize the minimum angle of all the angles of the triangles in the triangulation. This triangulation was invented by Boris Delaunay in 1934 [34]. The notion of triangulation is degenerate for the case of a set of points on the same line. For four points on the same circle, the Delaunay triangulation is not unique (e.g. for the four corners of a rectangle, there are two possible splits into triangles that satisfy the Delaunay condition that the circum-circles of all triangles have empty interiors.

The **Triangulation** predictor relies on the insight that features on the same rigid body generally maintain neighborhood relationships throughout short motions of the object. If a feature f_i is to the left of f_j at time t_1, it is expected to be to the left of that feature at time t_2. We use the Delaunay triangulation to efficiently determine this neighborhood relationship. The predictor computes the neighborhood relationship in the first frame of the sequence, and checks for violations in the current frame.

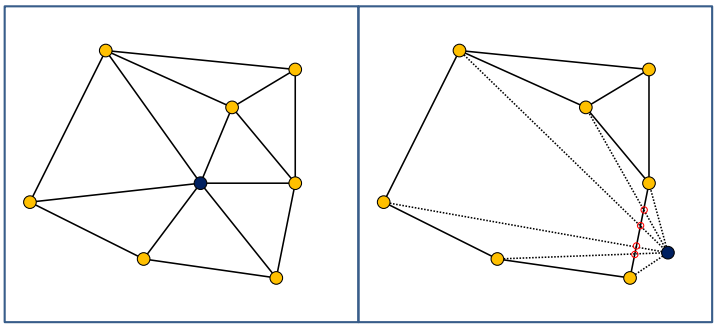


Figure 7.6. Triangulation predictor. The left image shows the Delaunay triangulation for a set of features at time t. The right image uses the adjacency relationship at time t for feature locations at time $t+1$. The blue feature is not consistent with the other features (orange), and therefore creates edge intersections.

Figure 7.6 illustrates the process of determining which features moves in a way that violates the neighborhood relationship. These features are not consistent with the single rigid body hypothesis. The left image shows the Delaunay triangulation for a set of features at time t. The right image corresponds to the adjacency relationship at time t for feature locations at time $t+1$. In this example, only one feature (blue circle) has moved, and therefore is not consistent with the other features (orange circles). The predictor detects features that are inconsistent with the single rigid body hypothesis by searching for edge intersections (red circles). It then assigns a weight of zero to edges between vertices that violate the neighborhood relationship. Edges that do not violate the neighborhood relationship are assigned weight of one:

$$P_{TR}(f_i, f_j) = \begin{cases} 0 & \text{, if neighborhood relationship is violated} \\ 1 & \text{, otherwise} \end{cases}$$

Please note that the first four predictors formulate local hypotheses. In contrast, the **Triangulation** predictor formulates a hypothesis encompassing all features—it leverages global information to determine feature-rigid body assignment. The computational complexity of this predictor is $\mathcal{O}(|V|^2)$.

7.3.6 Fundamental Matrix Predictor

The **Fundamental Matrix** leverages the following insight: the set of observations $f_1, f_2, ..., f_n$ contains subsets of features whose 3-D motion is consistent. To solve the feature-motion model association problem, we must solve a chicken-and-egg problem. On the one hand, we must identify a set of potential 3-D motion models. We can then assign features to these motion models based on how well the motion of a feature is explained by one of the models. On the other hand, to compute the motion models, we must identify a set of features whose motion is consistent.

There are two challenges that further complicate our chicken-and-egg problem. First, each observation is contaminated with noise due to digitization, lighting, and a variety of other natural phenomena. Second, the number of models to be recovered is not known a priori.

7.3.6.1 Random Sample Consensus

The RANSAC (**RAN**dom **SA**mple **C**onsensus) algorithm is a widespread solution to the problem of fitting a model to noisy data. It was first published in 1981 by Fischler and Bolles [48]. RANSAC works reliably when the data contains measurements from a single structure corrupted by gross outliers. It is an iterative method, estimating the parameters of a mathematical model from a set of observations, which contains outliers. RANSAC is a non-deterministic algorithm as it produces a reasonable result only with a certain probability. This probability increases as a function of the number of iterations (samples) allowed.

RANSAC methods usually assume that the data consists of "inlier" (data points whose distribution can be explained by some set of model parameters), and "outliers" (data points which do not agree with the model). All data can be subject to noise, which increases the complexity of distinguishing between inlier and outliers. RANSAC also assumes that, given a small set of inliers, the parameters of the model can be

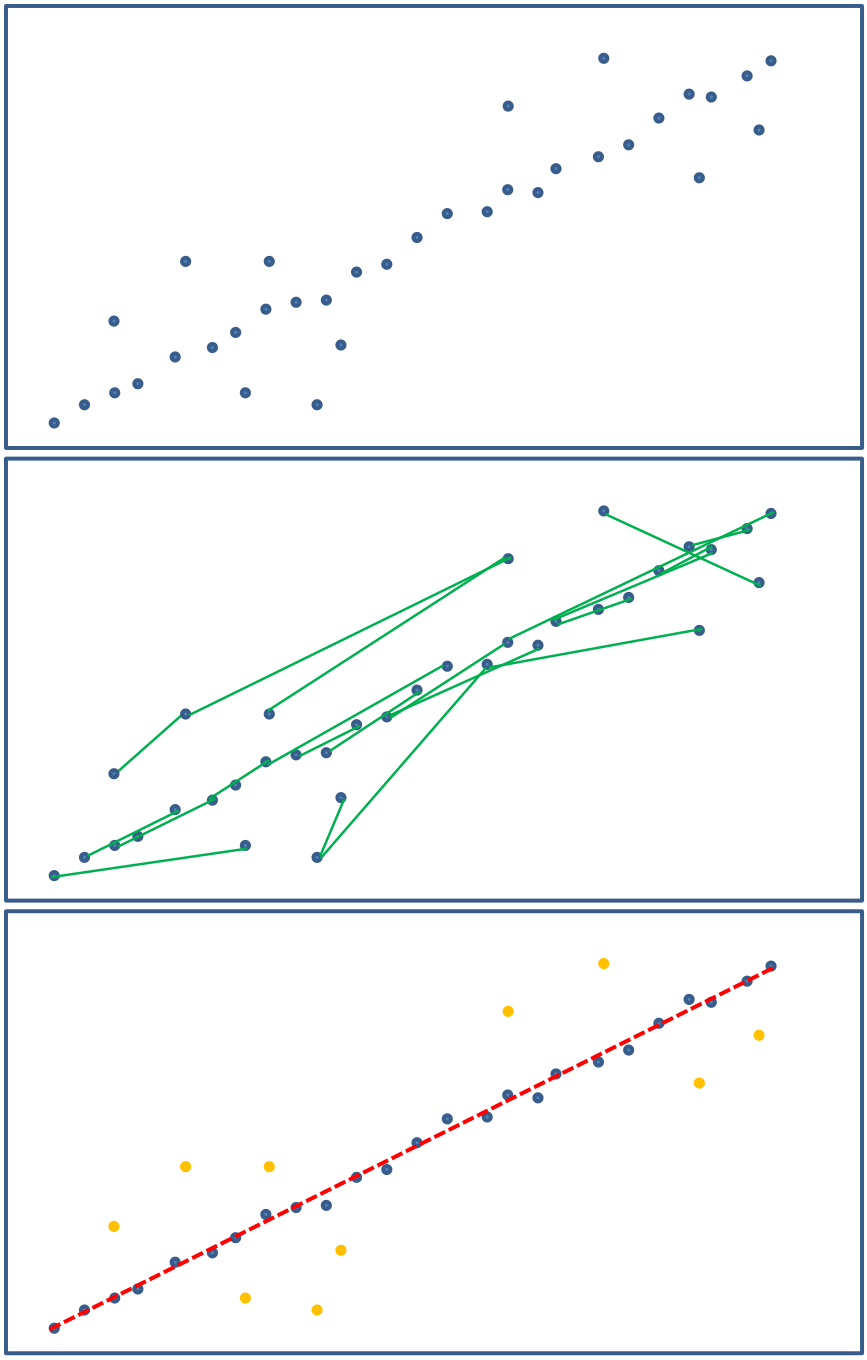


Figure 7.7. RANSAC: extracting 2D lines. RANSAC extracts a model underlying the data. In this case, it fits a line to the data. Top: A set of samples in 2D space. Middle: Random samples of 2 points are selected. We fit a line model (green line) to each pair of points. Bottom: The different potential line models are aggregated into one maximally consistent model (red line). Outliers are identified (orange circles).

estimated using a simple procedure, such that the resulting model optimally fits the data (see Figure 7.7).

The properties of RANSAC render it suitable for solving our chicken-and-egg problem of modeling the motion of a set of features while at the same time determining which features should be included in that set. However, most variants of RANSAC assume that all "inliers" are due to a single underlying model. When multiple instances of the same structure are present in the data, the problem becomes significantly more difficult, as the robust estimator must tolerate both gross outliers and pseudo-outliers. The former are due to noise, whereas the latter are defined as "outliers to the structure of interest but inliers to a different structure". Several methods have been proposed to enable the extraction of multiple models.

Sequential RANSAC, as its name suggests, applies RANSAC sequentially and removes the inliers from the data set as each model instance is detected (for example: [76]). This approach is not optimal. The multi-RANSAC algorithm [146], on the other hand, is optimal provided that the models do not intersect. However, it requires that the number of models is specified a priori. The Randomized Hough Transform (RHT) [142] builds an histogram over the parameter space. A minimal sample set is randomly selected and the parameters of the unique model that it defines are recorded in the histogram. Peaks in the histogram correspond to the sought model. Its extension to multiple models is straightforward: they are revealed as multiple peaks in the parameter space. RHT does not need to know the number of models beforehand. It suffers, however, from the typical shortcomings of Hough Transform methods, such as limited accuracy and low computational efficiency.

To leverage the robustness of RANSAC in the case of multiple structures and high percentage of gross outliers, we rely on the J-Linkage algorithm proposed in [133]. This algorithm adopts a conceptual representation: each data point is represented

with the characteristic function of the set of models preferred by that point. Multiple models are revealed as clusters in the conceptual space.

7.3.6.2 The J-Linkage Algorithm

The first step of the J-Linkage algorithm is random sampling. It generates M model hypotheses by drawing M minimal sets of data points necessary to estimate the model. This set of hypotheses is called a minimal sample set (MSS). Then, similar to RANSAC, J-Linkage computes the consensus set (CS) of each model. The CS of a model is the set of points such that their distance from the model is less than a threshold ϵ.

J-Linkage represents the consensus sets of all models using a matrix of size $N \times M$, where an entry (i, j) is 1 if point i belongs to the CS of model j, and 0 otherwise. Figure 7.8 shows an example for such a "Preference" matrix. Every column in this matrix represents the characteristic function of the consensus set of a single model hypothesis. Each row indicates the models with which point i is in consensus.

The characteristic function of the preference set of a point is a conceptual representation of that point. We expect that points belonging to the same structure will have a similar characteristic function (or: conceptual representation) because models generated with random sampling should cluster in the hypothesis space around the true models.

Minimal sample sets are constructed in a way that neighboring points are selected with higher probability. If a point p_i has already been selected, then p_j has the following probability of being drawn:

$$P(p_j|p_i) = \begin{cases} \frac{1}{Z} e^{-\frac{\|p_j - p_i\|}{\sigma^2}} & \text{, if } p_i \neq p_j \\ 0 & \text{, if } p_i = p_j \end{cases}$$

where Z is a normalization constant and σ is chosen heuristically. For a detailed discussion of the sampling process, we refer the curious reader to [133].

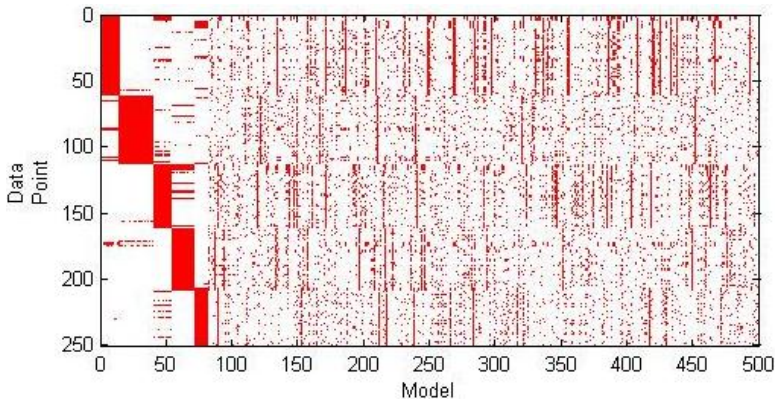


Figure 7.8. J-Linkage preference matrix. A row shows the consensus of a point. The rows are ordered by cluster. A columns represents a model, ordered by cluster size. In this example, the data fits into five large clusters. The rest of the data is considered to be noise.

Models are extracted by agglomerative clustering of data points in the conceptual space, where each point is represented by the characteristic function of its preference set. The agglomerative clustering algorithm proceeds in a bottom-up manner. It starts with all singletons; each iteration of the algorithm merges the two clusters with the smallest distance. The distance metric used for clustering has to fit the problem. In J-Linkage, the preference set of a cluster is computed as the intersection of the preference sets of its points. We then use the Jaccard distance metric between two preference sets to determine the distance between the respective two clusters.

The Jaccard distance metric measures the degree of overlap between two sets. Given two sets A and B, it is defined as: $dist_{Jaccard}(A, B) = \frac{|A \bigcup B| - |A \bigcap B|}{|A \bigcup B|}$. The Jaccard distance between two sets ranges from 0 to 1, where two identical sets have a Jaccard distance of 0, and two disjoint sets have a Jaccard distance of 1.

The clustering algorithm only links together elements whose preference sets overlap (i.e. Jaccard distance of 1). For each cluster, there exists at least one model that is in the preference set of all the points—that is, at least one model that fits all the points of the cluster. Likewise, one model cannot be in the preference set of all

points of two clusters because otherwise they would have a Jaccard distance of 1, and therefore linked together.

Please note that J-Linkage may generate more than one model fitting all points in a single cluster. In this case, these models should be very similar. The final model of a cluster is selected by least squares fitting.

7.3.6.3 Modeling Motion Using The Fundamental Matrix

Random sampling and clustering using J-Linkage provide us with a way to solve the chicken-and-egg problem of identifying motion models while at the same time assigning observations to rigid bodies. These methods are robust against noise (outliers) and can automatically determine the optimal number of clusters in the data. It remains to determine the minimal set for computing a model hypothesis, as well as a method for computing that model.

In our case, a data point is a visual point feature at two time instances, and the model hypothesis is the motion of a rigid body between these two time instances. It is not trivial to model the motion of a rigid body based on a set of feature observations. This is the case because our feature observations $f_1, f_2, ..., f_n$ are only a projection onto the camera plane of the corresponding set of 3-D points $p_1, p_2, ..., p_n$. We propose to represent a three-dimensional motion hypothesis using a fundamental matrix. Computing this matrix, as we will see, requires a minimal set of 8 features.

The fundamental matrix is the algebraic representation of epipolar geometry [62]. Given a pair of images, for each point feature p in one image, there exists a corresponding epipolar line l' in the other image. Any point feature p' in the second image matching the point feature p must lie on the epipolar line l'. The epipolar line is the projection in the second image of the ray from the point p through the camera center C of the first camera. For detailed algebraic derivation, we refer the reader to the formulation due to Xu and Zhang [141].

The fundamental matrix is a 3×3 matrix of rank 2. If the a three-dimensional point P is imaged as p_1 in frame t_1 and p_2 in frame t_2, than the fundamental matrix F satisfies the following relationship: $p_2 F p_1 = 0$. The fundamental matrix F therefore captures the relationship between the two views of the scene.

The fundamental matrix F has several important properties. First, if F explains the camera motion from frame t_1 to frame t_2, than F^T explains the camera motion in the opposite direction. Second, for any point p in the first image t_1, the epipolar line is $l' = Fp$. Similarly, $l = F^T p'$ represents the epipolar line corresponding to p' in the second image. Third, F has seven degrees of freedom. A general 3×3 matrix has 8 degrees of freedom (ignoring scale). The fundamental matrix has only 7 degrees of freedom because it satisfies the constraint: $\det F = 0$, which removes one degree of freedom.

The Normalized 8-point Algorithm is the standard method for computing the fundamental matrix. It is discussed in detail in Appendix A. To compute the fundamental matrix, this method requires at least 8 points. Hence, when we use J-Linkage to determine rigid body hypotheses, the feature set size will be set to 8 features.

If the scene is static, F indicates the motion of the camera between frame t_1 and frame t_2. If the scene is dynamic and the camera is static, the fundamental matrix indicates the motion of an observed object—assuming that all points used to determine the matrix belong to the same rigid body. Figure 7.9 shows two views of the same scene. Using eight features (yellow circles) that are matched between the images, we compute the fundamental matrix to explain the motion of the camera between frames. If the eight features are on a single rigid body, we can explain the motion regardless of whether the objects or the camera are moving, or both of them at the same time.

Figure 7.9. Fundamental Matrix predictor. Two views of the same unstructured scene. Eight features are matched across the two views (yellow circles). The fundamental matrix explains the motion of the camera between views.

7.3.6.4 Assigning Features to Rigid Bodies

The Fundamental Matrix Predictor relies on J-Linkage, random sampling and the normalized 8-point algorithm to determine an assignment of features to the rigid bodies in the scene. It also determines the number of rigid bodies and their individual motion models. The predictor uses random sampling to select sets of 8 features. Then, the normalized 8-point algorithm provides us with a fundamental matrix describing the motion of the 8 features between two views of the scene. Finally, with enough samples, J-Linkage identifies the most consistent motion models and the corresponding feature clusters.

In our experiments, we have generated about 500,000 samples of 8 points for a set of 500 feature observations per frame. This is an extremely small number of samples compared to the total number of possible selection: $\binom{500}{8} = \frac{500!}{492!8!} \approx 9 \cdot 10^{16}$. Nevertheless, computing the fundamental matrix 500,000 times is a time consuming process. In work that is outside the scope of this thesis, we have implemented and tested a version of this predictor on a GPU (Graphics Processing Unit). The results indicate the merit of parallelizing the computation of this predictor; we have seen reduction in run time of two orders of magnitude. The predictor can be further improved by considering a more intelligent sampling scheme. Currently, sampling is based on the distance between candidates, i.e. features that are close to each other are more likely to be drawn together. Future work will direct sampling by considering additional properties, such as color and texture.

The **Fundamental Matrix Predictor** sets $P_{FM}(f_i, f_j)$ to 1 if the motion of features f_i and f_j can be explained by the same fundamental matrix hypothesis, i.e. if f_i and f_j are clustered together by the same motion model between frames τ_1 and τ_2. The weight is set to $\frac{1}{2}$ if f_i and f_j are associated with different motion models:

$$P_{FM}(f_i, f_j) = \begin{cases} 1 & \text{, if } f_i \text{ and } f_j \text{ agree with the same motion model from } \tau_1 \text{ to } \tau_2 \\ \frac{1}{2} & \text{, otherwise} \end{cases}$$

Similar to the previous two predictors, here too, we utilize global information to formulate hypotheses about the motion of a group of features.

7.4 Step III: Identifying Rigid Bodies

The hypotheses computed by the six predictors are captured in our fully connected multi-graph $G = (V, E)$. The weight $w_{i,j}$ of an edge $e_{i,j} \in E$ indicates the confidence that $f_i(t)$ and $f_j(t)$ belong to the same rigid body. To compute this confidence value,

the weights provided by the different hypotheses (predictors) are multiplied together: $w_{i,j} = \prod_k P_k(f_i, f_j)$, where $P_k(f_i, f_j)$ indicates the likelihood of f_i and f_j being on the same rigid object as estimated by predictor k.

To extract a segmentation of the scene into rigid bodies from the resulting weighted graph, we first discard edges with weight zero. We expect that the remaining edges will form highly connected sub-graphs, where each sub-graph is associated with a single rigid body. To identify these highly connected sub-graphs, we rely on a recursive version of the min-cut algorithm.

The original min-cut algorithm separates a graph into two sub-graphs by removing as few edges as possible. In Appendix B we discuss in detail the min-cut max-flow problem. We note that within the context of a weighted graph, the term "few edges" refers to minimizing the sum of weights of removed edges. We invoke min-cut recursively to dissect graphs with more than two highly connected sub-graphs [63]. The recursion terminates when splitting a graph into two sub-graphs requires removing more than half of its edges.

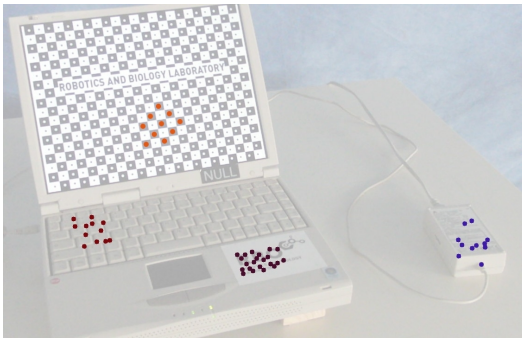
Our recursive version of min-cut has a worst case complexity of $\mathcal{O}(nm)$, where n is the number of nodes in the graph and m is the number of clusters [88]. The number of rigid bodies in an unstructured scene is typically much smaller than the number of trackable features. In most cases, therefore, $m \ll n$, and the complexity of min-cut is linear in the number of tracked features.

This procedure of identifying rigid bodies relies on a variety of visual cues. It considers distance, motion and color to formulate hypotheses about the proper segmentation of a scene into rigid bodies. It is therefore robust against sensor noise. Moreover, by fusing multiple sources of information together, no single predictor can associate or disassociate a feature from a rigid body (see Figures 7.10 and 7.11). It is the accumulation of evidence that leads to the correct decision. In this process, unreliable features are naturally placed in small clusters, most often of size one. We

can eliminate these features simply by discarding connected components with fewer than some minimum number of features. We use 8 features. The remaining connected components consist of reliable features. Each of these components corresponds to a rigid body in the scene.

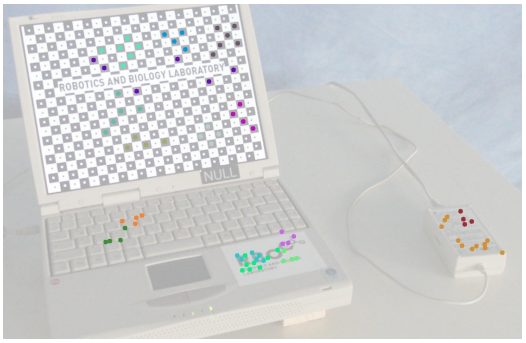
To understand the kinematic relationships in the scene, in the next chapter we turn our attention to acquiring a three-dimensional shape and motion model for each cluster (rigid-body).

(a) Relative Motion Predictor: Only static features are captured because other features move outside of the camera plane.

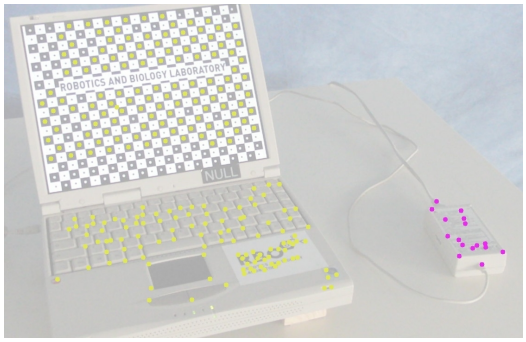


(b) Short Distance Predictor: Four small clusters are found. These clusters corresponds to regions with high density of features.

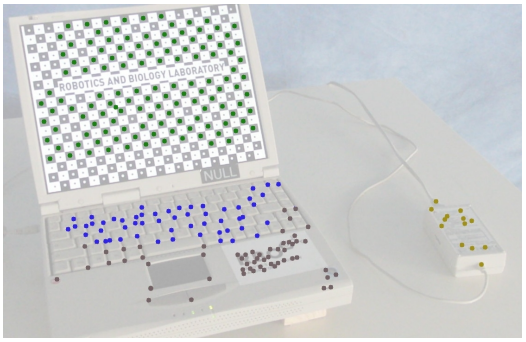
(c) Long Distance Predictor: No clusters. This predictor only eliminates edges; it does not create clusters by itself.



(d) Color Segmentation Predictor: Many small clusters; this outcome illustrates that relying on color and texture alone to identify rigid bodies is brittle.



(e) Triangulation Predictor: Two clusters detected. The relative motion between the screen and keyboard was not enough to violate the neighborhood relationship.



(f) Fundamental Matrix Predictor: Four clusters are detected. The correlation to rigid bodies is correct with the exception of the two keyboard clusters.

Figure 7.10. The contribution of individual predictors to segmentation. During the sequence, the laptop was manipulated, while the power supply remained stationary. Segmentation based on any single predictor is suboptimal.

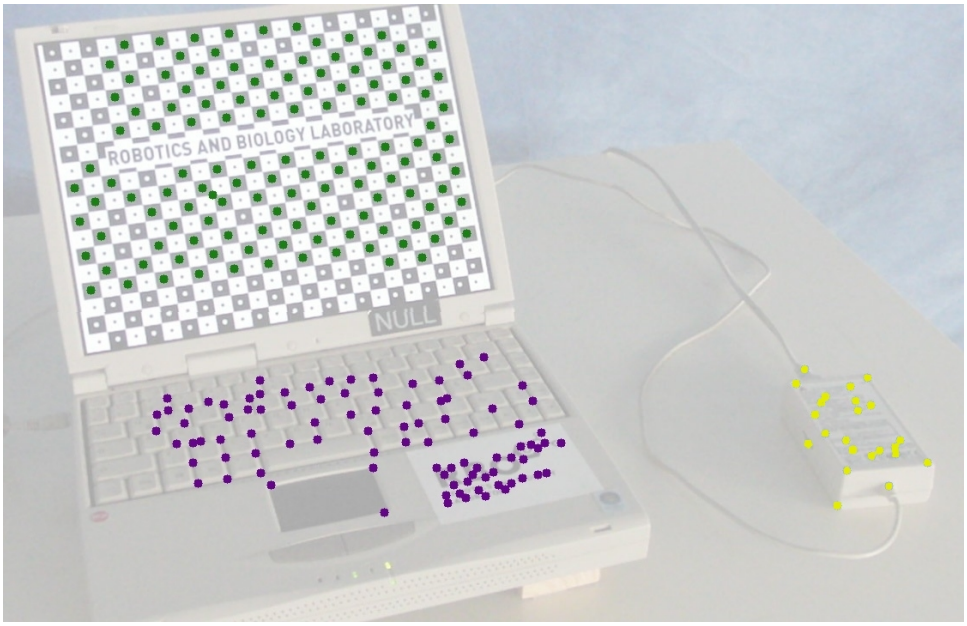


Figure 7.11. The contribution of all predictors to segmentation. When applying all 6 predictors to modeling a scene, we expect better results. Here, the three rigid bodies are correctly identified (screen, keyboard, and power supply). No single predictor achieves this result (see Figure 7.10)

CHAPTER 8

MODELING THREE-DIMENSIONAL SHAPE AND MOTION

In the previous chapter we identified 3 factors necessary to create a skill for perceiving three-dimensional articulated objects. We then developed the first factor, which identifies clusters of features which belong to the same rigid body. This chapter focuses on the second factor, extracting three-dimensional motion of a rigid body from the trajectories of the 2D features assigned to that body.

