# Talking to Robots: Learning to Ground Human Language in Perception and Execution

Cynthia Matuszek

A dissertation submitted
in partial fulfillment of the
requirements for the degree of

Doctor of Philosophy

University of Washington

2014

Reading Committee:

Dieter Fox, Chair

Luke Zettlemoyer, Chair

Rajesh Rao

Program Authorized to Offer Degree:
UW Computer Science and Engineering

University of Washington

**Abstract**

Talking to Robots: Learning to Ground
Human Language in Perception and Execution

Cynthia Matuszek

Co-Chairs of the Supervisory Committee:
Dieter Fox
Computer Science and Engineering

Luke Zettlemoyer
Computer Science and Engineering

Advances in computation, sensing, and hardware are enabling robots to perform an increasing variety of tasks in progressively fewer constraints. It is now possible to imagine robots that can operate in traditionally human-centric environments. However, such robots need the flexibility to take instructions and learn about tasks from nonspecialists using language and other natural modalities. At the same time, physically grounded settings provide exciting opportunities for language learning. This thesis describes work on learning to acquire language for human-robot interaction in a physically grounded space. Two use cases are considered: learning to follow route directions through an indoor map, and learning about object attributes from people using unconstrained language and gesture.

These problems are challenging because both language and real-world sensing tend to be noisy and ambiguous. This is addressed by reasoning and learning jointly about language and its physical context, parsing into intermediate formal representations that can be interpreted meaningfully by robotic systems. These systems can learn how to follow natural language directions through a map and how to identify objects

from human descriptions, even when the underlying concepts are novel to the system, with success rates comparable to or defining the state of the art. Evaluations show that this work takes important steps towards building a robust, flexible, and effective mechanism for bringing together language acquisition and sensing to learn about the world.

# TABLE OF CONTENTS

# LIST OF FIGURES

# ACKNOWLEDGMENTS

I have the superlative good fortune of having far too many people who deserve acknowledgements in this section. From the very beginning of this journey, I have been surrounded by truly remarkable mentors, colleagues, family, and friends, without whom my research and my life would have been poorer.

Working with my adviser, Dieter Fox, has been an extraordinary privilege. He is one of the quickest-witted people I have ever known, springboarding from any problem or conversation to a series of insights and solutions; I have had to learn to keep up in self-defense. His perceptiveness into the research community and joy in tackling intimidatingly disconnected problems have made me not only a researcher but a better member of the academic community. His drive spills over onto his advisees, but so does his humor and his warmth, and it has made working with him a joy as well as a success.

I am equally privileged to have worked with my other adviser, Luke Zettlemoyer. His open door, keen interest in any problem brought before him, and patience with the vagaries of research (and grad students) have let me explore interests I would not otherwise have known how to approach or dared to risk. I am a better writer, presenter, and (I hope) teacher after working with him; furthermore, his support, candor and clarity have made grad school and everything that follows it easier to understand and, again, a warmer place.

I'd also like to acknowledge the patience it took to have me as an advisee for all these years: Dieter and Luke both deserve medals. Thank you both for getting me here.

Aside from my mentors, the students who have been my colleagues and friends are my greatest source of ideas, feedback, and inspiration. Especially: to Karl Koscher, for knowing everything about the department and for years of friendship and support; to Matt Kay, for always being willing and able to provide feedback and brainstorming, and for shedding interesting ideas the way some people lose pens; to Nicholas FitzGerald, for vast and continuing help in my effort to combine natural language and robotics, and for a lifetime (I hope) of horrible puns; to Marc Deisenroth, for your level-headedness and support; to Natalie Linnell, for making the transition to grad school not only bearable, but vastly entertaining.

Both the RSE lab and the LIL lab are filled with great people. For the shared ideas, brainstorming, conferences, practice talks, lunches, and generally the last few years of my research life: thank you. Nobody could ask for better academic siblings. Many of you are also the next group who made it possible: my co-authers, every single one of whom has provided insight, ideas, and a breadth of skills and experience that has made my academic experience broader and my research better. The University of Washington's computer science department has been an amazing home, and you guys are the very best of it.