Formally, the second factor of our perceptual skill solves the following problem: Given as input a set of k feature trajectories $\{f_i(t)\}_{i=1,\ldots k}$, associated with a single rigid body, compute the three-dimensional structure and motion of the body. Here, we consider the body's structure to be the 3-D position of all features in the first frame.

In chapter 4, we reviewed the state of the art in structure from motion. The success of structure from motion has enabled a variety of applications in both the robotics and computer vision communities [21, 72, 132]. We too would like to take advantage of this success in order to reconstruct the three-dimensional shape and motion of articulated objects. The first component of our perceptual skill (described in Chapter 7), enables us to extract the set of point features associated with each rigid bodies in a scene. Thus, our goal here is to model the three-dimensional shape and motion of each one of the object's rigid parts, using the feature trajectories associated with that part.

Our instance of the problem, however, has the following two properties, which render it more challenging than the standard structure from motion case:

1. The set of features associated with a single rigid body is typically spatially condensed. This is in contrast with the standard case of structure from motion in which features are distributed across the entire image.

2. Manipulable objects often only exhibit small depth discontinuities. This is in contrast with the standard case of structure from motion in which scenes with large depth discontinuities are considered.

We cannot directly apply Bayesian filtering methods to our problem as these are sensitive to initialization [93]. Most existing filters for structure from motion initialize all features to some default depth value. These methods rely on a strong error signal to quickly converge onto the correct depth values. Convergence usually requires that features are spread over the entire visual field and have much variability in their distance to the camera. None of these preconditions applies to our instance of the problem.

We also cannot directly apply key-frame methods to our problem. Although key-frame methods not sensitive to initialization because they rely on global optimization (e.g. bundle adjustment), they do suffer from one significant drawback in the context of our perceptual skill: they reconstruct the scene and the camera's motion only at the selected key frames. To understand the kinematic relationship between the rigid parts of an object, we require their three-dimensional trajectories. In other words, we need to reconstruct the scene and the camera's motion throughout the sequence. And, applying key-frame methods to each frame of a typical sequence is simply too slow for manipulation.

We solve these problems by applying both types of methods sequentially. Our perceptual skill uses a key-frame based method to recover the 3-D structure of the scene in the first frame. It automatically selects the key-frames based on the observed feature motion. As soon as the average motion in the scene crosses some threshold, we generate a new key-frame. We then apply bundle adjustment, a global optimization

method to the key-frames. Our implementation is based on the open source package Bundler [120]. For more information about bundle adjustment, see Appendix C. With this initialization, we employ a simple extended Kalman filter (EKF) to compute the camera motion [1] throughout the entire sequence. The Bayesian filter propagates the 3-D shape from the first frame to the following frames. It can make slight corrections to the shape to achieve smoother camera (object) motions. We repeat this for each rigid body in the scene. For more information about the extended Kalman filter, see Appendix D.

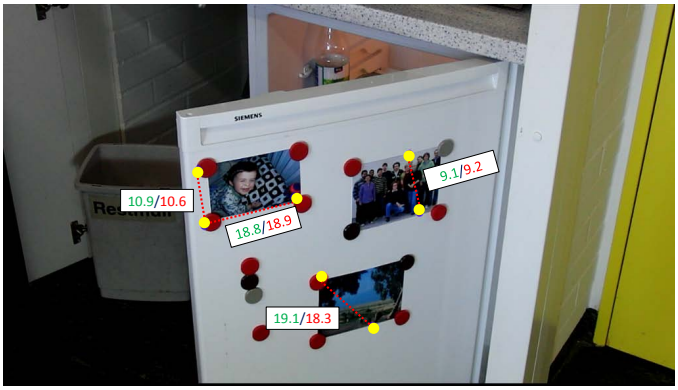


Figure 8.1. Quality of three-dimensional reconstruction. Yellow circles mark a few selected features on the fridge. Four 3-D distances are labeled by red dashed lines. Distances are showed in green (manual measurements) and red (computed by the reconstruction algorithm). They are consistent.

To assess the quality of the reconstruction, we compare a few distances as computed by our approach with manual measurements. It is difficult to accurately measure the quality of the reconstruction because it is not trivial to map a feature on the image to the corresponding point on the real object. Also, the precision of these manual measurements is not very high. Figure 8.1 compares the manual measure-

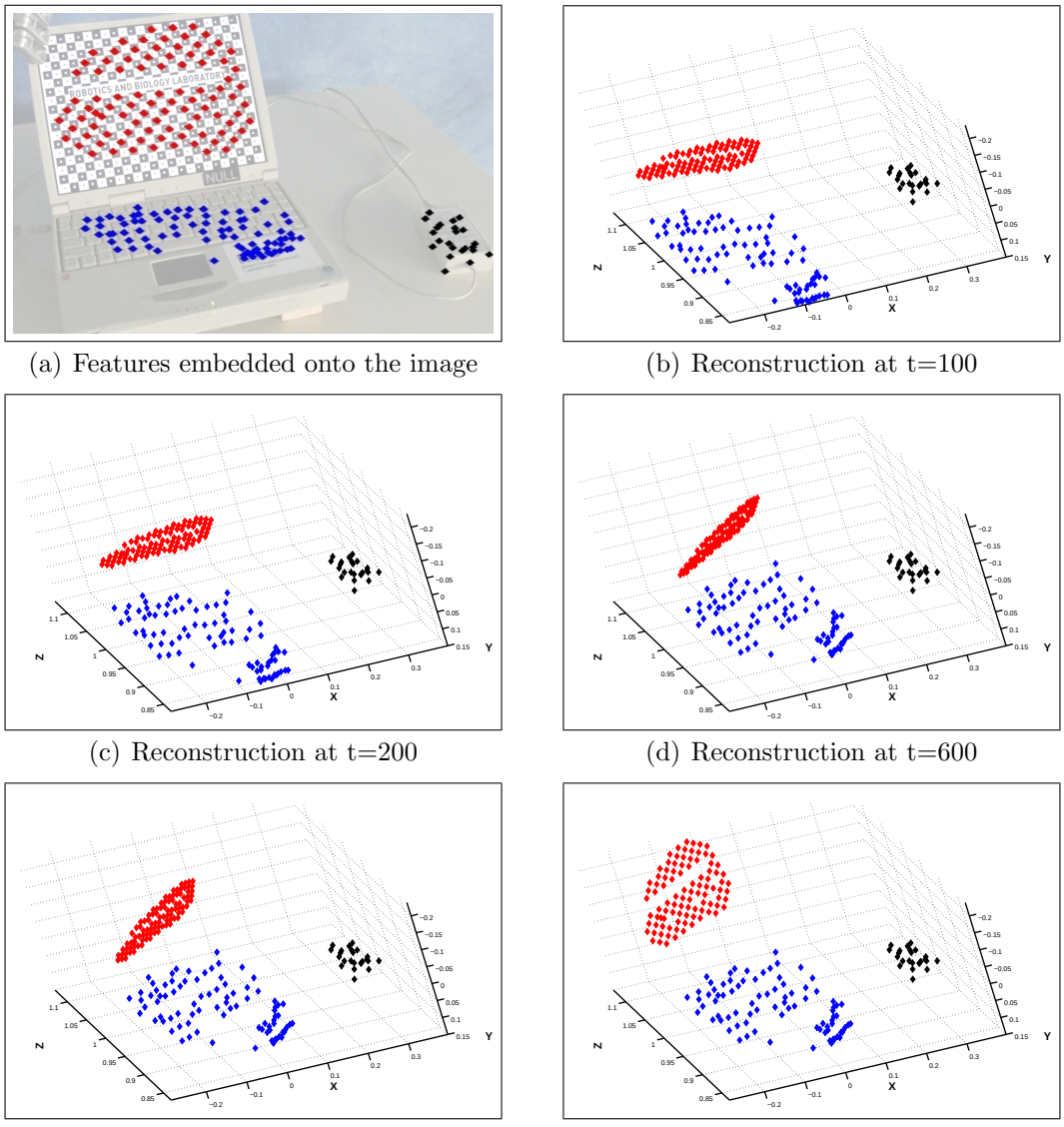
[1] In our case the camera is static. The rigid body is the one undergoing motion. There are, however, no implications on the reconstruction. It does not matter whether the world is static and the observer is moving or vice versa.

ments with the reconstructed distances. We will see in the next chapter that the reconstruction step is accurate enough to support our perceptual skill.

To achieve reliable reconstruction, our approach requires at least 25 features per rigid body. The disadvantage of our approach is that we create a perceptual bottleneck as we wait for enough key-frames to appear. This is generally not a strong limitation as the robot is responsible for generating the motion that creates key-frames. Since the implementation of this part of our perceptual skill, new methods that address this issue have appeared. Pan et al. [103], for example, have proposed a system that creates a 3-D model on-line as the input sequence is being collected. Partial models are reconstructed, and continuously updated as new key-frames become available.

It is also important to notice that monocular structure from motion, without prior information, is always accurate up to a scaling constant. Thus, our three-dimensional reconstruction assumes an arbitrary depth for one of the features, and the rest of the reconstruction is relative to that feature. The correct scale can be recovered either by using stereo information, exploiting prior knowledge about the size of an object, or by relying on proprioception. Thus, to assess the quality of 3-D reconstruction in figure 8.1, we had to determine a scaling factor.

An example of structure and motion reconstruction is shown in Figures 8.2 and 8.3. In the next chapter, we will discuss the impact of this unknown scaling factor on identifying the kinematic structure of objects.



(a) Features embedded onto the image

(b) Reconstruction at t=100

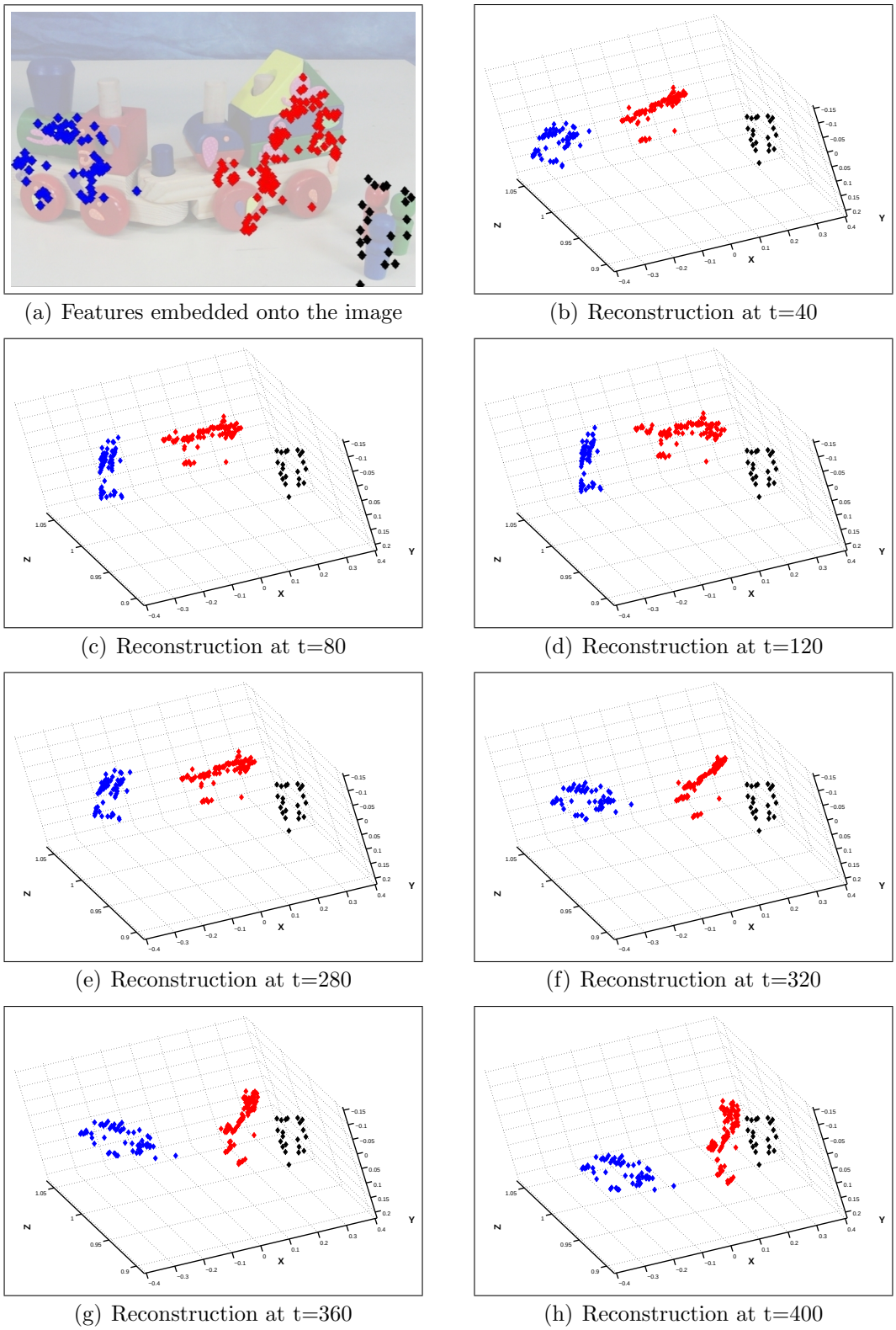
(c) Reconstruction at t=200

(d) Reconstruction at t=600

(e) Reconstruction at t=800

(f) Reconstruction at t=900

Figure 8.2. Three-dimensional reconstruction: laptop. The sequence shows 3-D reconstruction for a laptop in multiple time instances. Colored dots indicate the different rigid bodies (Monitor=red, Keyboard=blue, Power Supply=black).



(a) Features embedded onto the image

(b) Reconstruction at t=40

(c) Reconstruction at t=80

(d) Reconstruction at t=120

(e) Reconstruction at t=280

(f) Reconstruction at t=320

(g) Reconstruction at t=360

(h) Reconstruction at t=400

Figure 8.3. Three-dimensional reconstruction: train. The sequence shows 3-D reconstruction for a train toy in multiple time instances. Colored dots indicate the different rigid bodies (Car=red, Engine=blue, Game Tokens=black).

CHAPTER 9

MODELING THREE-DIMENSIONAL KINEMATIC STRUCTURES

This chapter describes the third factor of our perceptual skill for modeling three-dimensional rigid articulated objects. The first factor provided us with a segmentation of an unstructured scene into rigid bodies. The second factor computed models of the 3-D shape and motion of each of these rigid bodies. In this chapter we ask the following question: how can these three-dimensional models be used to determine the kinematic relationships between rigid bodies?

The third factor of our skill answers this question. The input to this factor is composed of n clusters of 3-D points. Each cluster represents a rigid body and each point corresponds to an image feature (a feature is a projection of the corresponding 3-D point onto the camera plane). For each point we also have its reconstructed 3-D trajectory. The output of this factor should be a set of kinematic constraints, i.e. joint types, that explain the relationships between the $\binom{n}{2}$ pairs of rigid bodies.

In our discussion, we limit ourselves to four types of kinematic constraints: revolute joint, prismatic joint, rigid connection (i.e. no relative motion), and free-motion (i.e. disconnected bodies). The last constraint, free-motion, captures any relative motion that is neither prismatic nor revolute. The algorithm should therefore report its confidence value for each joint type; the joint that best explains the observed motion, i.e. the highest confidence value, is declared as the type of kinematic constraint between the pair of bodies.

To facilitate our discussion, we begin by defining the terminology that will be used throughout this chapter:

Point p	A coordinate in 3-D space represented by $\{x, y, z\}$.
Body B_ϕ	A cluster of points associated with a single rigid body.
Trajectory f	The displacement of a feature over time.
Frame i	The position of a set of features in a specific time.
M	The number of frames.
$B_\phi(i)$	The position of all features associated with body ϕ in frame i. We refer to it as a **point-cloud**.
$\{B_\phi\}_{i=0}^{M-1}$	The trajectories of all features associated with body ϕ in frames 0 to $M-1$.
H	Homogenous 4×4 matrix composed of a rotation matrix R and a translation vector t.
$H_\phi(i)$	The homogenous transform between B_ϕ in frame 0 and B_ϕ in frame i

9.1 Computing Relative motion

To understand the kinematic relationship between a pair of rigid bodies, we must consider the observed relative motion between the bodies. If this motion is such that one body seems to rotate around the other, we will declare the bodies to be connected by a revolute joint. If we observe translation between the bodies, we will declare a prismatic joint. If we detect a motion that is inconsistent with both a revolute and a prismatic joint, we will declare that the bodies are disconnected. In the case where the two bodies have no relative motion between them, we will declare them to be rigidly connected.

The motion of a pair of rigid bodies is typically the sum of two types of motion: global and local (see Figure 9.1). The global motion is that of the entire articulated

object. In other words, this motion component is the same for all rigid parts of the same object. The local motion is the relative motion between a pair of bodies. The local motion is the one that exposes the kinematic relationships. Thus, to analyze the kinematic relationship between two rigid bodies B_1 and B_2, we remove the motion of B_1 from that of B_2 in every frame. The result is that B_1 seems to remain static, while the remaining motion of B_2 is precisely the local motion—the relative motion—between B_1 and B_2.

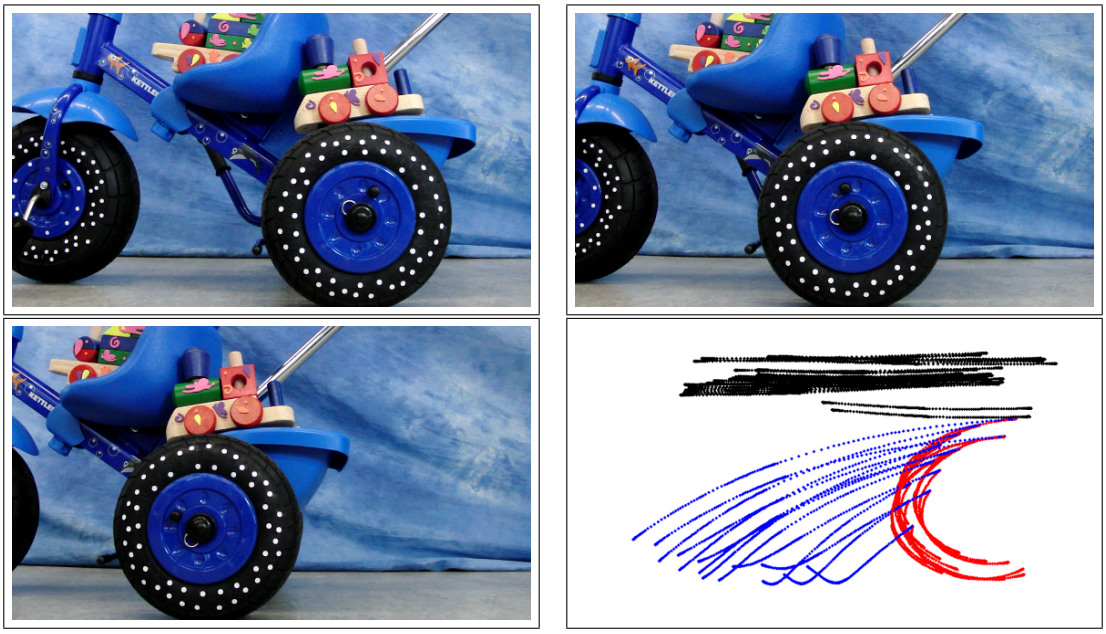


Figure 9.1. Relative motion between the parts of a tricycle. The global trajectories of wheel features (blue), the global trajectories of frame features (black), and the relative motion between the wheel and the frame of the tricycle (red). The relative motion curve reveals the revolute joint connecting the wheel to the frame.

9.1.1 Point-Cloud to Point-Cloud Registration

To analyze the relative motion between bodies B_1 and B_2 we first have to compute the motion of B_1. Thus, for every frame $i = \{0, ..., M-1\}$, we want to compute the transformation between $B_1(0)$ and $B_1(i)$. This transformation is represented by a homogenous $H_{4\times4}$ matrix, composed of a rotation matrix R and a translation vector

t, that relates the two 3-D point-clouds representing body B_1 in frames 0 and i respectively.

Point-cloud to point-cloud registration is well studied. Most of the existing methods address the registration of clouds with different number of points. These methods are usually iterative; for a good review see [91]. Our case is simpler because we are guaranteed to have the same number of point in every point-cloud (each point-cloud represents the same rigid body in different time instances). We can therefore employ a robust closed form solution, as described in [7, 65]. Our registration process requires that there are at least six points in each cloud. Let $P = p_1, ..., p_N$ and $Q = q_1, ..., q_N$ denote the point clouds of a single rigid body at two different time instances, where $N \geq 6$. Our goal is finding a homogenous transformation H between P and Q. Our task can be formulated as an optimization problem with the goal of finding R and t that minimize the least-square error between Q and $H \cdot P$ the two point-clouds (P and Q):

$$MSE = \sum_{j=1}^{N} \|q_j - (Rp_j - t)\|^2 \tag{9.1}$$

To solve this optimization problem, we first decouple translation and rotation. We begin by removing the translational component by centering both clouds around the origin [68]. Let p_c and q_c be the centroids of P and Q respectively. The centered point clouds are given by:

$$p'_j = p_j - p_c \tag{9.2}$$

$$q'_j = q_j - q_c \tag{9.3}$$

Now, with $t = 0$, our optimization problem can be restated considering only rotation:

$$MSE = \sum_{j=1}^{N} \|q'_j - Rp'_j\|^2 \tag{9.4}$$

To find R, we rewrite equation 9.4 as:

$$\begin{aligned} MSE &= \sum_{j=1}^{N} \|q'_j - Rp'_j\|^2 && (9.5)\\ &= \sum_{j=1}^{N} (q'_j - Rp'_j)^T (q'_j - Rp'_j) && (9.6)\\ &= \sum_{j=1}^{N} (q'^T_j q'_j + p'^T_j p'_j - 2q'^T_j R p_j) && (9.7) \end{aligned}$$

Note that $R^T R$ is the identify matrix because R is a rotation matrix. Minimizing the error function in equation 9.7 is equivalent to maximizing $2q'^T_j R p_j$, which leads to a new error function:

$$\begin{aligned} MSE' &= \sum_{j=1}^{N} q'^T_j R p'_j && (9.8)\\ &= tr(R \sum_{j=1}^{N} q'_j p'^T_j) && (9.9)\\ &= tr(RK) && (9.10) \end{aligned}$$

Where $tr(A)$ is the trace of matrix A. That is, the sum of the elements on the main diagonal of A.

The covariance matrix K in 9.10 is defined as $K = \sum_{j=1}^{N} q'_j p'^T_j$. To maximize RK, we compute a Singular Value Decomposition (SVD, [107]): $K = U\Lambda V^T$. If we set $R = VU^T$, than RK becomes $RK = VU^T U\Lambda V^T = V\Lambda V^T$, and $tr(RK)$ is maximized. Thus, the homogenous transformation matrix H becomes:

$$\begin{pmatrix} R & t \\ \vec{0} & 1 \end{pmatrix} = \begin{pmatrix} VU^T & q_c - (VU^T)p_c \\ \vec{0} & 1 \end{pmatrix} \tag{9.11}$$

This solution assumes that the point-clouds are not coplanar. In the coplanar case, there are two optimal solutions, one of which is the desired rotation matrix, while the other is a reflection matrix. We know that a point-cloud is coplanar if one of the eigenvalues of the covariance matrix K is zero. In this case, we can distinguish between the correct rotation matrix and the reflection matrix by computing the determinant of R. If $|R| = -1$, then the correct rotation matrix is simply $-R$ (because the determinant of a valid rotation matrix is always 1).

9.1.2 Removing Global Motion

The above method for point-cloud to point-cloud registration computes the homogenous transform $H_{B_1}(i)$ relating body B_1 in frames i and 0, for $i = \{1, ..., M-1\}$. We refer to this set of transformations as the global motion of B_1. If we apply $H_{B_1}(i)$ to all of the point in $B_2(i)$ to get $B_2'(i) = H_{B_1}(i)B_2(i)$, we effectively remove the motion of B_1 from that of B_2. Thus, the motion of B_2' is the relative motion between B_1 and B_2. This relative motion is marked by the red curve in Figure 9.1.

9.2 Analyzing Kinematic Constraints

To model the kinematic constraints between two rigid bodies B_1 and B_2 we must analyze the relative motion between these bodies (or the motion of B_2'). When analyzing the relative motion, we consider two types of joints: revolute and prismatic. We also detect when the two bodies are rigidly connected. If neither one of these three constraints applies, we declare the two bodies to be disconnected.

There are two challenges associated with analyzing the relative motion B_2'. First, the reconstructed feature trajectories that are the input to this part of our perceptual skill are inherently noisy. They rely on feature trajectories tracked by a low resolution

webcam. When removing the motion of B_1 from that of B_2, we essentially sum two noisy data points. Second, monocular structure from motion, as discussed in chapter 8, provides us with the three-dimensional shape and motion of each rigid body, up to an arbitrary scaling factor. As a result, when comparing B_1 and B_2, we have two arbitrary scaling factors to consider. Simply removing the motion of one from the other only works if they are both scaled by the same constant. For clarity, in this section we ignore the scaling problem and assume that all objects are reconstructed to the correct scale. In section 9.3 we will address the scaling problem.

We now discuss a method for detecting each of the three types of constraints: revolute joint, prismatic joint, and rigid connection. The discussion of each method highlights our approach towards noise. The output of each method is a confidence value about the analyzed joint type. To determine the correct kinematic constraint between B_1 and B_2, we compare the three confidence values.

9.2.1 Modeling Prismatic Joints

A prismatic joint connecting two bodies B_1 and B_2 forces the relative motion between the two bodies (i.e. the motion of B_2') to have the form of a pure translation (see Figure 9.2). That is, if two rigid bodies are connected by a prismatic joint, features on these bodies perform a relative motion in the form of straight, parallel lines of equal lengths.

In practice, due to noise in feature tracking and three-dimensional reconstruction, the relative motion trajectories will vary in orientation and length, and are therefore only rarely straight, parallel and of equal lengths. Thus, our method for detecting prismatic joints must consider the effects of noise. To decide whether the relative motion B_2' is due to a prismatic joint, we translate the relative motion trajectories to the origin of an arbitrary coordinate frame (see Figure 9.3).

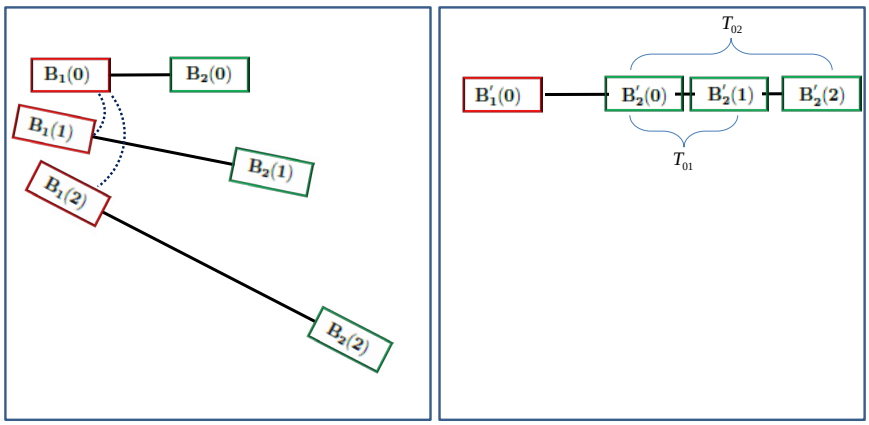


Figure 9.2. Relative motion between bodies connected by a prismatic joint. Left: two bodies B_1 and B_2 translating and rotating. At the same time, the prismatic joint connecting them is actuated. Right: a transformation $H_{B_1}(i)$ that aligns all $B_1(i)$ is computed (i.e. $B_1'(0) = B_1'(1) = B_1'(2)$), and applied to $B_2(i)$ respectively. The relative motion between $B_2'(0)$,$B_2'(1)$, and $B_2'(2)$ reveals the prismatic joint.

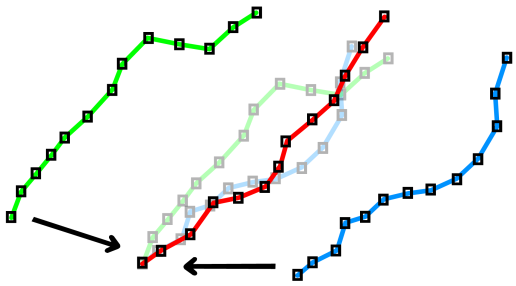


Figure 9.3. Aligning relative motion trajectories. Three relative motion trajectories: blue, green and red. Each trajectory corresponds to the relative motion of one point. The blue and green trajectories are projected to the origin of the red trajectory (transparent blue and green). For a prismatic joint, without any noise, we expect all projected trajectories to be identical and linear.

To determine a confidence value for the presence of a prismatic joint, we fit a cylinder around the projected trajectories. The radius of the cylinder is set to 5% of the longest trajectory. The confidence value $p_{\text{prismatic confidence}}$ is simply the percentage of the trajectories that lie entirely inside the fitted cylinder:

$$\mathrm{p}_{\text{prismatic confidence}} = \frac{\text{length of trajectories inside cylinder}}{\text{sum of lengths of all trajectories}} \tag{9.12}$$

We use a relatively wide cylinder to eliminate the effect of tracking and reconstruction noise. However, when the trajectories are due to a non-prismatic joint, they easily extrude the cylinder. The closer $\mathrm{p}_{\text{prismatic confidence}}$ is to 1, the higher is our confidence that the observed relative motion is due to a prismatic joint. Figure 9.4 shows two examples. The top image shows relative motion trajectories associated with a prismatic joint ($\mathrm{p}_{\text{prismatic confidence}} = 1$). The bottom image shows relative motion trajectories between two disconnected bodies ($\mathrm{p}_{\text{prismatic confidence}} \leq 0.50$).

9.2.2 Modeling Revolute Joints

A revolute joint connecting two bodies B_1 and B_2 forces the relative motion between the two bodies (i.e. the motion of B_2') to have the form of a pure rotation (Figure 9.5). That is, the features on these two rigid bodies trace out a set of parallel arcs centered on the rotational axis.

To compute whether a set of relative motion trajectories can be explained by a revolute joint, we project the trajectories onto a plane we compute from the feature trajectories; this plane is orthogonal to the hypothesized rotational axis. We then scale the projected trajectories and compute how well they match a unit circle. Based on this match, the algorithm computes a confidence value, indicating the degree to which it is certain that the two bodies are connected by a revolute joint.

We begin by fitting a plane to the relative motion trajectory of each feature on body B_2'. To that end, we use the parametric representation $Ax + By + Cz + D = 0$ of a plane. We find the plane parameters A, B, C and D by minimizing the distance of all points associated with the trajectory to the plane:

$$\min_{A,B,C,D} f(A,B,C,D) = \min_{A,B,C,D} \sum_{i=0}^{N} \frac{\|Ax_i + By_i + Cz_i + D\|^2}{A^2 + B^2 + C^2} \tag{9.13}$$

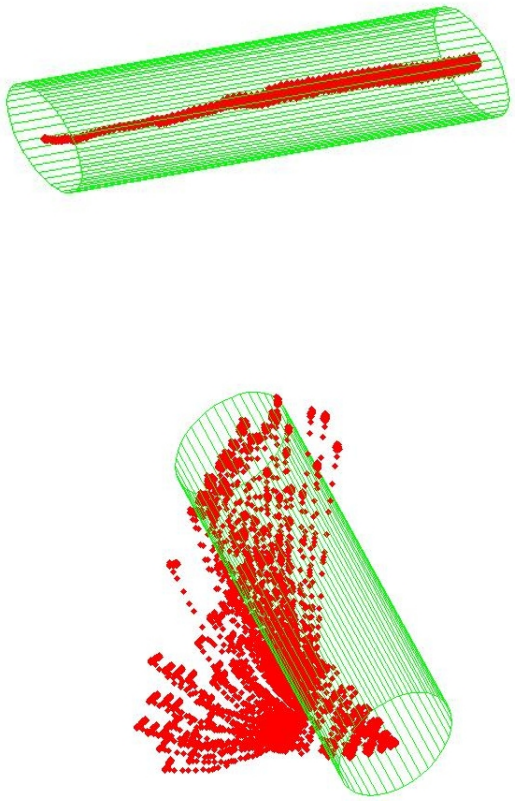


Figure 9.4. Detecting prismatic joints. We fit a cylinder to two sets of relative motion trajectories. Top: relative motion between two elevator doors (prismatic, nicely fit inside the cylinder). Bottom: relative motion between a laptop and its power supply (disconnected, poor fit)

Where x_i, y_i and z_i are the 3-D coordinates of all points on the trajectory of a single feature of $B_2^{'}$. To determine the parametric representation, we require that $i \geq 2$ (i.e. at least 3 trajectory points). When more frames are available, the result becomes more reliable and robust against noise.

Once the best plane is fit to the trajectory, we can use it to transform the trajectory onto the $x-y$ plane. This is done by rotating the plane such that its normal is aligned with the z axis. We then translate the rotated trajectory so the normal is at the origin

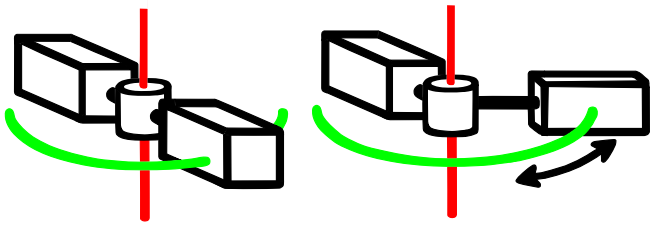


Figure 9.5. A revolute joint between two bodies. The motion of one of them can be described in terms of a rotational component only (green). The rotation is defined with respect to an axis, the point from which the distance to all points on the rotated body remains constant (red).

of our frame. Next, we fit a circle to the data (least square circle fit [130]). And finally, using the circle, we normalize the data onto the unit circle. We repeat this process for all trajectories, until they all lie in unit circle of $x - y$ plane. We extract the axis of rotation by finding a line that passes as closely as possible to the centers of all circles (in the Euclidean least square sense).

To estimate the quality of the fit, we compute the Euclidean distance of each point on every trajectory to the origin of the unit circle. If the relative motion trajectories indeed describe a revolute joint, and without any noise, the distance of all points to the center should be 1. To address noise, we compute a confidence interval c around the unit circle (Figure 9.6). If the distance of all points to the origin is within the interval $[1.0-c, 1.0+c]$, we consider the circle fit to be good. In our experiments, c was set 0.03. We can now compute a confidence value $\mathrm{p}_{\text{revolute confidence}}$ as the percentage of data points which lie inside the interval $[1.0 - c, 1.0 + c]$:

$$\mathrm{p}_{\text{revolute confidence}} = \frac{\text{points inside the interval}}{\text{all points}} \tag{9.14}$$

9.2.3 Modeling Rigid Connections

A rigid connection between two bodies B_1 and B_2 forces the relative motion between the two bodies to remain constant over time. This type of kinematic constraint

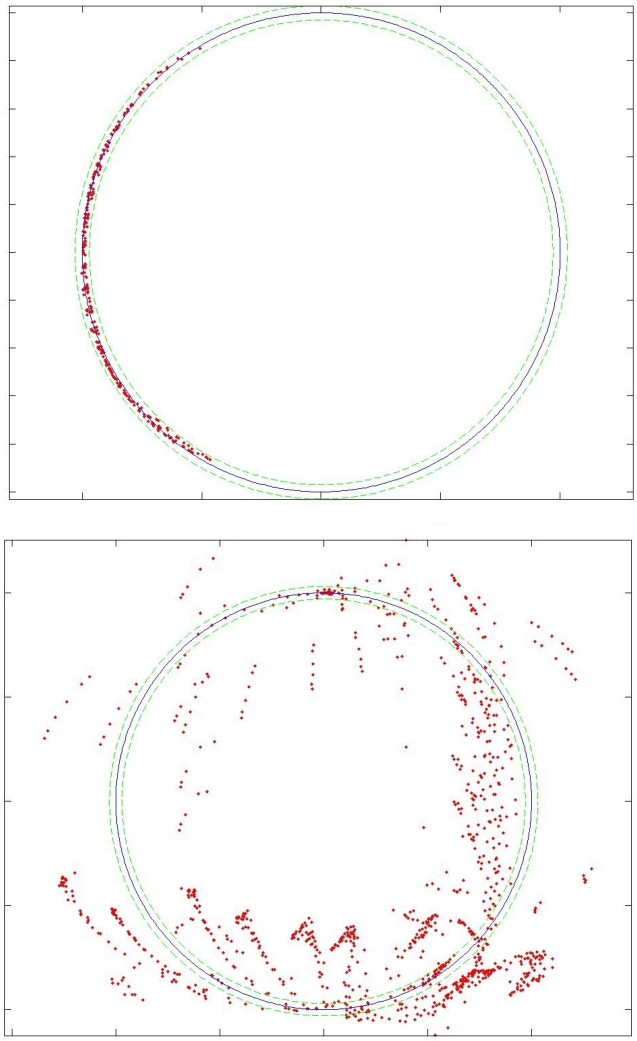


Figure 9.6. Detecting revolute joints. We fit a circle to two sets of relative motion trajectories. Top: relative motion between a laptop's monitor and keyboard (revolute, nicely fit inside the confidence interval around the unit circle). Bottom: relative motion between a box and a background object (disconnected, poor fit)

is important, as it can correct over-segmentation in the previous steps of the perceptual skill.

To determine our confidence score for a rigid connection, we compute $dist(B_1)$ and $dist(B_2)$, the distance traveled by bodies B_1 and B_1 respectively. We also compute the relative motion between the bodies $Rel(B_1, B_2)$. The confidence score is given by:

$$\mathrm{p}_{\text{rigid confidence}} = 1 - \frac{Rel(B_1, B_2)}{\min(dist(B_1), dist(B_2))} \tag{9.15}$$

We assume that the relative motion between bodies is smaller than the global motion. This is a reasonable assumption when interacting with everyday articulated objects.

9.2.4 Collecting The Evidence

We computed a confidence value for three different types of kinematic constraints: revolute, prismatic and rigid. To determine the most likely joint, we now merge these three values:

- $\mathrm{p}_{\text{rigid}} = \mathrm{p}_{\text{rigid conf.}}$
- $\mathrm{p}_{\text{prismatic}} = (1 - \mathrm{p}_{\text{rigid conf.}})\mathrm{p}_{\text{prismatic conf.}}$
- $\mathrm{p}_{\text{revolute}} = (1 - \mathrm{p}_{\text{rigid conf.}})(1 - \mathrm{p}_{\text{prismatic conf.}})\mathrm{p}_{\text{revolute conf.}}$
- $\mathrm{p}_{\text{disconnected}} = (1 - \mathrm{p}_{\text{rigid conf.}})(1 - \mathrm{p}_{\text{prismatic conf.}})(1 - \mathrm{p}_{\text{revolute conf.}})$

Note that to achieve a high confidence value for the revolute case, we require that the confidence value for the prismatic case is low. This is important because every prismatic joint can be explained by a revolute joint with a very far center of rotation. If the algorithm's confidence values for the rigid, prismatic, and revolute cases are low, we determine that the two bodies are disconnected (or connected by a six degree-of-freedom joint). We declare bodies B_1 and B_2 to be connected by the joint type that achieves the highest confidence score.

Figures 9.7 and 9.8 show the process of modeling the kinematic structure of a train toy. The results are summarized in Table 9.1. The skill correctly detects that the car and engine are connected via a revolute joint. It also discovers that the train is disconnected from the static games tokens.

Figure 9.7. Interacting with a train toy. Three frames showing different configurations of the scene.

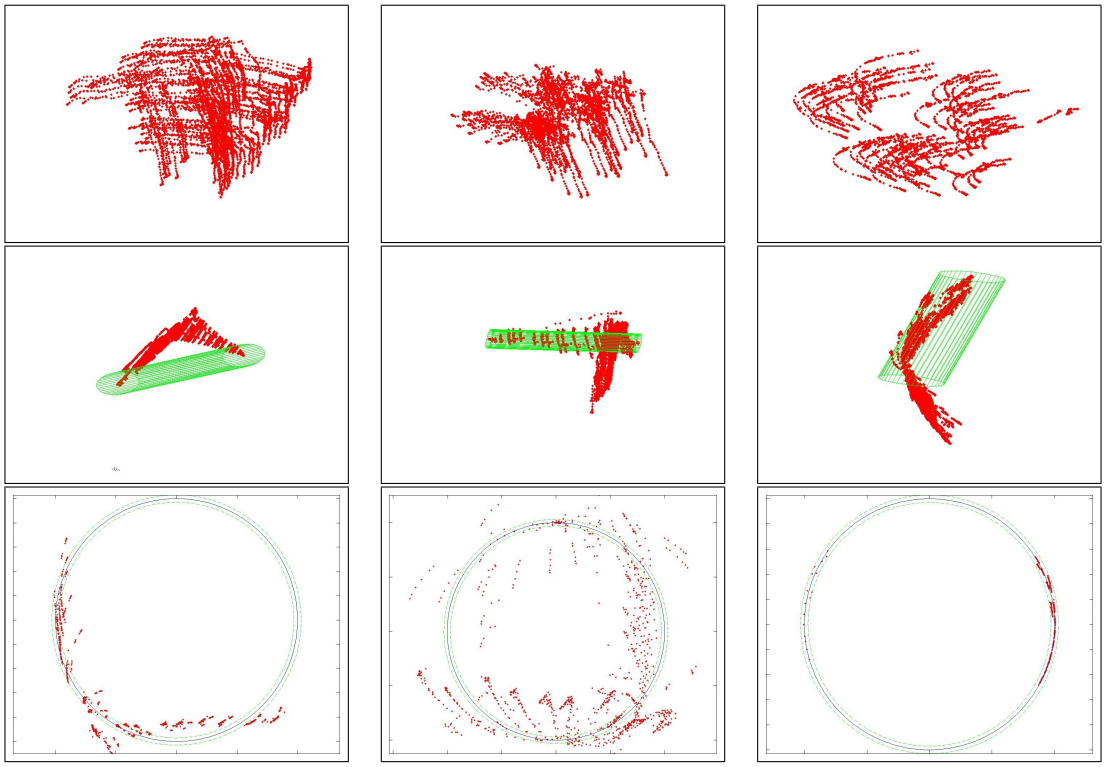


Figure 9.8. Modeling the kinematic structure of a train toy. **Top row:** The relative motion trajectories (B_2'). **Middle row:** The results of prismatic joint fit. **Bottom row:** The results of revolute joint fit. **Left column:** Engine and game tokens. Both cylinder and circle fit fail; the kinematic relationship is declared as disconnected. **Middle column:** Car and game tokens. Both cylinder and circle fit fail; the kinematic relationship is declared as disconnected. **Right column:** Engine and car. Cylinder fit fails. Circle fit is very good; the kinematic relationship is declared as revolute.

Results for Train Experiment				
Parts	Confidence scores for Relationships			
	Rigid	Prismatic	Revolute	Disc
Game Tokens-Engine	0.000	0.000	0.025	**0.975**
Game Tokens-Car	0.000	0.000	0.250	**0.750**
Engine-Car	0.000	0.004	**0.996**	0.000

Table 9.1. Modeling the kinematic structure of a train toy. Our skill correctly identifies the revolute joint between the engine and car, with a confidence value of 99.6%. It also detects that the train is disconnected from the static game token, with confidence values of 97.5% and 75.0%

9.3 The Scaling Problem

The three-dimensional point-clouds that we analyze are generated by a monocular 3-D reconstruction algorithm. This reconstruction, by definition, is correct up to a scaling factor (see Figure 9.9). In our discussion so far, we have assumed that all bodies are reconstructed up to **the same** scaling factor. In practice, however, this is almost never the case.

Let us consider the relative motion between two rigid bodies B_1 and B_2. Further, let us assume that B_1 and B_2 are reconstructed up to a scale of λ_1 and λ_2 respectively. A point $p_1 = (x_1, y_1, z_1)$ on body B_1 will be reconstructed as $\tilde{p}_1 = (\lambda_1 x_1, \lambda_1 y_1, \lambda_1 z_1)$. Similarly, a point $p_2 = (x_2, y_2, z_2)$ on body B_2 will be reconstructed as $\tilde{p}_2 = (\lambda_2 x_2, \lambda_2 y_2, \lambda_2 z_2)$. The distance between the $\tilde{p}_1$ and $\tilde{p}_2$ is:

$$d(\tilde{p}_1, \tilde{p}_2) = \sqrt{(d_x)^2 + (d_y)^2 + (d_z)^2} \tag{9.16}$$

where:

$$d_x = \lambda_1 x_1 - \lambda_2 x_2 \tag{9.17}$$

$$d_y = \lambda_1 y_1 - \lambda_2 y_2 \tag{9.18}$$

$$d_z = \lambda_1 z_1 - \lambda_2 z_2 \tag{9.19}$$

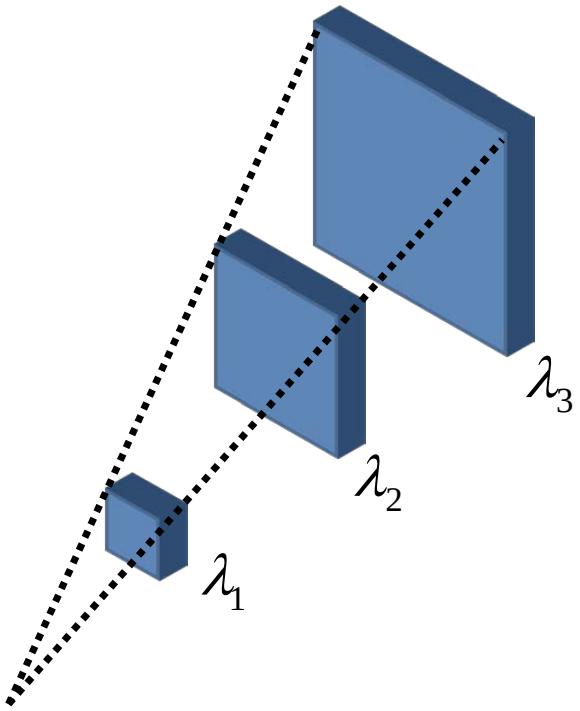


Figure 9.9. Scale ambiguity. The reconstructed three-dimensional shape of an object is known only up to a scaling factor λ_i. Scaling effects both the estimated position of every point on the object and the translational component of its trajectory (e.g. scaling an object by a factor of 2 means that it is twice as large, twice as far, and translates twice as fast).

If both B_1 and B_2 undergo identical translation: $t = (t_x, t_y, t_z)$, the position of the points become: $\hat{p}_1 = \tilde{p}_1 + \lambda_1 t(x, y, z) = (\lambda_1(x_1 + t_x), \lambda_1(y_1 + t_y), \lambda_1(z_1 + z_x))$ and $\hat{p}_2 = \tilde{p}_2 + \lambda_2 t(x, y, z) = (\lambda_2(x_2 + t_x), \lambda_2(y_2 + t_y), \lambda_2(z_2 + z_x))$. We would expect that the distance between $\hat{p}_1$ and $\hat{p}_2$ remains the same as the two bodies undergo the same motion. However, because the points are reconstructed at different scales, the distance between the points changes to:

$$d(\hat{p}_1, \hat{p}_2) = \sqrt{(\hat{d}_x)^2 + (\hat{d}_y)^2 + (\hat{d}_z)^2} \tag{9.20}$$

where:

$$\hat{d}_x = \lambda_1(x_1 + t_x) - \lambda_2(x_2 + t_x) = d_x + (\lambda_1 - \lambda_2)t_x \tag{9.21}$$

$$\hat{d}_y = \lambda_1(y_1 + t_y) - \lambda_2(y_2 + t_y) = d_y + (\lambda_1 - \lambda_2)t_y \tag{9.22}$$

$$\hat{d}_z = \lambda_1(z_1 + t_z) - \lambda_2(z_2 + t_z) = d_z + (\lambda_1 - \lambda_2)t_z \tag{9.23}$$

Thus, to determine whether a change in distance between bodies B_1 and B_2 is due to a joint, we must resolve this scaling ambiguity.

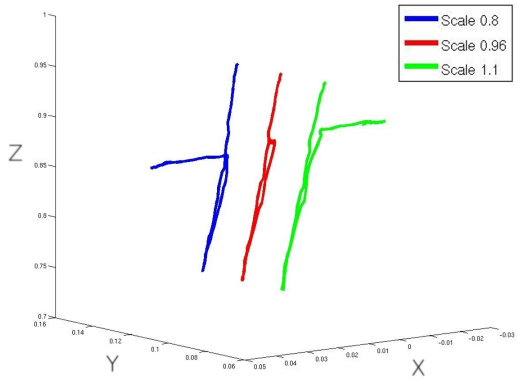


Figure 9.10. Relative motion at different scales (drawers). At the correct relative scale, the prismatic joint between the drawers is revealed (red). An additional translational component is added at the wrong relative scale (blue and green).

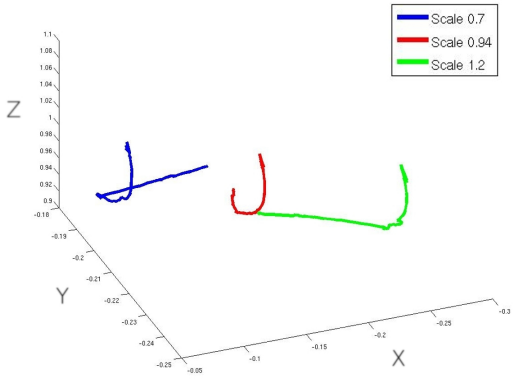


Figure 9.11. Relative motion at different scales (laptop). At the correct relative scale, the revolute joint between the laptop's monitor and keyboard is revealed (red). Additional translational component is added at the wrong relative scale (blue and green).

The impact of scale ambiguity on joint detection is illustrated in Figures 9.10 and 9.11. In both figures, the relative motion between two rigid bodies is shown at different relative scales (if bodies B_1 and B_2 are reconstructed up to scales λ_1 and λ_2 respectively, then the relative scale between them is $k = \frac{\lambda_1}{\lambda_2}$). Figure 9.10 shows the relative motion between two drawers. At the correct scale (red trajectory, $k = 0.96$), the motion reveals the presence of a prismatic joint. At the wrong scale, however, we would declare the bodies to be disconnected. Similarly, figure 9.11 shows the relative motion between a laptop monitor and keyboard. At the correct scale (red trajectory, $k = 0.94$), the presence of a revolute joint is exposed. At the wrong scale, however, we would declare the bodies to be disconnected.

To infer the correct relative scale k, we leverage the structure of the task. The relative motion between two bodies must be explained as either prismatic, revolute, rigid or disconnected. Therefore, we can determine the correct relative scale by searching the k which maximizes the confidence score of one of the four kinematic constraints. Our algorithm considers relative scales in the range of $k \in [0.5, 1.5]$, with a step size of 0.01. This approach is naive, but quite efficient. It relies on the fact that the parts of most real-world objects share a similar form factor. Future work could replace this method with more efficient search (i.e. gradient descent).

We note that to recover the real-world scale of objects, we could simply rely on the robot's proprioception. Our relative scale recovery, however, remains necessary as we would like our perceptual skill to also work when the robot observes objects put in motion by a teacher.

CHAPTER 10

EXPERIMENTAL EVALUATION OF MODELING THREE-DIMENSIONAL OBJECTS

By combining the three components (factors) described in Chapters 7, 8, and 9, we obtain a robust perceptual skill to identify, model, and track novel objects. This skill only makes two general assumptions: First, it assumes that the scene contains sufficient texture to identify visual features. And second, it assumes that the robot is able to make contact with the environment and to cause motion of the rigid bodies. This chapter discusses the experimental results obtained by combining the three factors of our perceptual skill: rigid body segmentation, three-dimensional reconstruction, and joint detection.

The goal of our experiments is to show that the robot can acquire the kinematic model of a variety of real-world articulated objects. In each experiment, the robot interacts with its environment while observing the outcome of this interaction. It then computes the shape and kinematic model of each object. We consider an experiment to be successful if the extracted model qualitatively matches the expected model. That is, we rely on visual inspection to determine whether the extracted axis of rotation and translation is aligned with the position and orientation of the real axis.