Apart from my colleagues, the people in my life deserve gratitude beyond what I can fit here. To Jennifer and Casey Cooper, Mike and Amanda Krainin, Matt and Caitlin Taylor, and too many other people to list, your companionship, encouragement, support, and occasional reminder that grad school isn't everything has kept me sane. Thank you.

Finally and especially, I am grateful beyond anything I could write for my family's support from the very beginning. To my parents, for everything from teaching me Prolog to proof-reading thesis chapters at midnight, and every-

thing in between; and to my siblings, for always being entertaining, inspiring, supportive people who are there for me. And finally, to my husband Toren, without whom none of this would have been possible.

I have inevitably omitted people here, because there are too many wonderful people who have been part of this process. Thank you all.

# DEDICATION

*To my family, who make me who I am, and*
*to Toren – I couldn't have done it without you.*

# Chapter 1

# INTRODUCTION

> "It can also be maintained that it is best to provide the machine with the best sense organs that money can buy, and then teach it to understand and speak English. This process could follow the normal teaching of a child. Things would be pointed out and named, etc." – A. M. Turing, *Computing Machinery and Intelligence*

Understanding natural language has been a core element of artificial intelligence since the inception of the field. Using language to conceptualize and communicate about the world is so innate to human intelligence that the only reference to it in the seminal paper *Computing Machinery and Intelligence* [Turing 1950] is in the conclusion, quoted above. The suggestion that, to create intelligence, we *begin* with language exposes the deep-rooted assumption that linguistic abilities are part of the underlying cognition of an intelligent machine.

There is a similarly long history of believing that robots capable of displaying human-level intelligence will be available in the near-term; automata such as da Vinci's mechanical knight, the original Mechanical Turk, and the protonymous robots of *Rosumovi Univerzální Roboti* [Capek et al. 1920] demonstrate the degree to which robots have captured human fancy. In practice, of course, both natural language processing and interactive robotics are among the most complex fields of study in artificial intelligence today. In this dissertation, I argue that outstanding problems in both areas can be addressed by combining state of the art techniques from each.

The main thesis of this research is that *grounded language acquisition*—taking linguistic tokens and learning to interpret them by connecting them to real-world percepts and actions [Mooney 2008]—can improve the efficiency and efficacy of both natural language understanding and robotics, by allowing them to use machine learning approaches to learn jointly from each other. Intuitively, the idea is that language can be better learned when presented and interpreted in the context of the world it pertains to, and robots can learn to be more useful and more flexible when language is used to describe and disambiguate the noisy, unpredictable world in which they operate.

More specifically, the work described here is on using interaction with the physical world to support the learning of language pertaining to that world, which in turn provides a mechanism by which robots can learn to understand instructions and statements given in the context of the physical world. This thesis is supported by the systems described in the following chapters, in which robots learn progressively more complex ideas and interactions from progressively more complex language and interactions.

## *1.1 Motivation and Goals*

Physically grounded settings provide exciting opportunities for language learning. At the same time, as robots become more capable, there are an increasing number of scenarios and environments in which they may be deployed usefully, meaning that the importance of enabling untrained users to interact with them in a natural way increases. However, before we can build practical, useful domestic or personal robots, a number of challenges and shortcomings in the current technology must be addressed. It is becoming increasingly clear that robots in such settings will need to be able to interact with non-expert users in natural ways, rather than requiring such users to master complex interfaces.

These concerns fit in a larger context of understanding how to interact gracefully and safely with humans in a shared workspace. A system that incorporates these capabilities is a necessary step towards eventual real-world situations in which flexible, inexpensive robots will be of use in assistive technology settings, the workplace, and the home.

## *1.2 Research Trajectory and Contributions*

The research described here is divided into three main categories: learning to follow instructions, learning about things in the world, and learning to interact with people. These three core capabilities all provide substantial contributions to the goal of building a robot that can be usefully deployed in such environments. To be useful to an end user, robots must do what they are told, and they must do so in environments and using objects that may be unfamiliar; they must also understand their users, who may engage in a rich and idiosyncratic range of communications. Robots that use learning to extend their understanding of these interactions are better equipped to adapt to the needs and abilities of end users.