10.1 Experimental Setup

We validate the proposed method for perceiving 3-D rigid articulated objects in real-world experiments. The experiments are conducted with our mobile manipulator UMan (see Figures 6.9 and 10.1). UMan has 14 degrees of freedom: a holonomic

mobile base with three planar degrees of freedom, a seven degree-of-freedom Whole Arm Manipulator and a three-fingered hand with four degrees of freedom.

Figure 10.1. Interacting with a toy train. Our Mobile Manipulator interacts with a toy train, and identifies a set of rigid bodies: train engine, train car, and wooden figures. These rigid bodies are segmented from the background throughout the interaction.

The robot interacts with various articulated objects. These objects vary in scale, shape, color, texture, and kinematic structure. The interaction itself is scripted. To perceive the environment, we use an off-the-shelf web camera with a resolution of 640×480 pixels. The camera provides a 30 frames-per-second video stream of the scene throughout the interaction. The position and orientation of the camera with respect to the scene is arbitrary.

In the following we describe experiments with 10 real-world objects: a box, an office door, a refrigerator, a laptop, a cupboard, a shelf on wheels, a toy train, a tricycle, an elevator, and a set of drawers (see Figure 10.2). An experiment lasts between 10 to 30 seconds.

Figure 10.2. Ten real-world objects. Our robot has interacted with and modeled these objects. The objects differ in size, shape, color, texture and their kinematic structure. Modeling their shape and kinematic structure provides the robot with necessary information for manipulation.

10.2 Experimental Results

To assess the performance of our perceptual skill, we now analyze ten experiments. In each experiment the robot interacted with one of the objects in Figure 10.2. For each experiment, we analyze both the correctness and the precision of the identified rigid bodies and kinematic constraints. We present five figures and one table per experiment. The first three figures show the scene before, during, and after the

interaction. The fourth figure shows the scene, overlayed with the detected rigid bodies (features are color coded according to rigid body association, graph clusters are shown in white). The fifth figure (large) shows the detected joints (pink).

10.2.1 Experiment I: Box

In the first experiment (figures 10.3, 10.4, and table 10.1), the robot interacts with a box by pushing it and closing the flap. During the experiment the robot first slides the box on the table. Then it interacts with the flap by pushing it downwards. We detect three clusters: one cluster is associated with the box, the second with the flap, and the third cluster with the picture cube (a static object). During the first factor, the fundamental matrix predictor identifies the difference in the three-dimensional motion of the three bodies. The long distance predictor provide information separating the picture cube from the box. And the short distance predictor reinforces the connectivity between close features on each rigid body.

The second factor computes the three-dimensional motion, enabling the third factor to determine, with high confidence, a revolute joint between the two box parts (97%) and that the flap is disconnected from the background (93%). The orientation and position of the axis of rotation are very accurate. Joint detection is less certain about the connectivity between the body of the box and the background (80% disconnected, 20% revolute). The lower confidence is due to the fact that the robot has actually pushed the box along an arc. An important property of our perceptual skill is its ability to adapt to new evidence. Further interaction with the box is likely to generate trajectories that are more difficult to explain as revolute. As a result, the robot's confidence in that the body of the box is disconnected from the picture cube will increase.

10.2.2 Experiment II: Door

In the second experiment (figures 10.5, 10.6, and table 10.2), the robot interacts with an office door. During the experiment the robot simply pushes open the door. To distinguish between the door and the frame, we rely on the fundamental matrix and triangulation predictors. The long distance predictor further supports this distinction. In this experiment, color-based segmentation provides us with misleading information. It encourages the clustering of all features together because most of the scene has a uniform yellow texture. Our perceptual skill is robust and reliable because it does not depend on a single property, such as color segmentation. The strong signals provided by other predictors enable the robot to correctly separate the door from the frame. Our skill correctly detects a revolute joint with 100% confidence. The orientation of the axis of rotation is parallel to the true axis, but its position is slightly offset.

10.2.3 Experiment III: Refrigerator

The third experiment (figures 10.7, 10.8, and table 10.3) is similar to the second, except that here the robot interacts with a smaller object—a refrigerator door. In contrast with the previous experiment, here the robot pulls on the door to open it. In this experiment, the objects' textures and colors are different. Also, the lighting conditions and the viewing angle are different than in the office door experiment. The result, however, remains similar, and the robot differentiates between the door, the frame, and two other distant static objects. Our perceptual skill detects a rigid joint between the static clusters and a revolute joint between the refrigerator door and the background with 100% confidence. The three static background clusters were separated during the earlier stages of the skill. In this stage, we discover that they are rigidly connected, and therefore correct the over-segmentation. Because the observed motion was small, the position of the axis is not very precise. Nevertheless, the axis'

orientation is accurate. We believe that this model is good enough to enable a robot to plan and execute a door opening action.

10.2.4 Experiment IV: Laptop

In the fourth experiment the robot interacts with a laptop (figures 10.9, 10.10, and table 10.4). During the experiment the robot slides the laptop on the table, and pushes on the lid. As a result, our skill identifies three clusters: the static power supply, the laptop's keyboard and its screen. The revolute joint between the keyboard and the screen is detected correctly (position and orientation), and with high confidence (95%). We also detect that the screen is disconnected from the static power supply (80%). The robot's interaction with the laptop was such that it moved along an arc. As a result, a revolute joint is detected between the keyboard and the power supply, with a low confidence of 70%. Further interactions would reveal that the keyboard is disconnected from the background.

10.2.5 Experiment V: Cupboard

In the fifth experiment, the robot interacts with a cupboard's door (figures 10.11, 10.12, and table 10.5). During the experiment the robot opens the door by translating it. The robot is able to distinguish between two static bodies: the picture cube and the left door. This success is due to the long distance predictor. It also detects a third rigid body: the right (moving) door. Our perceptual skill detects a prismatic joint between the moving door and the two static objects with a very high confidence value of 99.7%. The orientation of the axis is correct (approximately parallel to the floor). The rigid joint between the static door and the picture cube is detected with 100% confidence.

10.2.6 Experiment VI: Shelf on Wheels

In the sixth experiment, the robot interacts with a shelf on wheels (figures 10.13, 10.14, and table 10.6). During the experiment the robot pushes the shelf on the floor, tracing a straight line. Our perceptual skill separates the moving bodies from the static background. Because of the large distance between them, the objects on the top and bottom shelves are identified as different rigid bodies. Our skill detects, however, that the two bodies are rigidly connected, effectively correcting for over-segmentation. The confidence level that we compute for the two shelves being rigidly connected is of 69%. A careful examination of the experiment's movie reveals a small displacement of the objects on the top shelf as the table translates, causing this lower confidence value. A prismatic joint is detected between each shelf and the background (92% and 99%). The orientation of the joints is accurate (approximately parallel to the ground).

10.2.7 Experiment VII: Toy Train

The seventh experiment shows an interaction with a toy train (figures 10.15, 10.16, and table 10.7). During the experiment the robot pushes the engine to the right, and then apply force at the joint to translate the engine and the car towards the camera.

Here, color segmentation alone would fail as each rigid part is composed of multiple brightly colored blocks, and the base of the engine and car have identical wooden texture. Instead, our algorithm relies on the strong motion signal to distinguish between the static wooden figures and the two parts of the train. The train's revolute joint is detected with a confidence value of 99%. The joint's position and orientation are very accurate. The train parts are also segmented from the background by a disconnected joint with 97% (engine) and 75% (car) confidence.

10.2.8 Experiment VIII: Tricycle

In the eighth experiment, the robot interacts with a tricycle (figures 10.17, 10.18, and table 10.8). During the experiment the robot pushes the tricycle from right to left. Our algorithm over-segments the scene in this experiment. Rather than the expected two clusters, it discovers four clusters. Two clusters are associated with the wheel. Because of the complex motion of the wheel (simultaneous translation and rotation), it is difficult to find a fundamental matrix motion hypothesis explaining the wheel as a single body. The other two clusters are associated with the frame of the tricycle (i.e. toys on the seat and in the trunk that are static with respect to the frame). The long distance and color segmentation predictors separate these two clusters. During the analysis of the kinematic structure (third factor), the over-segmentation of the wheel and frame is corrected. The algorithm assigns a very high confidence to the two parts of the frame being rigidly connected (99.6%) and a maximum confidence for the parts of the wheel being rigidly connected (100.0%). Our analysis considers a virtual background body (not showed in the image). It detects that the wheel (both clusters) and the frame (both clusters) are disconnected from the static background. It also detects a revolute joint with a very high confidence level (99.9%) between the wheel and the frame of the tricycle. The position of the joint is very accurate. Also its orientation is accurate (approximately perpendicular to the wheel). Our perceptual skill succeeds in this case, despite the multiple moving bodies undergoing complex motion because it can detect and analyze any number of rigid bodies, even when moving simultaneously.

10.2.9 Experiment IX: Elevator

The interaction with the environment in the ninth experiment is different (figures 10.19, 10.20, and table 10.9). Here, the robot pushes on a button while observing the motion of the elevator's doors. The robot successfully identifies three

rigid bodies: the two doors and the elevator's frame. The relative motion predictor provides a strong cue here, as the motion is approximately parallel to the camera plane. This experiment demonstrates that our skill remains unchanged when interaction is performed by a teacher, who could have pushed the button instead of the robot. Our perceptual skill correctly discovers three prismatic joints between the two doors and the frame of the elevator (all three confidence values are above 95%). Two joints are between the the frame and the two doors; the third joint—without physical manifestation—exists between the two doors and is indicated by the longer pink line. The computed orientation of all three axes is correct: they are parallel to the floor.

10.2.10 Experiment X: Set of Drawers

In the tenth and final experiment, the robot interacts with a set of drawers (figures 10.21, 10.22, and table 10.10). During this experiment the robot first opens the bottom drawer, then it opens the top drawer, and finally it slides the entire set of drawers from left to right.

Our skill identifies the frame and the two drawers. In this experiment, the triangulation predictor does not provide useful information as the features never violate the neighborhood relationship. In contrast, color segmentation and motion information do provide the necessary information. Our perceptual skill detects the kinematic structure (the prismatic joints) with high confidence (above 97% in all cases). Note that the prismatic axes are pointing downwards. This is what we expect to see given the camera's perspective.

Figure 10.3. Identifying rigid bodies: box. Left to Right: The first column shows the box before the interaction; the second column shows the interaction itself; the third column shows the box after the interaction; and the fourth column shows the results of segmenting the graph of tracked features into clusters of features, where each cluster is associated with a different rigid body. Note how small the overall motion of the box is.

Figure 10.4. Detecting the kinematic structure of a box. Three clusters are found; a static cluster on the picture cube (black), the body of the box (blue), and the top flap (red). A revolute joint (pink) between the body of the box and the top flap is found.

Results for the Box Experiment				
Parts	Confidence scores for Relationships			
	Rigid	Prismatic	Revolute	Disc
Picture Cube - Box Body	0.000	0.000	0.205	**0.794**
Picture Cube - Box Top	0.000	0.000	0.070	**0.930**
Box Body - Box Top	0.000	0.000	**0.968**	0.032

Table 10.1. Detecting the kinematic structure of a box. A revolute joint between the body of the box and the top flap is found with high confidence (97%). The two parts of the box are disconnected from the static picture cube (confidence: 93% and 80%).

Figure 10.5. Identifying rigid bodies: door. Left to Right: The first column shows the door before the interaction; the second column shows the interaction itself; the third column shows the door after the interaction; and the fourth column shows the results of segmenting the graph of tracked features into clusters of features, where each cluster is associated with a different rigid body.

Figure 10.6. Detecting the kinematic structure of a door. Two clusters are found; a static cluster on the frame of the door (blue), and the door itself (red). A revolute joint (pink) between the door and its frame is found.

Results for the Door Experiment				
Parts	Confidence scores for Relationships			
	Rigid	Prismatic	Revolute	Disc
Frame - Door	0.000	0.000	**1.000**	0.000

Table 10.2. Detecting the kinematic structure of a door. A revolute joint between the door and its frame is found with maximum confidence (100%).

Figure 10.7. Identifying rigid bodies: refrigerator. Left to Right: The first column shows the refrigerator before the interaction; the second column shows the interaction itself; the third column shows the refrigerator after the interaction; and the fourth column shows the results of segmenting the graph of tracked features into clusters of features, where each cluster is associated with a different rigid body.

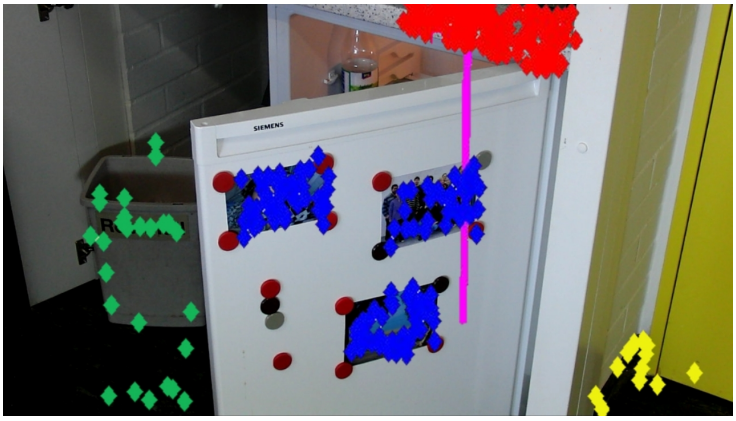


Figure 10.8. Detecting the kinematic structure of a refrigerator. Four clusters are found; Three static background cluster (green, yellow, and red), and one cluster on the Refrigerator door (blue). A revolute joint (pink) between the Refrigerator door cluster and the background clusters is found.

Results for the Box Experiment				
Parts	Confidence scores for Relationships			
	Rigid	Prismatic	Revolute	Disc
Refrigerator - Counter Top	0.000	0.000	**1.000**	0.000
Refrigerator - Trash Can	0.000	0.000	**1.000**	0.000
Refrigerator - Door	0.000	0.000	**1.000**	0.000
Counter Top - Trash Can	**1.000**	0.000	0.000	0.000
Counter Top - Door	**1.000**	0.000	0.000	0.000
Trash Can - Door	**1.000**	0.000	0.000	0.000

Table 10.3. Detecting the kinematic structure of a refrigerator. A revolute joint between the Refrigerator door cluster and the three background clusters is found with maximum confidence (100%). The three clusters that are associated with the background are declared rigidly connected, therefore correcting for over segmentation in earlier stages of the skill.

Figure 10.9. Identifying rigid bodies: laptop. Left to Right: The first column shows the laptop before the interaction; the second column shows the interaction itself; the third column shows the laptop after the interaction; and the fourth column shows the results of segmenting the graph of tracked features into clusters of features, where each cluster is associated with a different rigid body.

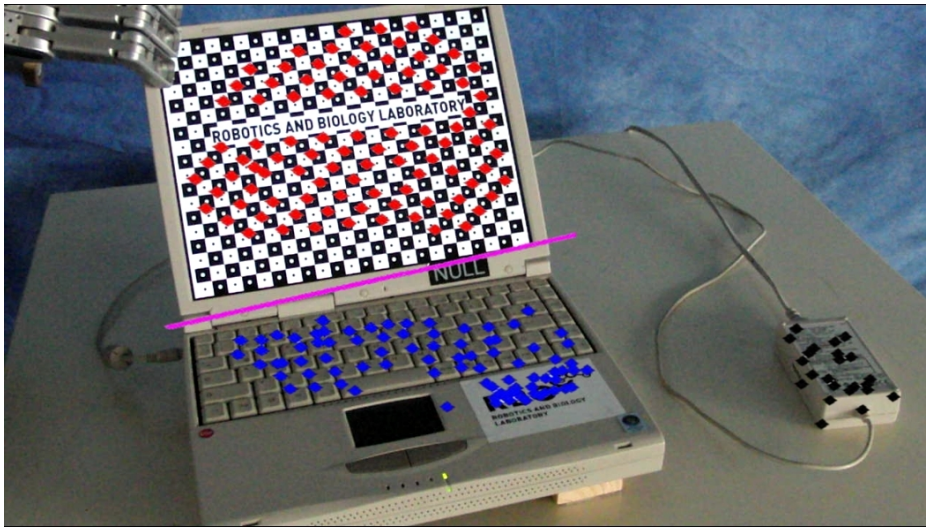


Figure 10.10. Detecting the kinematic structure of a laptop. Three clusters are detected; the screen (red), the keyboard (blue) and a static background cluster associated with the power supply (black). A revolute joint (pink) between the keyboard and screen is found.

Results for the Box Experiment				
Parts	Confidence scores for Relationships			
	Rigid	Prismatic	Revolute	Disc
Power Supply - Keyboard	0.000	0.000	**0.703**	0.297
Power Supply - Screen	0.000	0.000	0.204	**0.796**
Keyboard - Screen	0.000	0.000	**0.946**	0.054

Table 10.4. Detecting the kinematic structure of a laptop. A revolute joint between the keyboard and screen is found with high confidence (95%). The screen is found to be disconnected from the power supply (80%). The interaction with the laptop was such that it moved along an arc. We therefore find a revolute joint between the keyboard and the power supply, with a low confidence of 70%. More interactions with the laptop will expose the true nature of the kinematic relationship (disconnected).

Figure 10.11. Identifying rigid bodies: cupboard. Left to Right: The first column shows the cupboard before the interaction; the second column shows the interaction itself; the third column shows the cupboard after the interaction; and the fourth column shows the results of segmenting the graph of tracked features into clusters of features, where each cluster is associated with a different rigid body.

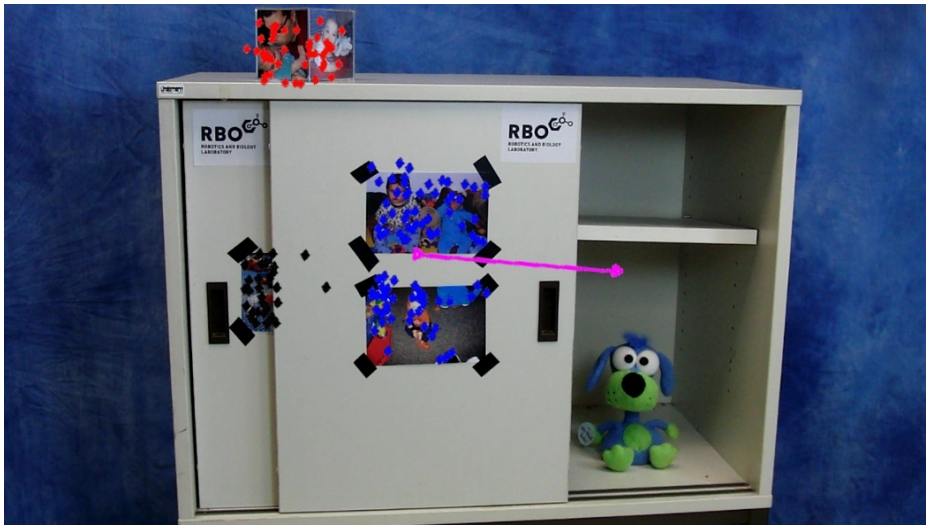


Figure 10.12. Detecting the kinematic structure of a cupboard. Three clusters are detected; the right door (blue), the left door (black) and a static background cluster associated with the picture cube (red). A prismatic joint (pink) between the right (moving) door and both the left (static) door and the picture cube is found. Both axes are shown in the figure (overlapping). A rigid connection is detected between the two static clusters (left door and picture cube).

Results for the Box Experiment				
Parts	Confidence scores for Relationships			
	Rigid	Prismatic	Revolute	Disc
Picture Cube-Right Door	0.000	**0.997**	0.000	0.003
Picture Cube-Left Door	**1.000**	0.000	0.000	0.000
Right Door-Left Door	0.000	**0.997**	0.000	0.003

Table 10.5. Detecting the kinematic structure of a cupboard. A prismatic joint between the moving door and the two static objects is found with very high confidence (99.7%). The rigid constraint between the static door and the picture cube is detected with maximum confidence (100%).

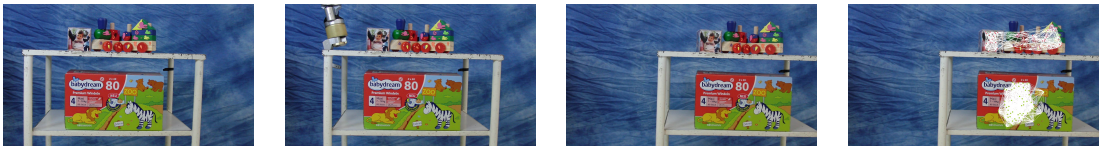


Figure 10.13. Identifying rigid bodies: shelf on wheels. Left to Right: The first column shows the shelf before the interaction; the second column shows the interaction itself; the third column shows the shelf after the interaction; and the fourth column shows the results of segmenting the graph of tracked features into clusters of features, where each cluster is associated with a different rigid body.

Figure 10.14. Detecting the kinematic structure of a shelf on wheels. Two clusters are detected; the top shelf (blue), and the bottom shelf (red). Two prismatic joints (pink) are found: one joint is between the top shelf and the background, and the second is between the bottom shelf and the background. The skill correctly determines the two shelf clusters to be rigidly connected, but has low confidence because the train moved slightly.

Results for the Box Experiment				
Parts	Confidence scores for Relationships			
	Rigid	Prismatic	Revolute	Disc
Background-Top Shelf	0.000	**0.917**	0.000	0.083
Background-Bottom Shelf	0.000	**0.995**	0.000	0.005
Top Shelf-Bottom Shelf	**0.690**	0.013	0.135	0.162

Table 10.6. Detecting the kinematic structure of a shelf on wheels. A prismatic joint between the top shelf and background is detected with high confidence (91.7%). A second prismatic joint is detected between the bottom shelf and the background with a higher confidence level of 99.5%. The rigid constraint between the two shelves detected with a confidence of (69%).

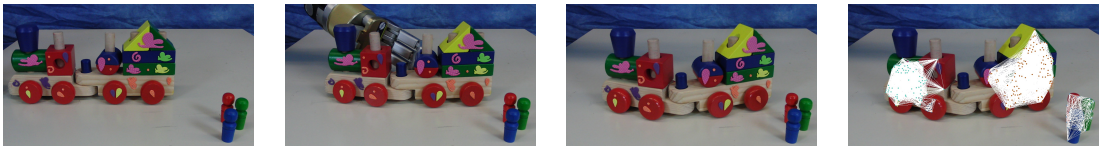


Figure 10.15. Identifying rigid bodies: toy train. Left to Right: The first column shows the train before the interaction; the second column shows the interaction itself; the third column shows the train after the interaction; and the fourth column shows the results of segmenting the graph of tracked features into clusters of features, where each cluster is associated with a different rigid body.

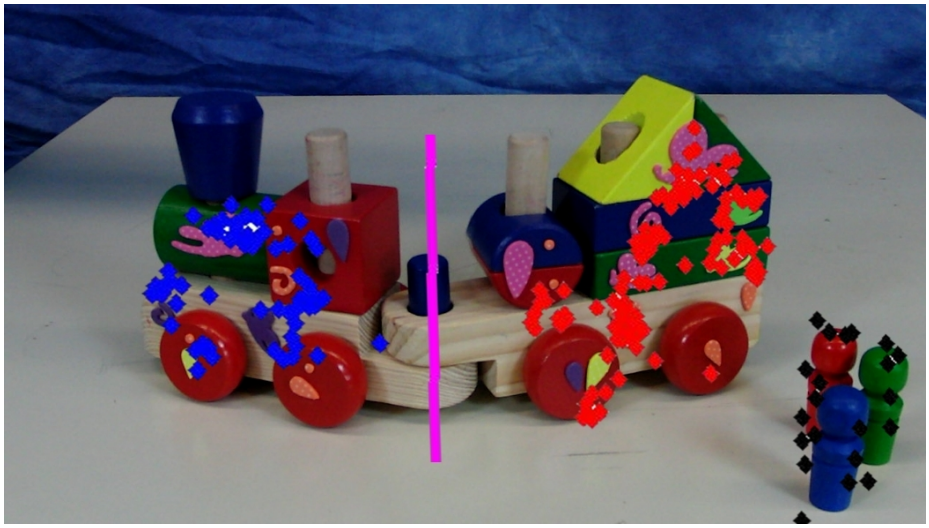


Figure 10.16. Detecting the kinematic structure of a toy train. Three clusters are detected; train engine (blue), train car (red), and static game tokens (black). The kinematic relationship between the train parts correctly identified as revolute (pink).

Results for the Box Experiment				
Parts	Confidence scores for Relationships			
	Rigid	Prismatic	Revolute	Disc
Game Tokens-Engine	0.000	0.000	0.025	**0.975**
Game Tokens-Car	0.000	0.000	0.250	**0.750**
Engine-Car	0.000	0.004	**0.996**	0.000

Table 10.7. Detecting the kinematic structure of a toy train. The train's revolute joint connecting the engine and car is detected with a very high confidence value of 99%. The train is also separated from the background (static game tokens); a disconnected joint is identified with 97% (engine) and 75% (car) confidence.

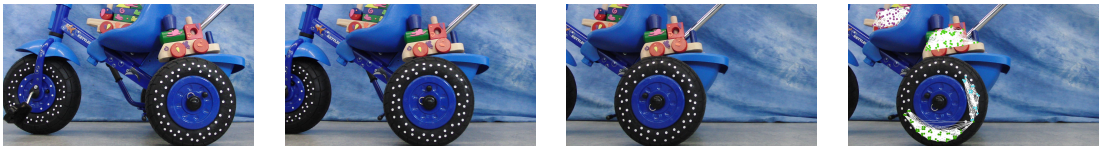


Figure 10.17. Identifying rigid bodies: tricycle. Left to Right: The first column shows the tricycle before the interaction; the second column shows the interaction itself; the third column shows the tricycle after the interaction; and the fourth column shows the results of segmenting the graph of tracked features into clusters of features, where each cluster is associated with a different rigid body.

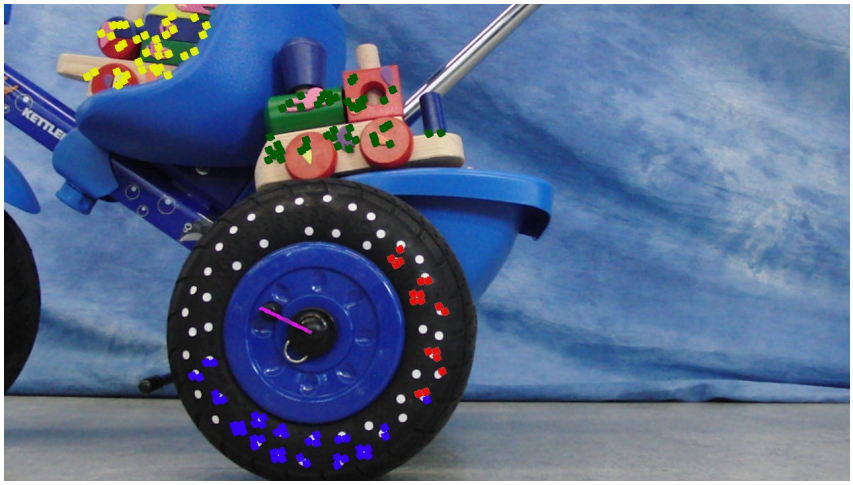


Figure 10.18. Detecting the kinematic structure of a tricycle. Four clusters are detected; two wheel clusters (red and blue) and two frame clusters (green and yellow). The kinematic relationship between the wheel and the frame is correctly identified as revolute (pink).

Results for the Box Experiment				
Parts	Confidence scores for Relationships			
	Rigid	Prismatic	Revolute	Disc
Wheel 1 - Wheel 2	**1.000**	0.000	0.000	0.000
Frame 1 - Frame 2	**0.996**	0.004	0.000	0.000
Background - Wheel	0.000	0.000	0.008	**0.992**
Background - Frame	0.000	0.000	0.125	**0.875**
Wheel - Frame	0.001	0.000	**0.999**	0.000

Table 10.8. Detecting the kinematic structure of a tricycle. The revolute joint connecting the wheel to the frame is detected with a very high confidence value of 99.9%. The parts of the tricycle are also separated from the background (not showed in the image). For clarity, the table groups together the two parts of the wheel and the two parts of the frame (after showing that they are declared as rigidly connected).

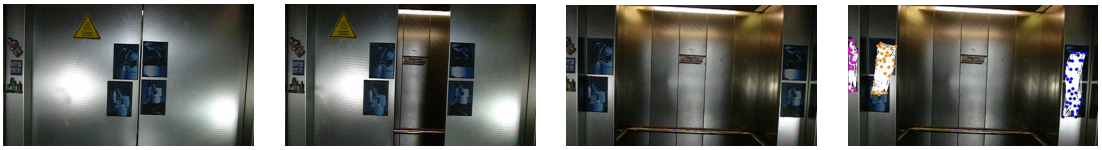


Figure 10.19. Identifying rigid bodies: elevator. Left to Right: The first column shows the elevator before the interaction; the second column shows the interaction itself; the third column shows the elevator after the interaction; and the fourth column shows the results of segmenting the graph of tracked features into clusters of features, where each cluster is associated with a different rigid body.

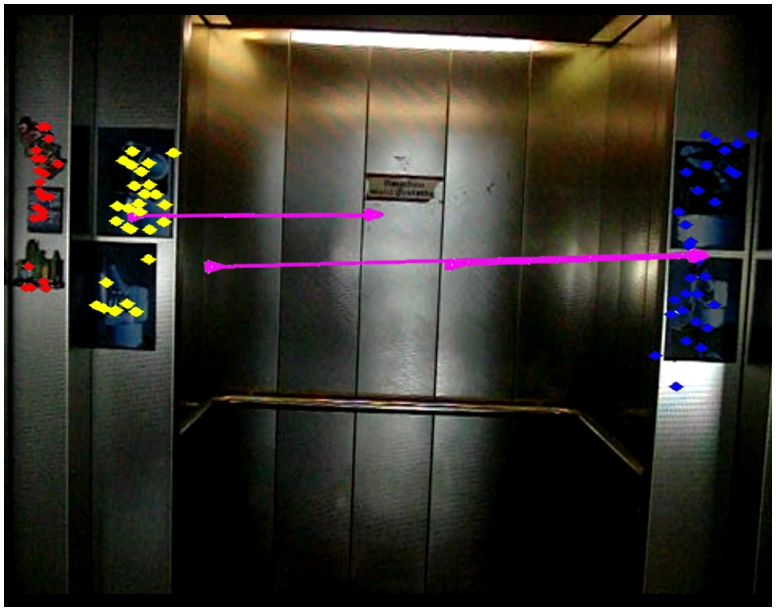


Figure 10.20. Detecting the kinematic structure of an elevator. Three clusters are detected; the static frame of the elevator (red), the right door (blue) and the left door (yellow). The prismatic joints between the doors and frame are marked in pink.

Results for the Box Experiment				
Parts	Confidence scores for Relationships			
	Rigid	Prismatic	Revolute	Disc
Left Door-Right Door	0.000	**0.995**	0.005	0.001
Left Door-Background	0.000	**0.994**	0.000	0.006
Background-Right Door	0.000	**0.955**	0.000	0.045

Table 10.9. Detecting the kinematic structure of an elevator. Three prismatic joints are found. The first is between the two doors. The second and third prismatic joints are between each door and the frame of the elevator. The confidence value for all three joints is 95% or higher.

Figure 10.21. Identifying rigid bodies: set of drawers. Left to Right: The first column shows the drawers before the interaction; the second column shows the interaction itself; the third column shows the drawers after the interaction; and the fourth column shows the results of segmenting the graph of tracked features into clusters of features, where each cluster is associated with a different rigid body.

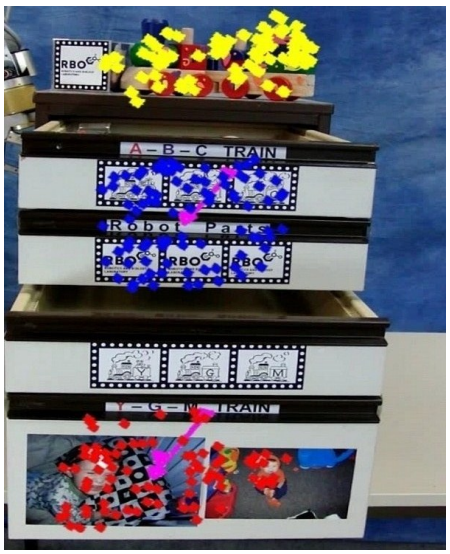


Figure 10.22. Detecting the kinematic structure of a set of drawers. Three clusters are detected; the cabinet's top (yellow), the top drawer (blue), and the bottom drawer (red). The three detected prismatic joints are marked in pink.

Results for the Box Experiment				
Parts	Confidence scores for Relationships			
	Rigid	Prismatic	Revolute	Disc
Top Drawer-Bottom Drawer	0.000	**0.969**	0.022	0.009
Top of Cabinet-Top Drawer	0.000	**0.996**	0.002	0.002
Top of Cabinet-Bottom Drawer	0.000	**0.975**	0.014	0.011

Table 10.10. Detecting the kinematic structure of a set of drawers. Three prismatic joints are found. The first is between the two drawers. The second and third prismatic joints are between each drawer and the top part of the cabinet. The confidence value for all three joints is 97% or higher.

10.3 Limitations and Failures

In the process of developing our algorithm we encountered two main reasons for failure: insufficient motion and insufficient texture. Both can effect the performance of our algorithm and in some cases cause it to fail.

The first potential cause of failure is too little motion. This occurred for example in the refrigerator experiment. The result of too little motion is that structure from motion is less reliable. In the refrigerator experiment, the result was that the correct kinematic structure was detected, but the position of the joint was estimated with an offset. Although too little motion can result in failure, we do not consider this to be a significant limitation of our approach. This is because the motion is generated by the robot using interactive perception. Indeed, too little motion will result in a less reliable kinematic model, and as soon as the robot attempts to use this model for manipulation it is likely to fail. The failure itself, however, is bound to provide additional motion, which enables the robot to improve its model.

A more significant limitation of our approach is its dependency on the presence of trackable visual features on each of the rigid bodies. Given our low-resolution camera, we had to add artificial features in 7 of our 10 experiments (e.g. place a photo on the feature-less refrigerator door). When features cannot be tracked reliably the performance of our three-dimensional reconstruction degrades. A more severe case is when no feature or very few unreliable features can be tracked. We propose to remedy this limitation by using a foveated or higher-resolution vision system, which would reduce the need for artificial features and increase the fidelity of feature tracking. It remains for future work to explore new types of visual feature that do not depend so strongly on the presence of texture.

In the current implementation, there is another important limitation: features that move out of the field of view of the robot are discarded. This means that the robot only models what it can see. During the above experiments the camera was always

positioned far enough from the object so that it would not move out of view. Future work, however, should address this limitation. This will allow for more complete shape models. Also, it will allow the robot to look more closely at objects, which means that weaker texture could be extracted, and therefore more reliable feature tracking will be achieved.

10.4 Summary

In all experiments, the proposed perceptual skill detected, segmented, and tracked all rigid bodies containing a sufficient number of visual features. The correct kinematic structure was found in 10 out of 10 experiments. Our skill detected the position and orientation of the joint correctly in 30 out of 31 cases. The one exception is the refrigerator experiment, in which the type of joint and the orientation of the axis of rotation are correct, but the position of the axis is offset. We performed the experiments under uncontrolled lighting conditions, different camera positions and orientations, and for different initial poses of the objects. The demonstrated robustness and effectiveness of the proposed perceptual skill provides evidence that it is suitable for manipulation in unstructured environments.

For our experiments, we do not have available the ground truth of the kinematic models in the scene. Therefore, we must rely on visual inspection in order to judge the effectiveness of our skill. Ultimately, the proposed perceptual skill will be combined with manipulation skills. Then, we will be able to determine whether the accuracy of the extracted kinematic models is sufficient to guide manipulation planning and execution. We believe that the experimental results presented in this chapter provide evidence that this is the case.

The runtime of all three steps of our skill depends on the number of tracked features, the duration of the experiment (i.e. number of frames), and the number of rigid bodies in the scene. Our experimental scenes are typically composed of 2-5 rigid

bodies. Feature tracking usually detects and tracks between 200 and 500 features, and an experiment takes between 10 and 30 seconds (300 to 900 frames). The runtime of all three parts of the skill varied from 5 to 15 minutes. We believe that a significant improvement will be gained by optimizing the implementation. Further improvement could be achieved by parallelizing parts of the code.

This chapter completes the presentation of the proposed interactive perceptual skill for modeling the shape, motion, and kinematic structure of three-dimensional rigid articulated objects. Thus far, we have assumed that the interaction with the environment is scripted. This assumption limits the applicability of our perceptual skill to unstructured environments. In the next chapter, we will propose a learning framework that will enable our robot to autonomously plan its interactions with objects in order to discover their kinematic structure.

CHAPTER 11

GROUNDED RELATIONAL REINFORCEMENT LEARNING

Throughout this thesis, we have discussed our approach for modeling the shape and kinematic structure of articulated objects (bottom two components of Figure 5.1). This approach relies on our first two hypotheses for handling the complex state space of manipulation in unstructured environments. Our first hypothesis was that using task-specific representation, in our case: links and joints, restricts the high-dimensional state space to the task-relevant dimensions. Our second hypothesis was that leveraging the robot's embodiment through interaction with the environment reveals important task-relevant information that would otherwise remain hidden. In our case, this information is relative motion between rigid bodies.

Deciding how, where and in what order to interact with the environment requires that the robot generates a manipulation plan. In our discussion thus far, we have assumed that the robot's interactions with the environment are scripted. This was the case in all the experiments described in chapter 10. In contrast, in this chapter we present our first steps towards removing this assumption. To that end, we ask the following question: how can a robot interact with the environment to robustly and efficiently acquire the shape and structure of previously unseen objects?

One trivial answer to this question is through random interactions. With enough random interactions, and given the finite size and complexity of real-world objects, the robot is very likely to eventually discover all relevant information needed to form a reliable object model. However, spending significant amount of time on modeling

is impractical for most applications. We therefore require that the process of model acquisition is directed and efficient.

Instead of relying on random interactions, we propose a learning-based approach to manipulation. Thus, this chapter focuses on the third component of our proposed approach—the planner (top component of Figure 5.1). Our planner permits a robot to autonomously acquire manipulation expertise from its interactions with the environment. The resulting expertise enables the robot to select the most effective manipulation action based on partial state information.

Our learning-based approach to manipulation enables a robot to autonomously acquire manipulation strategies for efficient modeling of the kinematic structure of unknown articulated objects. The robot physically interacts with an object by pushing or pulling it, while observing the object's motion. As these interactions create a change in the configuration of the object, the robot incrementally discovers the object's intrinsic and extrinsic degrees of freedom. The robot learns to select interactions that are most likely to reveal the maximum information about the kinematic structure. The acquired manipulation knowledge substantially reduces the number of interactions required to obtain an accurate kinematic model. Furthermore, the manipulation knowledge acquired by modeling one object transfers to other objects, even if they have different kinematic structures.

Manipulation expertise is learned in a relational state representation. This representation is essential, as it renders learning tractable by collapsing a large regions of the state space onto a single, task-relevant, relational state. The symbolic representation is carefully grounded in the perceptual and interaction skills of the robot via the perceptual skill discussed in previous chapters. This grounding ensures that relationally learned knowledge remains applicable in the physical world.

We begin our discussion by introducing the relational representations of kinematic structures that forms the basis of our learning-based approach to manipulation (sec-

tion 11.1). Then, in section 11.2, we describe how this representation can be grounded using the perception and manipulation capabilities of the robot. We proceed to discuss the relational learning framework and how it can be grounded with respect to the relational representation (section 11.3). We demonstrate the effectiveness of our approach in manipulation experiments with articulated objects (section 11.4). Finally, we briefly discuss our ongoing and future work in section 11.5.

11.1 Relational Representation

To describe the state space associated with manipulating rigid articulated objects using a propositional representation, we would have to include a proposition for every object encountered by the agent. We would also have to include a proposition for every action applicable to this object. Gathering and generalizing manipulation expertise becomes impossible with this representation due to the combinatorial explosion of actions and states. A relational representation allows us to describe an infinite number of states and actions using a finite set of relations. It is thus critical to the success of our learning-based approach to manipulation.

Our relational representation leverages the following insight: an agent may encounter, for example, many types of scissors. These scissors may vary in color, shape, and size. All scissors, however, have the same kinematic structure. This kinematic structure can be captured by a single relational formula.

What object properties should our relational representation include? To represent the kinematic structure of an object, we must consider joint types (revolute, prismatic, or disconnected), link properties (e.g. color and size), and the kinematic relationships between links. Therefore, our relational representation uses the following predicates:

- Revolute Joint: $R(\cdot, \cdot, ...)$
- Prismatic Joint: $P(\cdot, \cdot, ...)$

- Disconnected: $D(\cdot, \cdot, ...)$

Figure 11.1. Two Examples of Kinematic Structures: scissors with a single revolute joint and a wooden toy with a prismatic joint and two revolute joints.

For simplicity in capturing branching kinematic structures, our predicates are n-ary, with $n \geq 2$. The rigid body (link) passed as the first argument to the relation is the one in relationship with all other arguments. Thus, $R(x, y, z)$ is equivalent to $R(x, y) \wedge R(x, z)$. Figure 11.1 shows two examples of kinematic structures. The scissors have a single revolute degree of freedom and the wooden toy is a serial kinematic chain with a prismatic joint (on the left of the figure) and two revolute joints. Our relational representation enables us to describe the kinematic structure of the scissors as:

$$D(l_b, R(l_1, l_2)),$$

where l_1 and l_2 represent the two links of the scissors and l_b is a disconnected background link. Similarly, the kinematic structure of the wooden toy can be represented as

$$D(l_b, R(l_4, R(l_3, P(l_1, l_2)))).$$

This representation is not unique. The wooden toy could also be represented as:

$$D(P(l_4, R(R(l_1, l_2), l_3)), l_b).$$

The specific representation used by the agent depends on the order of discovery of the joints. The most deeply nested relation is the one discovered first.

Our representation also allows for branching structures and kinematic loops. For example, we describe a kinematic loop with 5 revolute joints by:

$$D(l_b, R(l_1, R(l_5, R(l_4, R(l_3, R(l_2, l_1)))))).$$

The representation of links can also be extended to an m-ary relation: $L(\cdot)$, where $m \geq 1$. This representation supports a variety of link properties such as size, color, and composition. In the work we describe here, we limit ourselves to a single link property: size. The extension to an arbitrary number of link properties, however, is straightforward. Using the extended link representation, the wooden toy can be described by:

$$D(l_b, R(L(s, f_4), R(L(s, f_3), P(L(s, f_1), L(s, f_2))))),$$

where s stands for the property *size=small* and f_i spatially identifies link i in the physical world (see section 11.2).

To complete our relational representation, we must also provide a representation for the actions performed by the agent. The actions that we allow are limited to

pushing or pulling a link. Each action can be applied either along the major axes of the link or at a forty-five degree angle to the major axes. An action is represented as $A(L(\cdot), \alpha)$, where $L(\cdot)$ represents a link and α is an atom describing one of the six possible actions.

The relational representation of links, joints, and actions allows us to reason and learn about objects based on their kinematic structure. The experience that an agent may acquire by manipulating scissors can be applied to all other scissors. The properties of an object that affect its manipulation behavior may not be limited to its kinematic structure. The relational representation of a link can be extended to include other relevant properties. With additional link properties, our agent will be able to distinguish between different objects with the same kinematic structure. The advantage of this approach is that it ignores information about the physical manifestation of objects (i.e. position, orientation, and configuration), as well as other properties irrelevant for manipulation. As a result, we achieve a significant reduction in the dimensionality of the state space, rendering the learning problem tractable.

11.2 Grounding the Relational Representation

The relational representation described in the previous section can only support the learning of manipulation knowledge if it is grounded in the physical capabilities of the robot. Grounding bridges between the symbols of our representation and the physical, continuous world [59]. It ensures that we can symbolically interpret the observations made by the robot with regards to its interactions with the world. At the same time, grounding ensures that the resulting symbolic manipulation knowledge maintains its relevance and predictive power for the robot's real-world interactions.

To ground our relational representation, we bind the relations $R(\cdot, \cdot)$, $P(\cdot, \cdot)$, and $D(\cdot, \cdot)$ as well as the links' properties to real-world perceptual capabilities of the

robot. These perceptual capabilities enable a robot to model rigid articulated objects (chapter 6 describes the skill for planar objects, and chapters 7 - 10 describe the skill for general three-dimensional objects). The robot's perceptual capabilities provide adequate grounding for our relational representation of links and their kinematic relationship.

11.3 Learning Manipulation Expertise

With the grounded relational representation of states (links and joints) and actions developed in the previous two sections, we can now cast the problem of incremental acquisition of manipulation knowledge as a relational reinforcement learning problem [42, 129, 136].

In reinforcement learning, an agent learns a policy for performing a task. This policy tells the agent which action to perform in a given state. The process of acquiring the policy is incremental; the agent learns the policy through a sequence of interactions with the environment. This incremental process is identical to the interactive perceptual skill that we developed throughout this thesis. For the perceptual skill, an experiment consisted of a sequence of actions. With every action, the robot may or may not discover new information. To formulate this process as a reinforcement learning problem, we simply have to attach a reward for every degree of freedom and every link property discovered by the robot. We expect the robot to incorporate new experiences into its policy, improving it over time. If learning succeeds, our robot will have acquired an effective policy for modeling the kinematic structure of novel rigid articulated objects.

11.3.1 Problem Definition

Relational Markov Decision Process (RMDP) provides opportunities for abstraction that are due to the structured form of the states and actions. Therefore, we

formalize our problem as an RMDP [136] and then apply Q-learning [138] to find an optimal policy:

We define $P = \{p_1/\alpha_1, ..., p_n/\alpha_n\}$ to be the set of predicates with their arities, $C = \{c_1, ..., c_k\}$ to be a set of constants, and $A' = \{a_1/\alpha_1, ..., a_m/\alpha_m\}$ to be the set of actions with their arities. Let S' be the set of all ground atoms that can be constructed from P and C, and let A be the set of all ground atoms over A' and C.

An RMDP is a tuple $M =< S, A, T, R >$, where:

- $S \subseteq S'$ is the set of possible states
- A is the set of available actions
- $T : S \times A \rightarrow \Pi(S)$ is a probabilistic transition function
- $R : S \times A \times S \rightarrow \mathbb{R}$ is a reward function

Relational MDPs extend MDPs by modifying the definition of states S and actions A (described in Section 11.1). The transition function T and the reward function R remain unchanged. The transition function T captures the state transitions that occur in the physical world when an action is applied. We do not require an explicit representation of the transition function. Instead, we rely on the real world and on our perceptual skills to provide us with the new state after applying a certain action. The reward function R returns the number of links, link properties, and joints that were discovered by performing an action in a particular state.

With this, the definition of our learning task as an RMDP is complete. It remains to provide the agent with a policy π for manipulating rigid articulated objects in unstructured environments. We now describe how π can be learned using Q-learning.

11.3.2 Q-Learning

Q-learning is a reinforcement learning technique [138]. It determines a policy $\pi : S \rightarrow A$ for selecting actions based on the current state. To determine this

policy, the agent must learn an approximation Q^* to the optimal Q-value function $Q^* : S \times A \to \mathbb{R}$. The agent learns Q^* by performing a series of experiments, each of which reveals how much reward a particular action can obtain in a particular state. The Q-value function accumulates information about the total expected reward for an entire trial. The policy defined by the Q-value function is given by:

$$\pi(s) = \arg\max_a Q^*(s, a)$$

The Q^* function is continuously updated by the agent, as it interacts with the environment and receives rewards for performing specific actions in specific states. Updating is done as follows:

$$Q^*(s_t, a_t) := (1 - \alpha)\, Q^*(s_t, a_t) + \alpha \left(r_{r+1} + \gamma \max_a Q^*(s_{t+1}, a) \right),$$

where α is the learning rate, γ is the discount factor, and r_t is the reward received at time t.

The **learning rate** α determines the extent to which newly acquired information overrides old information. A factor of $\alpha = 0$ prevents the agent from learning anything new, whereas $\alpha = 1$ makes the agent "forgetful" (i.e. it considers only the most recent information).

The **discount factor** γ determines the importance of future rewards. A factor of $\gamma = 0$ results in an "opportunistic" agent, only considering current rewards. A discount factor approaching $\gamma = 1$ results in an agent that strives for a long-term high reward. If γ exceeds 1, the Q-value will diverge.

11.3.3 Representation of Q-Value Function

Our representation of the Q-value function is instance-based [39]. That is, we store a set of instances of state-action pairs with their corresponding values. We update

this set by adding state-action pairs and changing the corresponding values during learning. Our agent stores each of its experiences as a tuple $< s_i, a_i, r_i >$ of state s_i, action a_i, and the reward r_i obtained when performing the action a_i in state s_i. In storing the experiences of the agent, the advantages of our relational representation become clear. Because our representation is relational, states and actions are stored un-instantiated (i.e. without reference to their spatial identity in the real-world). As a result, every experience remains applicable to a possibly infinite number of similar situations.

Given the current state, the agent has to estimate Q-values for actions based on its experience. This is straight forward when the agent has previously visited the current state. To estimate the Q-value of a previously unvisited state, the agent must seek for relevant prior experience, and apply it to the new situation. To identify the experience most relevant to the current state, we need to determine the similarity between states. In our case, similarity is affected by the kinematic structure and the link properties of the explored object. Neither of these aspects have to match perfectly for the robot to retrieve relevant experience. We now describe how the agent can identify similarity in link properties and in kinematic structure.

11.3.3.1 Finding Similarity between Link Properties

Let us assume that the robot at time t has uncovered the existence of three links, connected into a serial chain by revolute joints. The links' sizes are: small, large, and large. The corresponding relational state description is:

$$s_t = R(R(L(s, f_1), L(l, f_2)), L(l, f_3)),$$

For simplicity, let us assume that the Q-value function representation (the agent's experience table) contains a single experience entry:

$$(s, a, r) = (R(R(L(s, v_1), L(s, v_2)), L(s, v_3)), A(v_3, 45^\circ), 2.1)\,.$$

The state s represents a serial chain with two revolute joints and three small links. Note that the Q-value function stores a variable v_i for the spatial identification of each link. This variable can now be instantiated by unifying the agent's experience state s with the current state representation s_t. In this case, unification fails, because the link sizes do not match between the current and stored states. Fortunately, this experience is not useless. We can still retrieve somewhat less relevant experience by ignoring the unmatched link property. Now, unification leads to a binding of $v_3 \leftarrow f_3$, and the action is instantiated to $A(f_3, 45^\circ)$. Relying on its experience, the agent will now decide to push on the link described by f_3 at a 45° angle relative to the principal axes of the link.

This example illustrates the principles behind our process of unification. It begins with an attempt to find a perfect match. If such a match is not found, we incrementally ignore link properties (the least discriminative properties first), until unification becomes possible. Determining an order over link properties can be predefined; we expect that, in future work, ordering will be based on the robot's experience.

11.3.3.2 Finding Similarity between Kinematic Structures

Our relational representation of a kinematic structure is not necessarily unique. For example, the state s_t, from section 11.3.3.1 is equivalent to

$$R(L(s, f_1), R(L(l, f_2), L(l, f_3))).$$

To retrieve relevant experience in the presence of this representational ambiguity, we require a mapping from one structure onto another. Here too, we would like this mapping to enable partial matchings. For example, for a state $s_t' = R(L(l, f_1), L(l, f_2))$ we would like to be able to leverage our experience by realizing that $L(l, f_1)$ in s_t'

could represent $R(L(l, f_1), L(s, f_2))$ in s_t before the additional degree of freedom was discovered.

To compute this mapping between related kinematic structures, we represent a relational state with an undirected graph $G = (V, E)$. A vertex $v \in V$ corresponds to a link and an edge $e \in E$ represents the kinematic relationship between two links, and is labeled accordingly as either prismatic, revolute or disconnected. To identify a partial or complete match between kinematic structures we have to solve a graph-matching problem. Given two graphs G_t and $G_t^{'}$ corresponding to the current state s_t and a memorized state $s_t^{'}$, we compute subgraph isomorphism to find exact structural matches. This process requires that all matched link properties are identical. If the relational description of the links varies, we can compute sub-graph monomorphism to identify partial matches. In contrast to subgraph isomorphism, which is a bijection, subgraph monomorphism is an injection, thus the match is one-to-one but not onto.

When the agent searches its memory table for appropriate experiences, it may retrieve multiple matches. Only the most closely related match will be returned. In some cases, the agent will have no relevant experience to exploit. This can happen when a similar kinematic structure has not yet been encountered or when similarity exists, but the stored action cannot be instantiated (i.e. is not part of the partial match). When no relevant experience can be retrieved, the agent is forced to pick a random action. The result of this action forms a new experience.