*Learning to follow task instructions.*

Learning to follow directions through an indoor map is used as a platform for several reasons. First, it is a problem of independent interest in robotics and AI in general. More importantly, following natural-language route instructions through a building is a challenging, error-prone activity. Maps of spaces are often incomplete or inaccurate, the variety of ways a single path can be described is large, and people are surprisingly poor at giving directions. Nonetheless, giving and following directions is a common task encountered in day-to-day life.

Much of the work on grounded language acquisition assumes an existing world model—a fully labeled map, an environment completely populated with labeled objects, and

so on. The research goal is then to map language, particularly instructions, to actions in this model.

In practice, human language understanding allows for the interpretation of instructions into high-level concepts, which are not fully collapsed into groundings until the model is discovered. If someone gives instructions to go "until you reach an intersection," the high-level concept of the `until` loop is not collapsed into a specific number of steps until the intersection is seen. Work is described here that, by taking advantage of an appropriate layer of abstraction between the parsed meanings of language and the grounding of concepts into execution, navigates successfully through a map that is discovered simultaneously with navigation rather than provided in advance.

In the context of robotics, while logic-based systems have been used to provide a framework for robot control, most approaches have historically relied on a manually constructed parser rather than learning grounding relations from data. The work presented here avoids this simplification. Grounding relationships between natural language instructions and formal representations are learned entirely from data in all cases, including the learning of higher-level control structures such as 'while,' higher-order concepts such as '$n^{th}$,' and set operations.

*Learning about things in the world.*

Not only with respect to specific instructions, deployed robots will need to be able to learn about the world around them more generally, rather than being pre-programmed with a fixed set of tasks or beliefs about the world. Agents in the real world are exposed to new observations continuously, and will need to adapt to new situations and tasks that cannot be envisioned in advance. A potentially well focused way to learn about salient objects is from a human teacher.

Learning about objects is a core language grounding task. However, in general, what is learned is a mapping from language to an existing formal language defined by a

grammar. Generating novel symbols to functionally describe new concepts dates (at least) to the specification of LISP in the 1950s; however, to the best of my knowledge, the work in Chapter 6 represents the first effective approach to extending a formal representation based on a combination of novel language and percepts.

That work presents an approach for the efficient learning of joint visual classifiers and semantic parsers, to produce rich, compositional models that span directly from sensors to meaning. This capability is a key element of a system that can learn entirely from language in an unpredictable world.

*Learning how to interact with people.*

Before learning *from* human input, it is necessary to learn *about* human input. A robot that is able to learn the semantic intent of linguistic constructs from inevitably noisy input will need to: Be able to incorporate sensory data when learning how to interpret world objects and interactions; participate in its own learning productively; and, not least, interact gracefully with human partners in loosely constrained environments. Tasks such as learning to understand unscripted deictic gesture are a necessary element of such interactions, as is appropriate interactive feedback.

There exists work on robotically generating understandable, appropriate gestures, as well as on integrating very constrained natural language with deictic gesture, and on how robots may be controlled using gesture. However, it is almost universally in the area of *gesture recognition*—that is, disambiguating which of several predefined gestures is being made.

Rather than asking users to memorize such a lexicon of gestures, the system presented in Chapter 7 learns to perform sensor-based interpretation of *any* deictic (indicative) gesture, using a novel technique for unsupervised learning of features rich enough to enable accurate recognition. A high-level object indication model is trained from video data that combines language, gestures, and visual attributes to train a recognition

model, which supports a prototype implementation that can be instructed using voice and gesture.