Each time the agent performs an action, either random or based on its prior experiences, it receives a reward. We store (or update) this experience in the instance-based Q-value function. If an exact graph match is found between the current state and the agent's experience (graph isomorphism), we update the existing memory entry with the new experience. If a partial match or no match is found, we add a new instance to the representation of the Q-value function.

Both subgraph isomorphism and subgraph monomorphism are NP-complete problems. Real-world objects, however, typically possess a small number of links. As a result, we only have to consider graph matching for small graphs, and the theoretical computational complexity does not impose any practical limitations on our approach. In our implementation, we rely on an efficient algorithm for subgraph matching, which is specialized for the case of small graphs [28].

Although the complexity of graph matching does not limit our approach, the size of the agent's experience table does. Several methods have been proposed to consolidate the experiences of the agent [39]. Future work should look into the automatic generation of general manipulation rules from the agent's experiences. Such rules will lead to a significant reduction in the memory requirements of our instance-based representation for the Q-value function.

11.3.4 Selecting Optimal Actions

Given unlimited time for exploration, our agent can gather enough manipulation experiences to learn an optimal policy. To comply with the time constraints imposed by manipulation in unstructured environments, our agent must be able to discover an optimal (or nearly optimal) policy quickly. To that end, it must balance between exploration and exploitation. Exploration refers to the execution of an action to improve the Q-value function's estimate of the associated reward. In other words, when our agent explores, it either chooses an action it has never tried before, or the action which outcome the agent is most uncertain about. Exploitation, in contrast, refers to action selection based on maximizing reward. The balance between exploration and exploitation is important. If the agent explores too much, it will learn slowly. If it exploits too early, it will perform poorly because it has not gathered enough experience.

To decide if a new action should be executed, we compute the fraction ϕ of actions for which the robot already has gathered experience. If a number drawn uniformly at random from the interval $[0, 1]$ is smaller than $e^{-\beta\phi}$, the robot performs exploration. In our experiments, β is set to 2. Otherwise it selects one of the actions associated with state s. If the agent is to retrieve an action based on its experience, we use Interval Estimation (IE) [75]. Intuitively, IE picks the action that has the highest potential to perform well. Thus, IE also balances between exploration and exploitation.

11.4 Experiments With Planar Objects

Our experimental evaluation requires a large number of experiments. Because our objective is to demonstrate that manipulation knowledge can be learned, we performed experiments in a simulated environment. Further, the experiments that we describe in this section are limited to planar objects. That is, they rely on the perceptual skill for modeling planar objects presented in chapter 6. Due to the robustness of this perceptual skill, and because force guided pushing/pulling requires only a simple controller, we believe that our results will remain valid in real-world experiments.

To evaluate the effectiveness of our learning-based approach to manipulation in unstructured environments, we perform two types of experiments. First, we show that manipulation knowledge can be gathered from experience. And second, we show that the acquired experiences transfer, and can be applied to previously unseen objects.

11.4.1 Experimental Setup

We perform experiments in a simulated environment. This environment is based on the Open Dynamics Engine (ODE), a popular dynamics simulator. ODE is an open source, high performance library for simulating the dynamics of rigid bodies. It features various joint types and integrated collision detection. It simulates gravity, various sources of friction, and allows for some non-determinacy.

In our experiments, a robot interacts with an articulated planar object to extract its kinematic structure. Example objects are shown in Figures 11.2 - 11.5 and 11.6 - 11.9. Links (rigid bodies) are shown in blue. Revolute joints are represented by red cylinders, and prismatic joints are illustrated as green boxes.

An experiment consists of a sequence of trials. In each trial, the object that is being explored by the agent is initialized at a random configuration. A trial is composed of a number of steps. In each step, the robot applies a pushing or pulling action to the articulated object. The trial ends when an external observer signals that the robot has obtained the correct kinematic structure of the object. In each step of every trial the robot accumulates manipulation experiences that improve its future performance. The number of steps per trial measures the number of interactions necessary to discover the correct kinematic structure of the articulated object. It therefore measures the efficiency with which the robot accomplishes the task. Each step of a trial can be divided into three phases:

1. The robot selects an action and a link (rigid body) for interaction. The action is instantiated using the current state and the experience is stored in the representation of the Q-value function.

2. The selected action is applied to the link, and the resulting object motion is simulated. The trajectories of the visual features tracked by the perception skill are reported to the robot.

3. The robot analyzes the motion of visual features and determines the kinematic properties of the rigid bodies observed so far. These properties are then incorporated into the robot's current state representation.

For each trial, we report the average number of interactions used to discover the correct kinematic structure. This average is computed over 10 independent experiments.

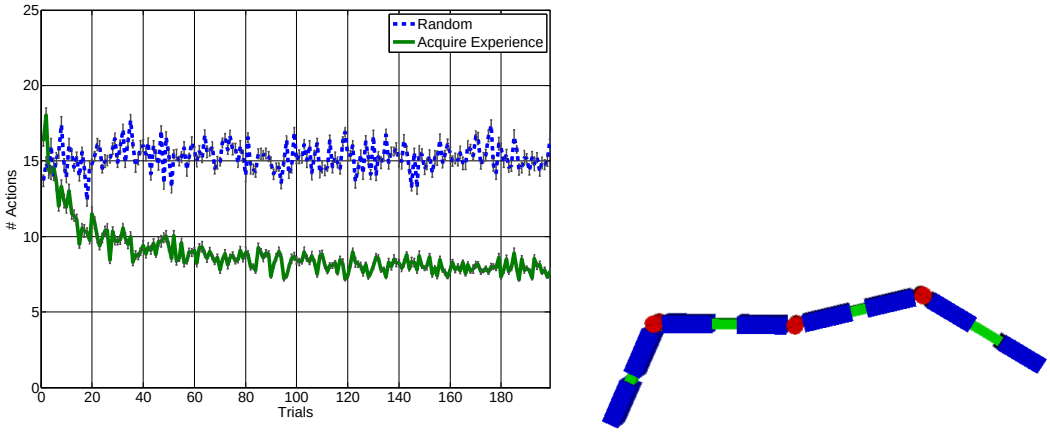
In our simulation experiments, when the correct kinematic structure is discovered, an external supervisor issues a reward signal to terminate the trial. This external supervisor is not a limitation. In real-world experiments, a robot can decide to perform manipulation based on (potentially) incomplete information. If new kinematic information is discovered during manipulation, the robot simply updates its kinematic model and revises its manipulation strategy accordingly. In other words, learning manipulation knowledge never stops; every interaction with the environment is monitored and informs our Q-value function. In real-world experiments, however, we will have to penalize the agent for every action, encouraging it to perform short action sequences.

11.4.2 Gathering Manipulation Knowledge

Our first type of experiments shows that manipulation knowledge can be gathered from experience. To demonstrate the effectiveness of learning , we observe the number of actions required to discover a kinematic structure. We compare the performance of the proposed grounded relational reinforcement learning approach to a random action selection strategy. Figures 11.2, 11.3, 11.4 and 11.5 show the objects presented to the robot, as well as the results (learning curves) of four experiments.

In the first experiment, we present the robot with an object with seven degrees of freedom and eight links. The resulting learning curve is shown in figure 11.2. Random action selection, as to be expected, does not improve its performance with additional trials. In contrast, action selection based on the proposed relational reinforcement learning approach results in a substantial reduction of the number of actions required to correctly identify the kinematic structure. This improvement already becomes apparent after about 10 trials. Using the learning-based strategy, an average of 8 pushing actions is required to extract the correct kinematic model of the object.

Compared to the approximately 16 pushing actions required with random action selection, learning achieves an improvement of about 50%.



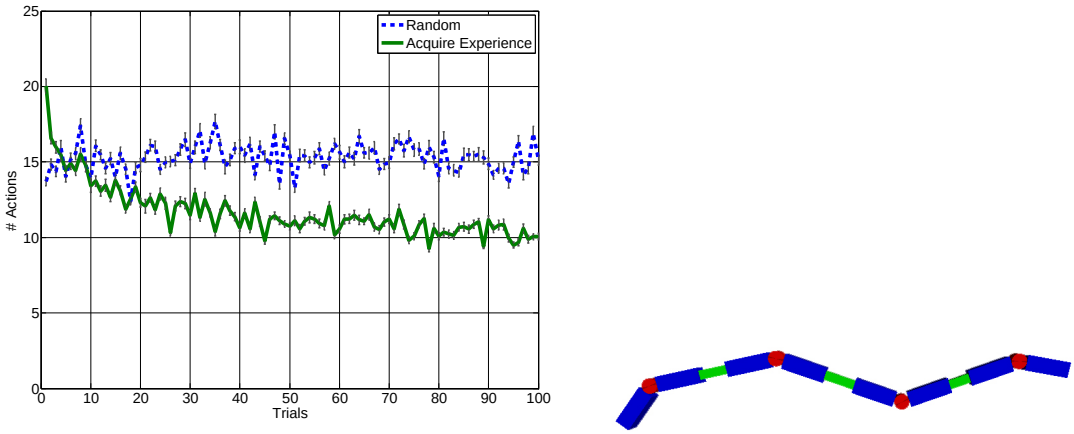
(a) Performance compared to random

(b) Articulated object

Figure 11.2. Experiments with a planar kinematic structure (PRPRPRP). The object possesses seven degrees of freedom (R = revolute, P = prismatic). The learning curve for our learning-based approach to manipulation (green solid line) converges to eight required actions with a decreasing variance, representing an improvement of 50% over the random strategy (blue dashed line).

In the second experiment, we present the robot with another object with seven degrees of freedom and eight links. The resulting learning curve is shown in figure 11.3. The improvement achieved by our learning approach becomes apparent after about 20 trials. Using the learning-based strategy, an average of 10 pushing actions is required to extract the correct kinematic model of the object. Compared to the approximately 15 pushing actions required with random action selection, learning achieves an improvement of about 30%.

In the third experiment, we present the robot with an object with eight degrees of freedom and nine links. The resulting learning curve is shown in figure 11.4. The learning-based strategy requires an average of 8 pushes, whereas the random strategy uses approximately 20 pushing actions. Learning achieves an improvement of about 60%.



(a) Performance compared to random

(b) Articulated object

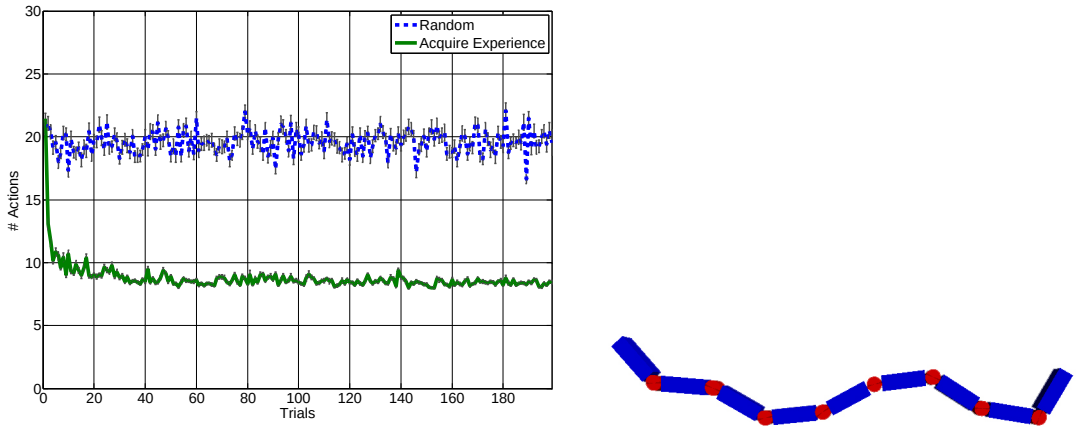
Figure 11.3. Experiments with a planar kinematic structure (RPRPRPR). The object possesses seven degrees of freedom (R = revolute, P = prismatic). The learning curve for our learning-based approach to manipulation (green solid line) converges to ten required actions with a decreasing variance, representing an improvement of 30% over the random strategy (blue dashed line).

In the fourth experiment, we present the robot with an object with nine degrees of freedom and ten links. The resulting learning curve is shown in figure 11.5. The learning-based strategy requires an average of 10 pushes, whereas the random strategy uses approximately 22 pushing actions. Learning achieves an improvement of about 60%.

These four experiments demonstrate that our approach to manipulation enables robots to gather manipulation knowledge and to apply this knowledge to improve manipulation performance.

11.4.3 Transferring Manipulation Knowledge

Our second type of experiments shows that manipulation experience acquired with one object transfers to other objects. To demonstrate the effectiveness of knowledge transfer, we again observe the number of actions required to discover a kinematic structure. We compare the performance of the proposed grounded relational reinforce-



(a) Performance compared to random

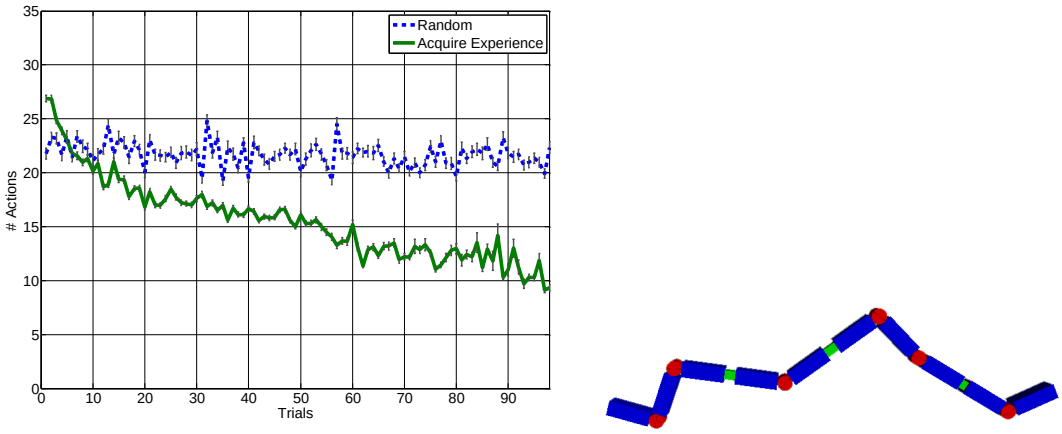
(b) Articulated object

Figure 11.4. Experiments with a planar kinematic structure (RRRRRRRR). The object possesses eight degrees of freedom (R = revolute). The learning curve for our learning-based approach to manipulation (green solid line) converges to eight required actions with a decreasing variance, representing an improvement of 60% over the random strategy (blue dashed line).

ment learning approach with and without prior experience. Figures 11.6, 11.7, 11.8 and 11.9 show the objects presented to the robot, as well as the results (learning curves) of four experiments.

In the first transfer experiment, the robot gathers experience with an articulated object with 7 degrees of freedom (see Figure 11.6). After 50 trials, the robot is given a simpler object with only 5 degrees of freedom. The simpler structure is a sub-structure of the more complex one. We compare the robot's performance with that of a robot without prior experience. The robot with prior experience consistently outperforms the robot without experience. In the first trial, which is the most important for real-world manipulation, the experienced robot requires 50% as many pushes. Over the following five trials, the performance improvement is approximately 20%. In trials 5 to 20, the performance improvement is much smaller.

In the second experiment, the robot learns to manipulate a complex articulated object with 5 revolute joints (see Figure 11.7). After 50 trials, the robot is given a



(a) Performance compared to random

(b) Articulated object

Figure 11.5. Experiments with a planar kinematic structure (RRPRPRRPR). The object possesses nine degrees of freedom (R = revolute, P = prismatic). The learning curve for our learning-based approach to manipulation (green solid line) converges to ten required actions with a decreasing variance, representing an improvement of 50% over the random strategy (blue dashed line).

slightly simpler structure that only possesses four revolute joints. Again, the simpler structure is a sub-structure of the more complex one. We compare the robot's performance after these initial 50 trials to the performance of a robot without prior experience. The experienced robot achieves convergence almost immediately. This corresponds to a performance improvement of about 50% in the first trial, relative to the robot without experience. After about 14 trials, both robots converge to approximately the same performance. This is to be expected for simple structures, exclusively consisting of revolute joints.

The third experiment complements the second experiment. Here, the robot learns to manipulate an articulated object with 4 revolute degrees of freedom (see Figure 11.8). After 50 trials, the robot is given a structure with an additional revolute joint (five altogether). We compare the robot's performance after these initial 50 trials to another robot's performance without prior experience. Again, experience

results in an improved performance in the first few trials (about 30%). After about 8 trials, both robots converge towards the same number of interactions.

The fourth experiment (see Figure 11.8) takes a step towards long term learning of manipulation expertise. Here, we compare the performance of three robots: The first has no prior knowledge, the second's prior knowledge is based on interactions with one object, and the third's prior knowledge relies on interactions with two objects. The results show the advantage that the more experienced robots have in the first few trials. More importantly, this experiment suggests that the more experience a robot gathers, the more it can transfer to new situations.

Our experimental results provide strong evidence that learning from past experience can significantly boost the robot's manipulation performance. Learning enables a robot to autonomously acquire manipulation expertise by interacting with the environment. Our results show that this expertise transfers across different instances of the manipulation task and substantially improves manipulation performance. Learning and generalization of manipulation knowledge become possible due to our relational representation of states and actions. This representation collapses the otherwise intractable state space and renders reinforcement learning feasible. We believe that the effectiveness of our approach is due to the proper, task-specific grounding of our relational representation in the robot's perceptual and interactive capabilities.

11.5 Future Work

There are many possible directions for extending this relational reinforcement learning framework. Our goal here was only to provide a proof of concept that learning manipulation expertise is possible. We now briefly discuss two interesting directions for future work. First, we discuss learning for more sophisticated mechanisms which requires planning to consider delayed reward. And second, we present our ongoing work on learning to manipulate three-dimensional articulated objects.

11.5.1 Planning With Delayed Reward

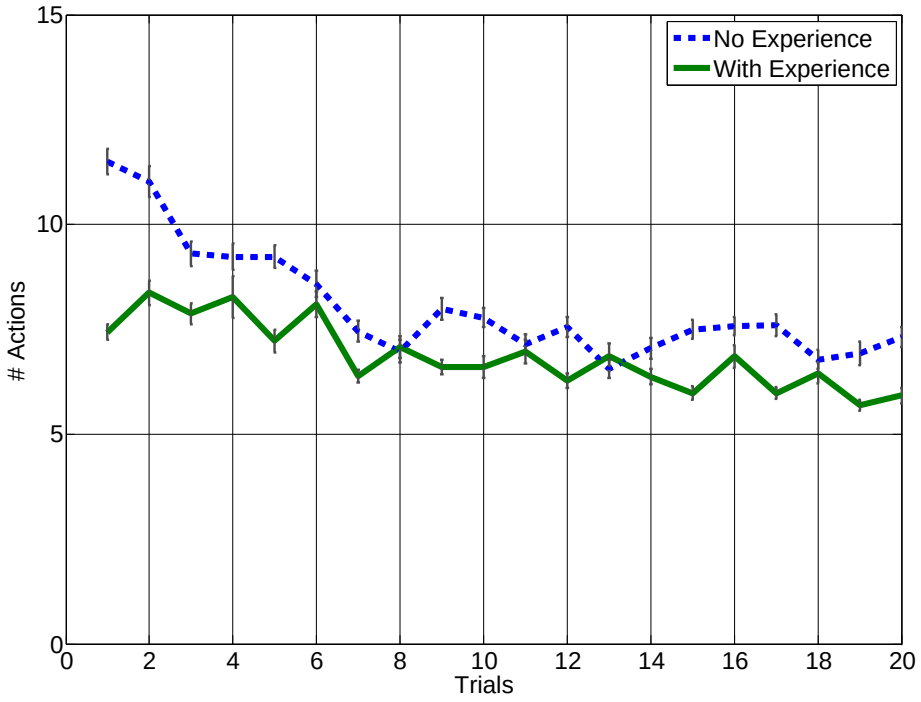
In our current implementation, the agent decides on the best action to take by only considering immediate reward—a one step horizon. This is reasonable in the current setup. However, to handle more complex mechanisms, this approach must be extended to consider delayed reward. For example, in order to model the hinge of a door, an agent may have to first manipulate the doorknob and then push-open the door. If knowledge of the doorknob is already available, manipulating it will not generate any reward. In our current implementation, the agent will have no reason to take this action, and will therefore never discover the hinge. Thus, future work should extend our implementation to have a longer horizon.

11.5.2 Experiments With 3-D Objects

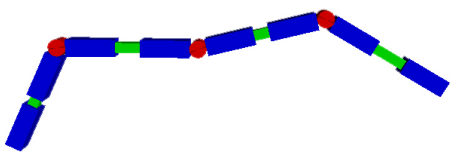
We are currently working on the development of a new simulation environment for three-dimensional objects. This work is still in its early stages. Our primary objective is to replicate the success of learning for planar objects in the more general case of 3-D articulated objects. An example of the type of three-dimensional objects we plan to explore in the new simulation environment is shown in figure 11.10(a). In this simulation environment, we also intend to explore the relevance of a variety of object properties, such as size, color, texture, the existence of parallel lines, or a sharp change in contrast.

In designing the new simulation environment, we also consider a new and exciting research direction: comparing human and machine object exploration. We are interested in analyzing how intelligent agents balance exploration vs. exploitation when studying a new object. Our new simulator includes a haptic interface (see Figure 11.10(b)), enabling human subjects and the simulated agent to interact with the same kinematic structures. We would like to use it to answer questions about

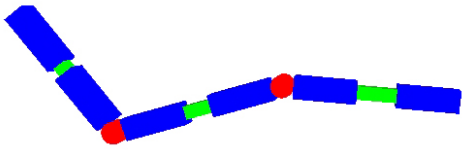
information reduction and representation during object exploration. These answers are of interest both to the robotics and psychology communities.



(a) Performance with and without prior experience

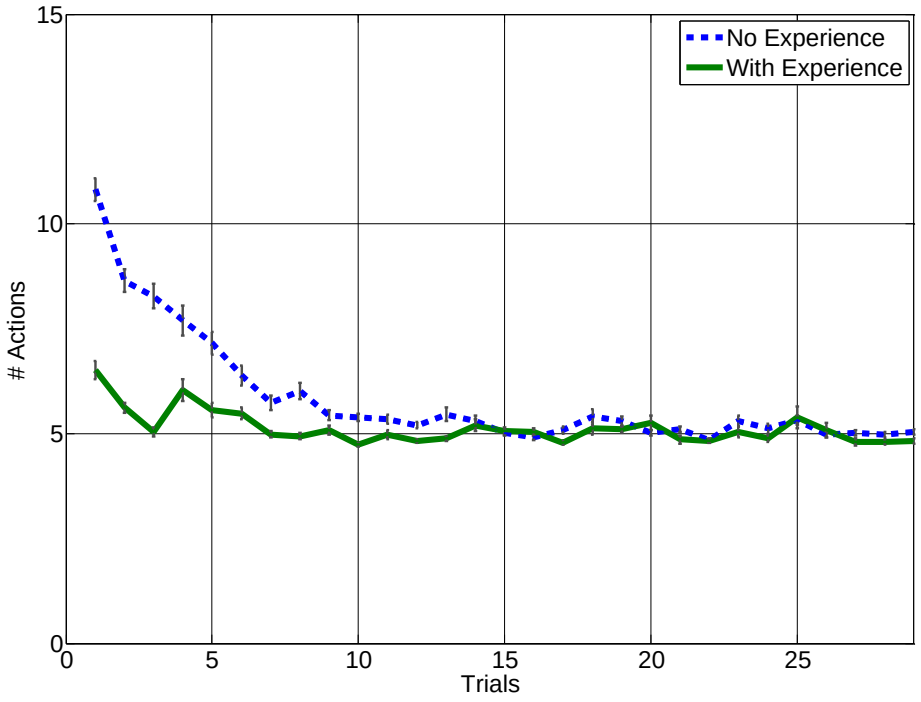


(b) Object used to gather experience

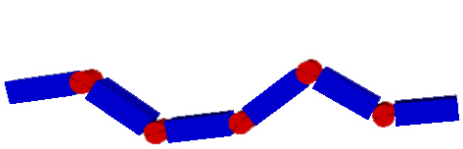


(c) Object used to measure transfer

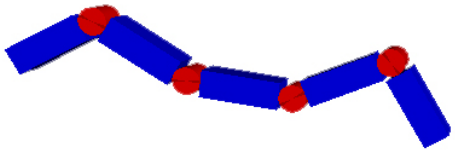
Figure 11.6. Experimental validation of knowledge transfer (PRPRPRP to PRPRP). We compare a robot with experience manipulating the PRPRPRP object (bottom left, solid green line) to an inexperienced robot (dashed blue line). Both robots learn to acquire the kinematic structure of a less complex object (PRPRP, bottom right). Experience improves performance by about 40% on the first trial.



(a) Performance with and without prior experience

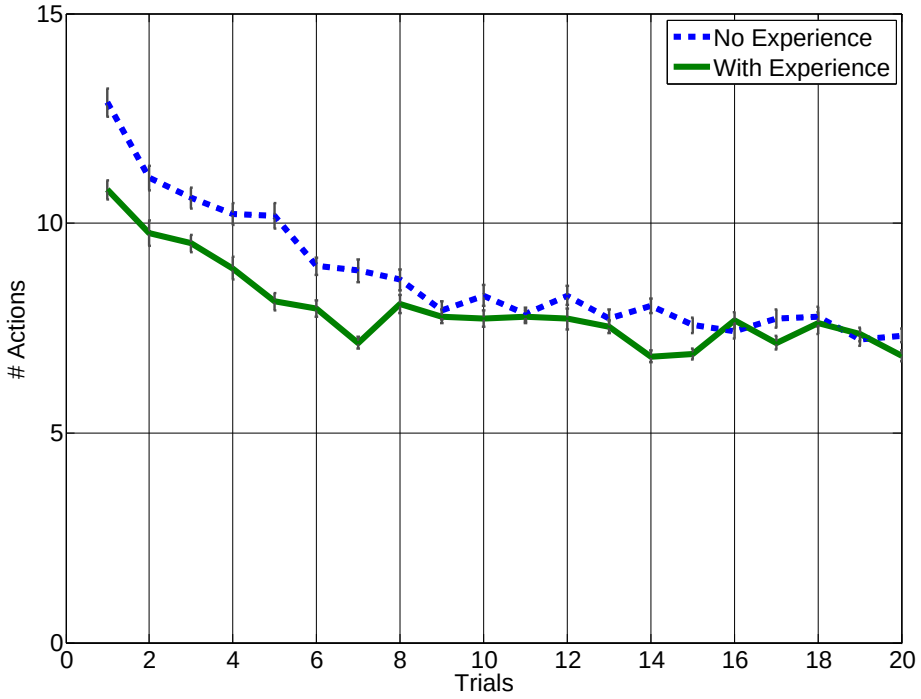


(b) Object used to gather experience

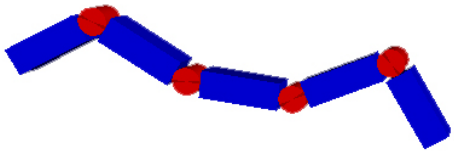


(c) Object used to measure transfer

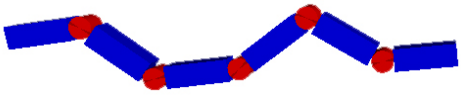
Figure 11.7. Experimental validation of knowledge transfer (RRRRR to RRRR). We compare a robot with experience manipulating the RRRRR object (bottom left, solid green line) to an inexperienced robot (dashed blue line). Both robots learn to acquire the kinematic structure of a simpler object (RRRR, bottom right). Experience improves performance by about 50% on the first trial. The performance of the unexperienced robot converges after about 15 trials.



(a) Performance with and without prior experience

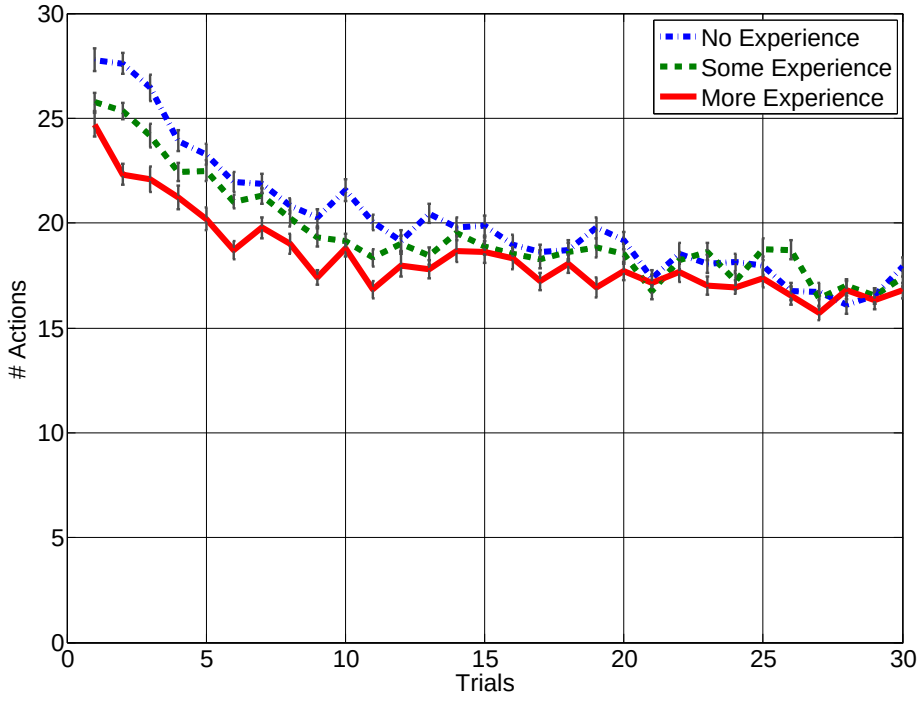


(b) Object used to gather experience

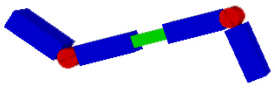


(c) Object used to measure transfer

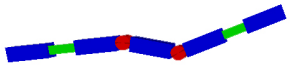
Figure 11.8. Experimental validation of knowledge transfer (RRRR to RRRRR). We compare a robot with experience manipulating an RRRR object (bottom left, solid green line) to an inexperienced robot (dashed blue line). Both robots learn to acquire the kinematic structure of a more complex object (RRRRR, bottom right). Experience improves performance by about 30% on the first trial, and remains relevant for the following 7 trials.



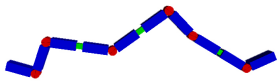
(a) Performance with and without prior experience



(b) Object used to gather experience

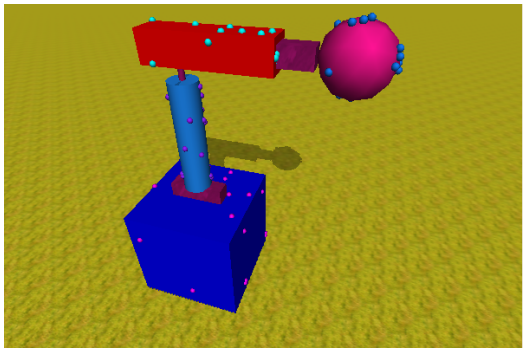


(c) Object used to measure transfer

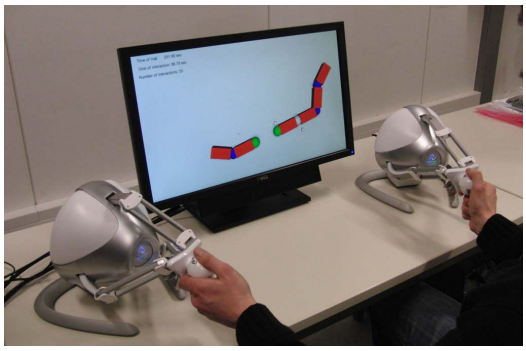


(d) Object used to measure transfer

Figure 11.9. Experimental validation of knowledge transfer (RPR to PRRP to RRPRPRRPR). We compare a robot with experience manipulating the RPR object (bottom left, dashed green line) to a robot with experience manipulating both the RPR (bottom left) and PRRP (bottom middle) objects (solid red line) to a robot with no prior experience (dashed blue line). All robots learn to acquire the kinematic structure of a more complex object (RRPRPRRPR, bottom right). Experience improves performance on the first trial, and more experience (2 training objects) leads to a greater improvement.



(a) Simulated three-dimensional rigid articulated object



(b) The display shows an articulated object. On the left and right sides: two haptic devices operated by a human subject to interact with the object.

Figure 11.10. Three-dimensional simulation environment

CHAPTER 12

CONCLUSION

To accomplish a manipulation task in an unstructured environment, a robot must have knowledge about the environment and be able to assess the outcome of its own actions. This knowledge is unlikely to be complete. There are simply too many aspects of the environment to consider, and too many actions that the robot could apply. Thus, we have argued that the main challenges for autonomous manipulation are to determine what to measure and how to interpret these measurements. Our approach to both challenges is the same: leveraging our knowledge of the task. In this thesis, we have explored these challenges through the task of manipulating rigid articulated objects. This task is important; it is a fundamental prerequisite for autonomous manipulation of many objects in our everyday environments.

How can knowledge of the task address these two challenges? First, the task identifies the aspects of the environment that are essential for its successful completion. A robot can limit itself to measure these aspects alone, thus significantly decreasing the complexity of the associated state space. In our case, knowledge of the task allows us to limit our measurements to aspects such as links, joints, and motion. Second, the task imposes structure on the state space. Within this structure, the meaning of measurements can be interpreted. For example, in our case, measurements of position can be processed to identify relative motion (eventually leading to kinematic relationships). For another task, GPS based navigation for example, position information will have a different meaning.

We have proposed three hypotheses for developing manipulation capabilities in unstructured environments. All three hypotheses leverage task-knowledge.

1. **Representation:** The symbolic representation of the environment must be task-specific

2. **Grounding:** Symbols must be grounded in the robot's sensorimotor interactions with the environment

3. **Hierarchy:** Within a task-hierarchy, simple subtasks enable more complex tasks by "reshaping" the state space

Based on these hypotheses, we have developed a system for perceiving and manipulating articulated objects. The system is composed of several components, each solving a subproblem. We approached the decomposition of the solution into these specific components by considering the context of the task, and each component uses a task-specific grounded representation. We believe that the success of our approach, demonstrated in simulated and real-world experiment supports our initial hypotheses.

We started our journey by proposing a solution to the simplified problem of manipulating **planar** rigid articulated objects. Our understanding of the task indicated what should be measured (relative motion), and how to interpret this information (joints and links). We proposed a skill for identifying planar rigid bodies, and determining the kinematic relationships between these bodies. This skill tightly integrates perception and action with model acquisition. The integration is task-specific; it leverages the embodiment of the robot to generate the important relative motion signal. Our solution provides a task-specific symbolic representation of the environment in the form of links and joints (following our 1^{st} hypothesis). These symbols are grounded in the robot's sensorimotor skills (following our 2^{nd} hypothesis). This grounded, task-specific representation "reshapes" the state space into a simpler, discrete state-space (following our 3^{rd} hypothesis) which forms the basis for the more

complex task of learning manipulation knowledge. The demonstrated robustness and effectiveness of this skill in unstructured environments support our decomposition of the problem. We believe that we could not have achieved this result with a non task-specific decomposition.

Encouraged by our success with planar objects, we have next developed a solution to the general case of three-dimensional rigid articulated objects. Here, we decomposed the problem into three subproblems (factors): identifying rigid bodies, reconstructing the bodies' three-dimensional shape and motion, and detecting kinematic constraints. We leveraged the structure of each subproblem to develop a simple and efficient solution. To identify rigid bodies (links) in an unstructured scene, we leveraged the notion of an "object" for manipulation. To measure "object-ness" we considered several relevant aspects of the environment, including color, texture, spatial contingency, and motion. Here too we relied on the robot's embodiment to reveal task-relevant information (i.e. relative motion). Similar to the planar case, here too our representation is grounded and task-specific (following our 1^{st} and 2^{nd} hypotheses). To compute the three-dimensional structure and motion of objects, we exploited the fact that the first factor has "reshaped" the state space (following our 3^{rd} hypothesis); we could now consider one rigid body at a time. The structure of the task of our second factor imposes a relationship between the three-dimensional motion of a body and its structure (i.e. rigid body constraints). Our third factor, joint detection, computes kinematic relationships. Its task is straight forward given the individual 3-D trajectories of a set of rigid bodies and the limited number of 1-degree-of-freedom joints. Its simplicity is due to our decomposition of the problem. Finally, to form a solution to the more difficult original problem, we simply recomposed the three subproblems (following our 3^{rd} hypothesis). The demonstrated robustness and effectiveness of our 3-D skill support our decomposition of the overall problem.

The above two skills provide our robot with the perceptual capabilities necessary to manipulate rigid articulated objects. However, manipulation in unstructured environment has to be not only reliable but also directed and efficient. The last part of this thesis focused on the problem of enabling a robot to autonomously acquire and generalize manipulation expertise. The task-specific representation provided by the robot's perceptual capabilities turned out to be crucial for enabling a simple reinforcement learning approach to an originally high-dimensional problem. The grounded symbols allowed learning in the symbolic domain to be applied to the real-world. Learning only became possible because the state space it had to consider was "reshaped" by our solution to the simpler task of modeling articulated objects (following our 3^{rd} hypothesis).

We believe that the concept of task-specific decomposition is an important enabler of progress for manipulation in unstructured environments. This thesis shows results in support of this view. If this is indeed an important principle, it should encourage researchers to shift their emphasis from developing narrow, high-performance systems to building robust, versatile, and integrated systems with lower-level capabilities that can be applied to a rich variety of problem domains. Furthermore, this principle should also encourage researchers to build these integrated systems incrementally, starting with very basic skills, such as the ones presented here, rather than by integrating state-of-the-art approaches to individual aspects of real-world problems.

In summary, the main contributions of this thesis are:

1. A set of perceptual skills for acquiring and interpreting shape and kinematic models for rigid articulated objects

2. A learning framework, enabling a robot to obtain general domain knowledge for manipulations

3. An "enactive" approach to perception (interactive perception), which exploits the synergies at the boundary between action and perception

The following statement by Hans Moravec, therefore, seems an appropriate conclusion to the principles underlying the work that we presented in this thesis: "Encoded in the large, highly evolved sensory and motor portions of the human brain is a billion years of experience about the nature of the world and how to survive in it. The deliberate process we call reasoning is, I believe, the thinnest veneer of human thought, effective only because it is supported by this much older and much powerful, though usually unconscious, sensorimotor knowledge. We are all prodigious olympians in perceptual and motor areas, so good that we make the difficult look easy. Abstract thought, though, is a new trick, perhaps less than 100 thousand years old. We have not yet mastered it. It is not all that intrinsically difficult; it just seems so when we do it."

APPENDIX A

THE NORMALIZED 8-POINT ALGORITHM

The 8-point algorithm for computing the essential matrix was originally introduced by Longuet-Higgins [81]. The algorithm was used to compute the structure of a scene from two views using calibrated cameras. The advantage of the original 8-point algorithm is its efficiency. It is a linear algorithm, and therefore fast and easy to implement. The algorithm requires at least 8 point matches across the two views. When exactly 8 point matches are available, the algorithm simply solves a set of linear equations. When more than 8 point matches are available, a linear least squares minimization problem must be solved.

The essential matrix conveniently encapsulates the epipolar geometry of the imaging configuration. The same algorithm may be used to compute a matrix with this property from un-calibrated cameras. In this case, however, we refer to the matrix encapsulating the epipolar geometry as the fundamental matrix. The fundamental matrix may be used to reconstruct the scene from two un-calibrated views. However, unlike the calibrated case, the scene can be reconstructed only up to a projective transformation.

An improved version of the original algorithm was proposed by Hartley and Zisserman [62]. This algorithm does not require the camera to be calibrated, and therefore produces the fundamental matrix rather than the essential matrix. Its greatest improvement is due to the careful normalization of the input data before constructing the equations to be solved. This normalization is a simple transformation of the points in the image before formulating the linear equations. It translates and scales each

image such that the centroid of the reference points is at the origin of the coordinates and the Root Mean Square (RMS) distance of the points from the origin is equal to $\sqrt{2}$. The proposed normalization leads to a large improvement in the conditioning of the problem and therefore increases the stability of the result. We note that the additional complexity associated with normalization in insignificant.

The objective of the normalized 8-point algorithm is the following: Given $n \geq 8$ image points correspondences $\{x_i \leftrightarrow x_i^{'}\}$, determine the fundamental matrix F such that $x_i^{'T} F x_i = 0$. Its outline, as presented in [62], is given in Algorithm 1. We note that further improvements of this algorithm are available. We refer the interested reader to [61] for information about these other methods, as well as the advantages of the normalized 8-point algorithm.

Algorithm 1 The Normalized 8-point Algorithm

Input: $n \geq 8$ pairs of corresponding image points $\{x_i \leftrightarrow x_i^{'}\}$
Output: the fundamental matrix F such that $x_i^{'T} F x_i = 0$ (when $n > 8$, $x_i^{'T} F x_i \approx 0$)

1: **Normalization:** Transform the image coordinates according to $\hat{x}_i = T x_i$ and $\hat{x}_i^{'} = T^{'} x_i^{'}$ where T and $T^{'}$ are normalizing transformations consisting of a translation and scaling, such that the centroid is the origin, and $RMS = \sqrt{2}$.
2: Find the fundamental matrix $\hat{F}^{'}$ corresponding to the matches $\{\hat{x}_i \leftrightarrow \hat{x}_i^{'}\}$ by:

   1. **Linear Solution:** determine $\hat{F}$ from the singular vector corresponding to the smallest singular values of $\hat{A}$, where $\hat{A}$ is composed from the matches $\{\hat{x}_i \leftrightarrow \hat{x}_i^{'}\}$ as defined in section 11.3 of [62].

   2. **Constraint Enforcement:** replace $\hat{F}$ by $\hat{F}^{'}$ such that $\det \hat{F}^{'} = 0$ using the Singular Value Decomposition (SVD). For more information, see section 11.1.1 in [62].

3: **Denormalization:** set $F = T^{'T} \hat{F}^{'} T$. Matrix F is the fundamental matrix corresponding to the original data $\{x_i \leftrightarrow x_i^{'}\}$.

APPENDIX B

THE MAX-FLOW MIN-CUT ALGORITHM

The max-flow min-cut theorem states that in a flow network, the maximum amount of flow passing from the source s to the sink t is equal to the minimum capacity which when removed in a specific way from the network, separates the source from the sink. In other words, when this capacity is removed, no flow can pass from the source to the sink. Max-flow and min-cut are in fact dual problems. The maximum value of an s-t flow is equal to the minimum capacity of an s-t cut.

Formally, the problem is defined as follows:

Let $G = (V, E)$ be a directed graph with s being the source and t being the sink of G.

The capacity of an edge represents the maximum amount of flow that can pass through that edge. Edge capacity is a mapping $c : E \to \mathbb{R}^+$, denoted by c_{uv} where , $u \in V$, $v \in V$ and $(u, v) \in E$.

A flow is a mapping $f :\to \mathbb{R}^+$, denoted by f_{uv}, subject to the following two constraints:

1. $f_{uv} \leq c_{uv}$ for each $(u, v) \in E$ (capacity constraint)

2. $\sum_{u:\ (u,v)\in E} f_{uv} = \sum_{u:\ (v,u)\in E} f_{vu}$ for each $v \in V \setminus \{s, t\}$ (conservation of flows)

The value of a flow represents the amount of flow passing from the source s to the sink t of the graph G. The flow's value is defined as $|f| = \sum_{v\in V} f_{sv}$, where s is the source of G. Thus, the max-flow problem is to maximize $|f|$, i.e. to route as much flow as possible from the source s to the sink t.

An s-t cut $C = (S, T)$ is a partition of V such that $s \in S$ and $t \in T$. The cut-set of C is the set $\{(u, v) \in E | u \in S, v \in T\}$. Note that if the edges in the cut-set of C are removed, the flow is eliminated: $|f| = 0$.

The capacity of an s-t cut is defined by $c(S, T) = \sum_{(u,v) \in S \times T} c_{uv}$. Thus, the min-cut problem is to minimize $c(S, T)$, i.e. to find S and T such that the capacity of the S-T cut is minimal.

There are several closely related solutions to the min-cut/max-flow problem. The following pseudocode outlines the Edmonds-Karp algorithm. It implements the Ford-Fulkerson method for computing the maximum flow in a directed graph. The algorithm's run time is $\mathcal{O}(|V| \cdot |E|^2)$. It was first published by a Russian scientist, Yefim (Chaim) Dinic in 1970 [36]. It was independently proposed by Jack Edmonds and Richard Karp in 1972 [44].

Algorithm 2 The Edmonds-Karp Algorithm

Input:

1. Capacity matrix: $C[1..n, 1..n]$
2. Neighbor lists: $E[1..n, 1..?]$
3. Source: s
4. Sink: t

Output:

1. Value of maximum flow: f
2. A matrix giving a legal flow with the maximum value: F

```
1: f := 0 (Initial flow is zero)
2: F := array(1..n, 1..n) (Residual capacity from u to v is C[u,v] - F[u,v])
3: while m ≠ 0 do
4:    m, P := BreadthFirstSearch(C, E, s, t)
      where: m=capacity of path found, P=parent table
5:    f := f + m (Backtrack search, and write flow)
6:    v := t
7:    while v ≠ s do
8:       u := P[v]
9:       F[u,v] := F[u,v] + m
10:      F[v,u] := F[v,u] - m
11:      v := u
12:   end while
13: end while
14: return (f, F)
```

APPENDIX C

BUNDLE ADJUSTMENT

Bundle adjustment (BA) solves the following problem: given a set of image coordinates x_j^i, find the set of camera matrices P^i, and the points X_j such that $P^i X_j = x_j^i$. Without making further restrictions on the P^i or X_j, we refer to this reconstruction as "projective reconstruction". In other words, we are reconstructing the scene up to an arbitrary 3-D projective transformation on the points X_j.

Bundle adjustment jointly refines a set of initial camera and structure parameter estimates. Its goal is finding the set of parameters that most accurately predict the locations of the observed points in the set of available images. More formally: We assume that n 3-D points are observed in m views, and let x_j^i be the projection of the j^{th} point on image i. Further, we also assume that each camera i is parameterized by a vector a_i and each 3-D point j by a vector b_j.

Let v_j^i be binary variables denoting whether point j is visible in image i:

$$v_j^i = \begin{cases} 1 & \text{, if point } j \text{ is visible in image } i \\ 0 & \text{, otherwise} \end{cases}$$

The objective of Bundle adjustment is to minimize the sum of re-projection error with respect to all 3-D point and camera parameters. Formally, we can state this optimization problem as follows: $\min_{a_i,b_j} \sum_{j=1}^{n} \sum_{i=1}^{m} v_j^i d(Q(a_i, b_j), x_j^i)^2$ Where:

$Q(a_i, b_j)$	The predicted projection of point j on image i
$d(\alpha, \beta)$	The Euclidean distance between the image points α and β

We note that, by definition, Bundle adjustment can deal with missing image projections. We refer the reader to [82] and [62] for a detailed review of Bundle adjustment.

APPENDIX D

THE EXTENDED KALMAN FILTER

The extended Kalman filter (EKF) is a non-linear Kalman filter. Like a standard Kalman filter, it is linear with respect to the current mean and covariance, but the state transition and observation models are not necessarily a linear function of the state. Instead, it requires that the state transition and observation models are differentiable functions (i.e. both f' and h' exist):

1. $x_k = f(x_{k-1}, u_{k-1}) + w_{k-1}$

2. $z_k = h(x_k) + v_k$

Where w_k is the process noise and v_k is observation noise. Both sources of noise are assumed to be zero mean multivariate Gaussian noises. The covariance matrices associated with the process and observation noise are Q_k and R_k, respectively.

The function f computes the predicted state based on the previous state estimate. The function h relies on the predicted state to compute the predicted measurement. Both function, $f()$ and $h()$, cannot be applied directly to update the covariance matrices. We use instead a Jacobian matrix (a matrix of partial derivatives). The Jacobian is recomputed using the current predicted state at every time step. It provides a local linearization of the non-linear function around the current estimate.

Following are the state predict and update equations for the extended Kalman filter:

The Predict Equations

Predicted state	$\hat{x}_{k\|k-1} = f(\hat{x}_{k-1\|k-1}, u_{k-1})$
Predicted estimate covariance	$P_{k\|k-1} = F_{k-1} P_{k-1\|k-1} F_{k-1}^{\top} + Q_{k-1}$

The Update Equations

Innovation or measurement residual	$\tilde{y}_k = z_k - h(\hat{x}_{k\|k-1})$
Innovation or residual covariance	$S_k = H_k P_{k\|k-1} H_k^{\top} + R_k$
Optimal Kalman gain	$K_k = P_{k\|k-1} H_k^{\top} S_k^{-1}$
Updated state estimate	$\hat{x}_{k\|k} = \hat{x}_{k\|k-1} + K_k \tilde{y}_k$
Updated estimate covariance	$P_{k\|k} = (I - K_k H_k) P_{k\|k-1}$

The state transition and observation matrices are defined to be the following Jacobians:

- $F_{k-1} = \frac{\partial f}{\partial x}\big|_{\hat{x}_{k-1\|k-1}, u_{k-1}}$
- $H_k = \frac{\partial h}{\partial x}\big|_{\hat{x}_{k\|k-1}}$

The extended Kalman filter is not an optimal estimator (except for the trivial case of linear observation model and linear state transition model). The EKF is sensitive to the initial estimate. With wrong initialization, the EKF will typically quickly diverge. Naturally, mistakes in modeling the process itself will also result in divergence. Despite these limitations, the performance of the extended Kalman filter is often reasonable. Arguably, it is the defacto method in navigation systems and GPS.

BIBLIOGRAPHY

[1] Agrawal, Motilal, and Konolige, Kurt. FrameSLAM: From Bundle Adjustment to Real-Time Visual Mapping. *IEEE Transactions on Robotics 24*, 5 (October 2008), 1066–1077.

[2] Aloimonos, John, Weiss, Isaac, and Bandyopadhyay, Amit. Active Vision. *International Journal of Computer Vision 1*, 4 (January 1988), 333–356.

[3] Altendorfer, Richard, Moore, Ned, Komsuoglu, Haldun, Buehler, Martin, Brown, Hb, McMordie, Dave, Saranli, Uluc, Full, Robert J, and Koditschek, Daniel E. RHex: A Biologically Inspired Hexapod Runner. *Autonomous Robots 11*, 3 (2001), 207–213.

[4] Anderson, Dean, Herman, Herman, and Kelly, Alonzo. Experimental Characterization of Commercial Flash Ladar Devices. In *International Conference of Sensing and Technology* (November 2005), G. Sen Gupta, S.C. Mukhopandhyay, and C.H. Messom, Eds.

[5] Anguelov, Dragomir, Koller, Daphne, Parker, Evan, and Thrun, Sebastian. Detecting and Modeling Doors with Mobile Robots. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)* (New Orleans, LA, USA, April 2004), IEEE, pp. 3777–3784.

[6] Ariely, Dan. *Predictably Irrational: The Hidden Forces That Shape Our Decisions*. HarperCollins, February 2008.

[7] Arun, K. S., Huang, T. S., and Blostein, S. D. Least-Squares Fitting of two 3D Point Sets. *IEEE Transactions on Pattern Analysis and Machine Intelligence 9*, 5 (September 1987), 698–700.

[8] Azarbayejani, Ali, and Pentland, Alex P. Recursive Estimation of Motion, Structure, and Focal Length. *IEEE Transactions on Pattern Analysis and Machine Intelligence 17*, 6 (June 1995), 562–575.

[9] Bajcsy, R. Active Perception. *Proceedings of the IEEE, Special issue on Computer Vision 76*, 9 (August 1988), 996–1006.

[10] Barton, M. E., and Komatsu, L. K. Defining Features of Natural Kinds and Artifacts. *Journal of Psycholinguistic Research 18*, 5 (September 1989), 433–447.

[11] Blake, Andrew, and Yuille, Alan. *Active Vision*. MIT Press, November 1992.

[12] Boehm, Jan. Model-Based Segmentation and Recognition from Range Data. In *Optomechatronic Machine Vision* (December 2005), Kazuhiko Sumi, Ed., vol. 6051, SPIE.

[13] Bogoni, Luca, and Bajcsy, Ruzena. Interactive Recognition and Representation of Functionality. *Computer Vision and Image Understanding 62*, 2 (September 1995), 194–214.

[14] Bonawitz, Elizabeth Baraff, Ferranti, Darlene, Saxe, Rebecca, Gopnik, Alison, Meltzoff, Andrew N, Woodward, James, and Schulz, Laura E. Just do it? Investigating the Gap Between Prediction and Action in Toddlers' Causal Inferences. *International Journal of Cognitive Science 115*, 1 (April 2010), 104–117.

[15] Boutilier, C., Dean, T., and Hanks, S. Decision theoretic planning: Structural assumptions and computational leverage. *Journal of Artificial Intelligence Research 11* (1999), 1–94.