### *1.3 Organization of Dissertation*

The rest of this document is organized as follows. Chapter 2 presents an overview of some of the foundational and state of the art work in relevant areas, including natural language understanding and robotics application domains. Chapter 3 describes the development and testing of a manipulation platform used for the research in later chapters. Chapter 4 and Chapter 5 describe two different approaches to inducing semantic parsers capable of learning to follow natural language task instructions, both of which use natural language route directions through an indoor map as a testbed; Chapter 6 extends those techniques to learning to ground descriptions of physical objects to sensor data. Chapter 7 describes the collection and interpretation of a data set of teaching interactions about those physical objects, and Chapter 8 presents an initial feasibility study of how robots can best provide feedback that improves the users' experience while teaching. Finally, Chapter 9 concludes with high-level contributions and ongoing work.

# Chapter 2

# RELATED WORK

This thesis describes the development of systems that perform grounded language acquisition in a physical context—robots that learn the meanings of human language as it relates to the perceived world. This is a fundamentally multidisciplinary task, relying on related work from from cognitive science, computational linguistics, robotics, and human-computer interaction. In addition, many of the problem domains described, such as navigation and game-playing, have historically been of interest to artificial intelligence research in general.

The discussion of work in each category further breaks work down according to its area, goals, and scope. This chapter is not meant to be a comprehensive overview of these fields. Instead, in each category, this chapter includes a selection of both foundational and current papers, intended both to provide context for the work presented and to allow comparison to state of the art research.

## *2.1 Grounded Language Acquisition*

Like natural language understanding, the understanding of symbols and symbolic reasoning have been a core element of artificial intelligence since the beginning of the field. Researchers in AI, psychology, linguistics, and cognitive science have all studied aspects of grounded language acquisition, also called the embodied language problem and situated language acquisition, among other things. In AI, the more general term is the *s*ymbol grounding problem, after Harnad [Harnad 1990].

The concept of grounding symbols in some external reality became an explicit topic of conversation after Newell and Simon described the physical symbol system hypothesis [Newell and Simon 1976]; the discussion intensified when Searle defined the Chinese Room experiment [Searle 1980], describing the lack of grounding as an inability of then-current systems to "understand" concepts. More recently, in "Learning to Connect Language and Perception," Mooney argues that advances in AI make this an appropriate time to consider integrative grounded language learning [Mooney 2008], and current successes in using weakly supervised learning to support situated language learning over large corpora of data support the idea [Matuszek et al. 2012a; Tellex et al. 2014b].

The language learning component of this work fits into the category of grounded language acquisition: The learned interpretation of human language into semantically informed structures in the context of perception and actuation, often by learning over a corpus of parallel language and contextual data. The type of context in which language is situated can take many forms; examples range from generated sportcasts of simulated soccer games [Chen et al. 2010], to generating [Thomason et al. 2014; Guadarrama et al. 2013] and searching on [Tellex et al. 2010] linguistic descriptions of spatial elements in video clips, to GUI interaction [Branavan et al. 2010].

These approaches ground language into pre-defined language formalisms. There has been some work in matching or extending an underlying ontology during semantic parsing for question answering [Kwiatkowski et al. 2013; Berant et al. 2013; Berant and Liang 2014] and textual entailment analysis [Beltagy et al. 2014]; however, extending the underlying grammar model to account for and learn about entirely novel input (as in Chapter 6) is a contribution of this dissertation, later elaborated on by [Krishnamurthy and Kollar 2013].

## *2.2 Semantic Parsing*

One approach to interpreting language for robotic interfaces is to transform natural language to expressive formal representations, which can be queried, reasoned over, and sanity-checked. There has been significant work on supervised learning for inducing semantic parsers [Zelle and Mooney 1996; Zettlemoyer and Collins 2005; He and Young 2006; Wong and Mooney 2007], which parse English to such formally defined representations.

A number of semantic parsing systems and underlying representation systems have been studied and expanded on. In Chapter 4, the underlying parsing induction system is WASP, a Probabilistic Synchronous Context-Free Grammar (SCFG) based statistical parser [Wong 2007; Wong and Mooney 2006]; the remaining chapters depend on parser induction systems built over Steedman's Combinatorial Categorial Grammar (CCG) [Steedman 1996] formalism.