[16] Brooks, Rodney A. A Robust Layered Control System for a Mobile Robot. *International Journal of Robotics and Automation RA-2*, 1 (1986), 14–23.

[17] Brooks, Rodney A. Intelligence without Reason. In *Proceedings of the International Joint Conference on Artificial Intelligence* (Sydney, Australia, August 1991), John Mylopoulos and Raymond Reiter, Eds., vol. 1, Morgan Kaufmann, pp. 569–595.

[18] Campbell, R.J., and Flynn, P.J. Eigenshapes for 3D Object Recognition in Range Data. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (Fort Collins, CO , USA, June 1999), vol. 2, IEEE, pp. 505–510.

[19] Canny, J. A Computational Approach to Edge Detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 6 (November 1986), 679–698.

[20] Chaigneau, Sergio E., and Barsalou, Lawrence W. The Role of Function in Categories. *Theoria et Historia Scientiarum* (2001).

[21] Chiuso, A., Favaro, P., Jin, H., and Soatto, S. Structure from Motion causally Integrated over Time. *IEEE Transactions on Pattern Analysis and Machine Intelligence 24*, 4 (April 2002), 523–535.

[22] Chli, Margarita, and Davison, Andrew J. Active Matching for Visual Tracking. *Robotics and Autonomous Systems 57*, 12 (December 2009), 1173–1187.

[23] Christiansen, Alan D., Mason, Matthew, and Mitchell, Tom. Learning Reliable Manipulation Strategies without Initial Physical Models. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)* (Cincinnati, Ohio, USA, May 1990), vol. 2, IEEE, pp. 1224–1230.

[24] Civera, J. *Real-Time EKF-Based Structure from Motion*. PhD thesis, Universidad de Zaragoza, Zaragoza, Spain, 2009.

[25] Civera, J., Grasa, Oscar G., Davison, A. J., and Montiel, J. M. M. 1-Point RANSAC for EKF-Based Structure from Motion. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (St. Louis, MO, USA, October 2009), IEEE/RSJ, pp. 3498–3504.

[26] Cohn, J. F, and Tronick, E. Z. Mother-Infant Face-to-Face Interaction: Influence is Bidirectional and Unrelated to Periodic Cycles in Either Partner's Behavior. *Developmental psychology 24*, 3 (May 1988), 386–392.

[27] Comport, Andrew I., Malis, Ezio, and Rives, Patrick. Accurate Quadrifocal Tracking for Robust 3D Visual Odometry. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)* (Rome, Italy, April 2007), IEEE, pp. 40–45.

[28] Cordella, L. P., Foggia, P., Sandone, C., and Vento, M. Performance Evaluation of the VF Graph Matching Algorithm. In *Proceedings of the International Conference on Image Analysis and Processing* (Venice, Italy, September 1999), vol. 2, IEEE Computer Society, pp. 1038–1041.

[29] Cormen, Thomas H., Leiserson, Charles E., Rivest, Ronald L., and Stein, Clifford. *Introduction to Algorithms*. The MIT Press, September 2009.

[30] Costeira, J.P., and Kanade, T. A Multibody Factorization Method for Independently Moving Objects. *International Journal of Computer Vision 29*, 3 (September 1998), 159–179.

[31] Craig, John J. *Introduction to Robotics: Mechanics and Control*. Addison-Wesley Longman Publishing Co., Inc., 1989.

[32] Davison, Andrew J. Real-Time Simultaneous Localisation and Mapping with a Single Camera. In *Proceedings of the International Conference on Computer Vision* (Washington, DC, USA, October 2003), vol. 2, IEEE Computer Society, pp. 1403–1410.

[33] Davison, Andrew J., Reid, Ian D., Molton, Nicholas D., and Stasse, Olivier. MonoSLAM: Real-Time Single Camera SLAM. *IEEE Transactions on Pattern Analysis and Machine Intelligence 29*, 6 (June 2007), 1052–1067.

[34] Delaunay, Boris N. Sur la sphère vide. *Bulletin of Academy of Sciences of the USSR*, 6 (1934), 793–800.

[35] Dellaert, Frank, and Kaess, Michael. Square Root SAM: Simultaneous Localization and Mapping via Square Root Information Smoothing. *International Journal of Robotics Research 25*, 12 (December 2006), 1181–1203.

[36] Dinic, E. A. Algorithm for Solution of a Problem Of Maximum Flow in Networks with Power Estimation. *Soviet Math. Dokl 11* (1970), 1277–1280.

[37] Dissanayake, M. W. M. G., Newman, P., Clark, S., Durrant-Whyte, H. F., and Csorba, M. A solution to the simultaneous localization and map building (SLAM) problem. *IEEE Transactions on Robotics and Automation 17*, 3 (June 2001), 229–241.

[38] Dollar, Aaron M., and Howe, Robert D. The SDM Hand as a Prosthetic Terminal Device: A Feasibility Study. In *Proceedings of the IEEE International Conference on Rehabilitation Robotics* (Noordwijk, Netherlands, June 2007).

[39] Driessens, Kurt, and Ramon, Jan. Relational Instance Based Regression for Relational Reinforcement Learning. In *International Conference on Machine Learning* (Washington, DC, USA, August 2003), Tom Fawcett and Nina Mishra, Eds., AAAI Press, pp. 123–130.

[40] Durrant-Whyte, Hugh, Majumder, Somajyoti, Thrun, Sebastian, de Battista, Marc, and Scheding, Steve. A Bayesian Algorithm for Simultaneous Localisation and Map Building. In *Robotics Research*, Raymond Jarvis and Alexander Zelinsky, Eds., vol. 6 of *Springer Tracts in Advanced Robotics*. Springer Berlin / Heidelberg, Berlin, Heidelberg, June 2003, ch. 4, pp. 49–60.

[41] Džeroski, Sašo, and Blockeel, Luc de Raedt H. Relational reinforcement learning. In *International Conference on Machine Learning* (August 98), Morgan Kaufmann, pp. 136–143.

[42] Džeroski, Sašo, de Raedt, Luc, and Driessens, Kurt. Relational Reinforcement Learning. *Machine Learning 43*, 1–2 (2001), 7–52.

[43] Eade, E., and Drummond, T. Scalable Monocular SLAM. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (Washington, DC, USA, July 2006), vol. 1, IEEE Computer Society, pp. 469–476.

[44] Edmonds, Jack, and Karp, Richard M. Theoretical Improvements in Algorithmic Efficiency for Network Flow Problems. *Journal of the ACM 19*, 2 (1972), 248–264.

[45] Edsinger, Aaron, and Kemp, Charles C. Manipulation in Human Environments. In *Proceedings of the IEEE International Conference on Humanoid Robots* (Genova, Italy, December 2006), IEEE, pp. 102–109.

[46] Ellis, Willis D. *A Source Book of Gestalt Psychology*. The Gestalt Journal Press, January 1997.

[47] Felzenszwalb, Pedro F., and Huttenlocher, Daniel P. Efficient Graph-Based Image Segmentation. *International Journal of Computer Vision 59*, 2 (September 2004), 167–181.

[48] Fischler, Martin A., and Bolles, Robert C. RANdom SAmple Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography. *Communication of the ACM 24*, 6 (1981), 381–395.

[49] Fitzpatrick, Paul. First Contact: An Active Vision Approach to Segmentation. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (Ocotber 2003), vol. 3, IEEE, pp. 2161–2166.

[50] Forsyth, David A., and Ponce, Jean. *Computer Vision: A Modern Approach*. Prentice Hall Professional Technical Reference, 2002.

[51] Fowlkes, Charless C., Martin, David R., and Malik, Jitendra. Local Figure-Ground Cues are Valid for Natural Images. *Journal of Vision 7*, 8 (June 2007).

[52] Gallagher, Shaun. *How the Body Shapes the Mind*. Oxford University Press, USA, December 2006.

[53] Gibson, James J. *The Senses Considered as Perceptual Systems*. Greenwood Press Reprint, June 1983.

[54] Gibson, James J. *The Ecological Approach to Visual Perception*. Lawrence Erlbaum, September 1986.

[55] Goh, Alvina, and Vidal, Rene. Segmenting Motions of Different Types by Unsupervised Manifold Clustering. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (Minneapolis, Minnesota, USA, June 2007), IEEE Computer Society.

[56] gon Roh, Se, and Choi, Hyouk Ryeol. 3D Tag-Based RFID System for Recognition of Object. *IEEE Transactions on Automation Science and Engineering 6*, 1 (January 2009), 55–65.

[57] Green, Kevin, Eggert, David, Stark, Louise, and Bowyer, Kevin. Generic Recognition of Articulated Objects through Reasoning about Potential Function. *Computer Vision and Image Understanding 62*, 2 (September 1995), 177–193.

[58] Guerra-Filho, Gutemberg. Optical Motion Capture: Theory and Implementation. *Journal of Theoretical and Applied Informatics 12*, 2 (2005), 61–89.

[59] Harnad, Stevan. The Symbol Grounding Problem. *Physica D: Nonlinear Phenomena 42*, 1–3 (June 1990), 335–346.

[60] Harris, C., and Stephens, M. A Combined Corner and Edge Detector. In *Proceedings of The Fourth Alvey Vision Conference* (1988), pp. 147–151.

[61] Hartley, R. I. In Defense of the Eight-Point Algorithm. *IEEE Transactions on Pattern Analysis and Machine Intelligence 19* (1997), 580–593.

[62] Hartley, R. I., and Zisserman, A. *Multiple View Geometry in Computer Vision*. Cambridge University Press, 2004.

[63] Hartuv, E., and Shamir, R. A Clustering Algorithm based on Graph Connectivity. *Information Processing Letters 76*, 4–6 (December 2000), 175–181.

[64] Holmes, Steven A., Klein, Georg, and Murray, David W. An $\mathcal{O}(N^2)$ Square Root Unscented Kalman Filter for Visual Simultaneous Localization and Mapping. *IEEE Transactions on Pattern Analysis and Machine Intelligence 31* (2009), 1251–1263.

[65] Horn, Berthold K. P. Closed-Form Solution of Absolute Orientation using Unit Quaternions. *Journal of the Optical Society of America. A 4*, 4 (1987), 629–642.

[66] Horn, Berthold Klaus Paul. *Robot Vision*. The MIT Press, March 1986.

[67] Hu, Zhilan, Wang, Guijin, Lin, Xinggang, and Yan, Hong. Recovery of Upper Body Poses in Static Images based on Joints Detection. *Pattern Recognition Letters 30*, 5 (April 2009), 503–512.

[68] Huang, T. S., Blostein, S. D., and Margerum, E. A. Least-squares Estimation of Motion Parameters from 3D Point Correspondences. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (Miami Beach, FL, USA, 1986), IEEE, pp. 24–26.

[69] Hutchinson, Seth, Hager, Gregory, and Corke, Peter. A Tutorial on Visual Servo Control. *IEEE Transactions on Robotics and Automation 12*, 5 (October 1996), 651–670.

[70] J. A. Castellanos, J. M. M. Montiel, J. Neira, and Tardós, J. D. The SPmap: A Probabilistic Framework for Simultaneous Localization and Map Building. *IEEE Transactions on Robotics and Automation 15*, 5 (October 1999), 948–952.

[71] Jägersand, Martin, Fuentes, Olac, and Nelson, Randal. Experimental Evaluation of Uncalibrated Visual Servoing for Precision Manipulation. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)* (Albuquerque, NM, USA, April 1997), vol. 4, IEEE, pp. 2874–2880.

[72] Jin, Hailin, Favaro, Paolo, and Soatto, Stefano. A Semi-Direct Approach to Structure from Motion. *The Visual Computer 19*, 6 (2003), 377–394.

[73] Julier, S., and Uhlmann, J. A New Extension of the Kalman Filter to Nonlinear Systems. In *International Symposium on Aerospace/Defense Sensing, Simulation and Controls* (Orlando, FL, 1997), pp. 182–193.

[74] Kaelblin, L.P., Oates, T., Hernandez, N., and Finney, S. Learning in worlds with objects. In *The AAAI Spring Symposium* (March 2001).

[75] Kaelbling, Leslie Pack. *Learning in Embedded Systems*. MIT Press, May 1993.

[76] Kanazawa, Yasushi, and Kawakami, Hiroshi. Detection of Planar Regions with Uncalibrated Stereo using Distribution of Feature Points. In *British Machine Vision Conference* (Kingston, UK, 2004), pp. 247–256.

[77] Kirk, Adam G., O'Brien, James F., and Forsyth, David A. Skeletal Parameter Estimation from Optical Motion Capture Data. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (Washington, DC, USA, 2005), IEEE Computer Society, pp. 782–788.

[78] Klein, Georg, and Murray, David. Parallel Tracking and Mapping for Small AR Workspaces. In *International Symposium on Mixed and Augmented Reality* (Nara, Japan, November 2007), IEEE.

[79] Klein, Georg, and Murray, David. Improving the Agility of Keyframe-Based SLAM. In *European Conference on Computer Vision* (Marseille, France, October 2008), IEEE, pp. 802–815.

[80] Konolige, Kurt, Agrawal, Motilal, and Solà, Joan. Large Scale Visual Odometry for Rough Terrain. In *Proceedings of the International Symposium of Robotics Research* (Hiroshima, Japan, November 2007).

[81] Longuet-Higgins, H. C. A Computer Algorithm for Reconstructing a Scene from Two Projections. *Nature 293* (September 1981), 133–135.

[82] Lourakis, M.I. A., and Argyros, A.A. SBA: A Software Package for Generic Sparse Bundle Adjustment. *ACM Transactions on Mathematical Software 36*, 1 (2009), 1–30.

[83] Lowe, David G. Distinctive Image Features from Scale-Invariant Keypoints. *International Journal of Computer Vision 60*, 2 (November 2004), 91–110.

[84] Lu, F., and Milios, E. Globally Consistent Range Scan Alignment for Environment Mapping. *Autonomous Robots 4*, 4 (1997), 333–349.

[85] Lucas, Bruce D., and Kanade, Takeo. An Iterative Image Registration Technique with an Application to Stereo Vision. In *Proceedings of the International Joint Conference on Artificial Intelligence* (Vancouver, BC, Canada, August 1981), pp. 674–679.

[86] Malik, Jitendra, Belongie, Serge, Leung, Thomas, and Shi, Jianbo. Contour and Texture Analysis for Image Segmentation. *International Journal of Computer Vision 43*, 1 (June 2001), 7–27.

[87] Marr, David. *Vision*. H. Freeman and Co., July 1982.

[88] Matula, D. W. Determining Edge Connectivity in $\mathcal{O}(nm)$. *Proceedings of the 28th Symp. on Foundations of Computer Science* (October 1987), 249–251.

[89] Metta, Giorgio, and Fitzpatrick, Paul. Early Integration of Vision and Manipulation. *Adaptive Behavior 11*, 2 (June 2003), 109–128.

[90] Mikhail, E.M., Bethel, J.S., and J.C., McGlone. *Introduction to Modern Photogrammetry*. John Wiley & Sons, March 2001.

[91] Mitra, Niloy J., Gelfand, Natasha, Pottmann, Helmut, and Guibas, Leonidas. Registration of Point Cloud Data from a Geometric Optimization Perspective. In *Proceedings of the Eurographics Symposium on Geometry Processing* (New York, NY, USA, 2004), ACM, pp. 22–31.

[92] Montemerlo, M., Thrun, S., Koller, D., and Wegbreit, B. FastSLAM: A Factored Solution to the Simultaneous Localization and Mapping Problem. In *Proceedings of the National Conference on Artificial Intelligence (AAAI)* (Menlo Park, CA, USA, 2002), American Association for Artificial Intelligence, pp. 593–598.

[93] Montiel, J., Civera, J., and Davison, Andrew. Unified Inverse Depth Parametrization for Monocular SLAM. In *Proceedings of Robotics: Science and Systems (RSS)* (Philadelphia, Pennsylvania, USA, August 2006).

[94] Mouragnon, E., Lhuillier, M., Dhome, M., Dekeyser, F., and Sayd, P. Real Time Localization and 3D Reconstruction. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (New York, NY, USA, 2006), vol. 1, IEEE, pp. 363–370.

[95] Natsev, Apostol. Multimodal Search for Effective Video Retrieval. In *International Conference on Image and Video Retrieval* (Tempe, AZ, USA, July 2006), Hari Sundaram, Milind R. Naphade, John R. Smith, and Yong Rui, Eds., vol. 4071 of *Lecture Notes in Computer Science*, Springer, pp. 525–528.

[96] Nistér, David. An Efficient Solution to the Five-Point Relative Pose Problem. *IEEE Transactions on Pattern Analysis and Machine Intelligence 26*, 6 (April 2004), 756–770.

[97] Nistér, David, Naroditsky, Oleg, and Bergen, James R. Visual Odometry. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2004), vol. 1, IEEE Computer Society, pp. 652–659.

[98] Nistér, David, Naroditsky, Oleg, and Bergen, James R. Visual Odometry for Ground Vehicle Applications. *Journal on Field Robotics* (2006), 3–20.

[99] Noë, Alva. *Action in Perception*. MIT Press, March 2004.

[100] Noë, Alva, and Thompson, Evan. *Vision and Mind: Selected Readings in the Philosophy of Perception*. MIT Press, September 2002.

[101] O'Donohue, William, and Kitchener, Richard. *Handbook of Behaviorism*. Academic Press, October 1998.

[102] Oullier, O., de Guzman, G. C, Jantzen, K. J, Lagarde, J., and Kelso, J. A. Scott. Social Coordination Dynamics: Measuring Human Bonding. *Social Neuroscience 3*, 2 (2008), 178–192.

[103] Pan, Q., Reitmayr, G., and Drummond, T. ProFORMA: Probabilistic Feature-based On-line Rapid Model Acquisition. In *British Machine Vision Conference* (London, UK, September 2009).

[104] Pereira, Alfredo, Smith, Linda, and Yu, Chen. Social Coordination in Toddler's Word Learning: Interacting Systems of Perception and Action. *Connection Science 20*, 2 (2008), 73–89.

[105] Pfeifer, Rolf, and Bongards, Josh. *How the Body Shapes the Way We Think—A New View of Intelligence*. MIT Press, November 2006.

[106] Pfeifer, Rolf, and Iida, Fumiya. Morphological Computation: Connecting Body, Brain and Environment. *Japanese Scientific Monthly 58*, 2 (2005), 48–54.

[107] Poole, David. The Singular Value Decomposition. In *Linear Algebra Modern Introduction*. Addison-Wesley, Reading, MA, USA, June 2006, ch. 7, pp. 599–615.

[108] Poppe, Ronald, and Poel, Mannes. Example-Based Pose Estimation in Monocular Images using Compact Fourier Descriptors. Ctit technical report series, University of Twente, Centre for Telematics and Information Technology, Enschede, 2005.

[109] Prokhorov, D.V. Object Recognition in 3D Lidar Data with Recurrent Neural Network. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2009), 9–15.

[110] Puterman, M. L. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. John Wiley and Sons, Inc., 1994.

[111] Qian, Gang, and Chellappa, Rama. Structure from Motion Using Sequential Monte Carlo Methods. *International Journal of Computer Vision 59*, 1 (August 2004), 5–31.

[112] Rasiwasia, Nikhil, Moreno, Pedro J., and Vasconcelos, Nuno. Bridging the Gap: Query by Semantic Example. *IEEE Transactions on Multimedia 9*, 5 (2007), 923–938.

[113] Ren, Xiaofeng, and Malik, Jitendra. A Probabilistic Multi-scale Model for Contour Completion Based on Image Statistics. In *European Conference on Computer Vision* (London, UK, 2002), vol. 1, Springer-Verlag, pp. 312–327.

[114] Rensink, R. A, and Clark, J. J. On the Failure to Detect Changes in Scenes Across Brief Interruptions. *Visual Cognition 7*, 1 (2000), 127–145.

[115] Ross, David A., Tarlow, Daniel, and Zemel, Richard S. Unsupervised Learning of Skeletons from Motion. In *European Conference on Computer Vision* (Berlin, Heidelberg, 2008), Springer-Verlag, pp. 560–573.

[116] Russell, Stuart, and Wefald, Eric H. *Do the Right Thing: Studies in Limited Rationality*. The MIT Press, January 2003.

[117] Santrock, John W. *Topical Approach to Life-Span Development*. McGraw-Hill Publishing Company, 2008.

[118] Smith, Linda B., and Heise, Diana. Perceptual Similarity and Conceptual Structure. In *Percepts, Concepts and Categories - The Representation and Processing of Information*, Barbara Burns, Ed., vol. 93 of *Advances in Psychology*. North-Holland, 1992, pp. 233–272.

[119] Smith, Randall, Self, Matthew, and Cheeseman, Peter. A Stochastic Map for Uncertain Spatial Relationships. In *Proceedings of the International Symposium of Robotics Research* (Cambridge, MA, USA, 1988), MIT Press, pp. 467–474.

[120] Snavely, Noah, Seitz, Steven M., and Szeliski, Richard. Photo Tourism: Exploring Photo Collections in 3D. In *Proceedings of ACM Special Interest Group on Computer Graphics and Interative Techniques (SIGGRAPH)* (New York, NY, USA, 2006), ACM Press, pp. 835–846.

[121] Stoytchev, Alexander. Behavior-Grounded Representation of Tool Affordances. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)* (Barcelona, Spain, April 2005), IEEE, pp. 3071–3076.

[122] Stuart, Bradley, Baker, Patrick, and Aloimonos, Yiannis. Towards the Ultimate Motion Capture Technology. In *DEFORM/AVATARS* (2000), pp. 143–157.

[123] Sturm, Jürgen, Jain, Advait, Stachniss, Cyrill, Kemp, Charlie, and Burgard, Wolfram. Operating Articulated Objects Based on Experience. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (Taipei, Taiwan, 2010), IEEE.

[124] Sturm, Jürgen, Konolige, Kurt, Stachniss, C., and Burgard, W. Vision-Based Detection for Learning Articulation Models of Cabinet Doors and Drawers in Household Environments. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)* (Anchorage, Alaska, May 2010), IEEE.

[125] Sturm, Jürgen, Plagemann, Christian, and Burgard, Wolfram. Adaptive Body Scheme Models for Robust Robotic Manipulation. In *Proceedings of Robotics: Science and Systems (RSS)* (Zurich, Switzerland, June 2008).

[126] Sturm, Jürgen, Plagemann, Christian, and Burgard, Wolfram. Body Schema Learning for Robotic Manipulators from Visual Self-Perception. *Journal of Physiology 103*, 3–5 (2009), 220–231.

[127] Sutton, Melanie A., Stark, Louise, and Bowyer, Kevin W. Function from Visual Analysis and Physical Interaction: A Methodology for Recognition of Generic classes of Objects. *Image and Vision Computing 16*, 11 (1998), 745–763.

[128] Sutton, R. S., and Barto, A. G. *Reinforcement Learning: An Introduction*. MIT Press, 1998.

[129] Tadepalli, Prasad, Givan, Robert, and Driessens, Kurt. Relational Reinforcement Learning: An Overview. In *Proceedings of the Workshop on Relational Reinforcement Learning at ICML '04* (Banff, Canada, 2004).

[130] Taubin, Gabriel. Estimation of Planar Curves, Surfaces, and Nonplanar Space Curves Defined by Implicit Equations with Applications to Edge and Range Image Segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence 13*, 11 (November 1991), 1115–1138.

[131] Taycher, Leonid, Iii, John W. Fisher, and Darrell, Trevor. Recovering Articulated Model Topology from Observed Motion. In *Neural Information Processing Systems* (2002), MIT Press, pp. 1311–1318.

[132] Thrun, Sebastian, Burgard, Wolfram, and Fox, Dieter. *Probabilistic Robotics*. MIT Press, September 2005.

[133] Toldo, R., and Fusiello, A. Robust Multiple structures Estimation with J-Linkage. In *European Conference on Computer Vision* (2008), Andrew Zisserman, David Forsyth, and Philip Torr, Eds., vol. 5302, pp. 537–547.

[134] Tombari, Federico, and Konolige, Kurt. A Practical Stereo System based on Regularization and Texture Projection. In *International Conference on Informatics in Control, Automation and Robotics* (2009), pp. 5–12.

[135] Triggs, Bill, McLauchlan, Philip F., Hartley, Richard I., and Fitzgibbon, Andrew W. Bundle Adjustment — A Modern Synthesis. In *Proceedings of the International Conference on Computer Vision* (London, UK, 1999), Springer-Verlag, pp. 298–372.

[136] van Otterlo, Martijn. A Survey of Reinforcement Learning in Relational Domains. Tr-ctit-05-31, Department of Computer Science, University of Twente, Twente, Netherlands, 2005.

[137] Varela, Francisco J., Thompson, Evan, and Rosch, Eleanor. *The Embodied Mind*. MIT Press, 2003.

[138] Watkins, C. J. C. H., and Dayan, P. Q-Learning. *Machine Learning 8*, 3 (1992), 279–292.

[139] Wikipedia. Intel. http://www.intel.com/technology/computing/opencv/.

[140] Wikipedia. Microsoft kinect. http://en.wikipedia.org/wiki/Kinect, 2011.

[141] Xu, Gang, and Zhang, Zhengyou. *Epipolar Geometry in Stereo, Motion and Object Recognition*. Kluwer Academic Publishers, 1996.

[142] Xu, L., Oja, E., and Kultanen, P. A new Curve Detection Method: Randomized Hough Transform (RHT). *Pattern Recognition Letters 11*, 5 (May 1990), 331–338.

[143] Yamazaki, Kimitoshi, Tomono, Masahiro, and Tsubouchi, Takashi. Picking up an Unknown Object through Autonomous Modeling and Grasp Planning by a Mobile Manipulator. In *Proceedings of the International Conference on Field and Service Robotics* (Chamonix, France, July 2007), Christian Laugier and Roland Siegwart, Eds., vol. 42 of *Springer Tracts in Advanced Robotics*, Springer, pp. 563–571.

[144] Yan, Jingyu, and Pollefeys, Marc. Automatic Kinematic Chain Building from Feature Trajectories of Articulated Objects. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (Washington, DC, USA, 2006), vol. 1, IEEE Computer Society, pp. 712–719.

[145] Zappella, Luca. Motion Segmentation from Feature Trajectories. Master's thesis, University of Girona, Girona, Spain, 2008.

[146] Zuliani, M., Kenney, C. S., and Manjunath, B. S. The Multiransac Algorithm and its Application to Detect Planar Homographies. In *International Conference on Image Processing* (Genova, Italy, September 2005), IEEE Computer Society.