This allows the semantic parsing of natural language into representations such as $\lambda$-calculus [Zettlemoyer and Collins 2005, 2007], which is able to represent complex robot control systems [Dzifcak et al. 2009]. This research builds on work on weakly supervised induction of CCG parsers [Kwiatkowksi et al. 2010; Kwiatkowski et al. 2011], which have been used to support other forms of contextual language learning.

There is also work on performing semantic analysis with alternate forms of supervision, including questions paired with database answers [Clarke et al. 2010; Liang et al. 2011; Kwiatkowski et al. 2013], conversational logs [Artzi and Zettlemoyer 2011], distant supervision [Cai and Yates 2013], and sentences paired with system behavior [Chen and Mooney 2011; Goldwasser and Roth 2014; Artzi and Zettlemoyer 2013]. With the exception of [Krishnamurthy and Kollar 2013], which adapts the model described in Chapter 6 to use the output of visual classifiers to build a simple ontology before performing queries, none of these approaches incorporate joint models of lan-

guage and vision; this is a contribution of the work presented in Chapter 6 and Chapter 7.

## *2.3 Robotics Applications and Domains*

The primary focus of the work presented in this dissertation is the understanding of grounded language and human-robot interaction in a variety of distinct application domains, each of which is in itself an area of study and interest for robotics and artificial intelligence researchers. This section briefly describes relevant background work in those application domains.

### *2.3.1 Navigation, Following Directions, and Spatial Language*

Robot navigation is a critical and widely studied task in mobile robotics, and this thesis uses learning to follow natural language route directions as a testbed for learning to understand human instructions when performing a task, as following instructions is a key component of natural, multi-modal human/robot interaction [Skubic et al. 2001]. There are many proposed mechanisms for reasoning about natural language directions, largely relying on formal representations of such instructions, some of which are learned from examples and some of which are hard-coded. These representation systems vary widely in expressiveness, provable correctness and consistency, and directness of robot control.

There has been substantial work on mapping and localization [Thrun et al. 2005], segmenting and describing a map from sensor data [Friedman et al. 2007; Gutmann et al. 2008], and navigating through such an environment [Ratliff et al. 2006; Ziebart et al. 2008]. While many previous efforts have treated the language grounding task as a problem of parsing commands into formal meaning representations, without such exploration and mapping, natural language route instructions can only be parsed to

sequences of atomic actions that must be grounded into fully specified world models [Macmahon et al. 2006; Kollar et al. 2010; Wei et al. 2009; Vogel and Jurafsky 2010; Matuszek et al. 2010] (i.e., a known map). A robot otherwise lacks the physically situated context to ground instructions.

Logic-based control systems have been used successfully in robotics [Hähnel et al. 1998; Boutilier et al. 2000; Burgard et al. 1999; Ferrein and Lakemeyer 2008; Kress-Gazit et al. 2011], providing a powerful framework that can be readily mapped to robot actions; however, most approaches rely on a manually constructed parser to map from NL commands to $\lambda$-calculus, rather than learning grounding relations from data. The work presented here avoids this simplification. Grounding relationships between natural language instructions and formal representations are learned entirely from data in all cases, including the learning of higher-level control structures such as 'while,' higher-order concepts such as '$n^{th}$,' and set operations.

Macmahon [Macmahon and Kuipers 2007] studied the inference requirements and language used for natural-language navigational instructions in complex simulated environments, providing both a widely used data set and insight into human directional language, including experimental evidence of the poor quality of instructions provided by human direction-givers. The underlying semantically annotated model of the world used in these trials is more complex than is realistically available to existing sensors and object recognition systems, and example instructions are annotated with a semantic grammar based on the semantically rich environment; the system described also relies on human-supplied heuristics for managing missing or implied knowledge. Artzi et. al [Artzi and Zettlemoyer 2013] later use this simulated environment in a parser-induction based navigation system.

The work described in Chapter 4 is similar to [Shimizu and Haas 2009] with respect to the representation of the path description layer and map layer, but does not rely on labeled training data or map-based natural language segmentation. Much other work

on language understanding in the navigation context takes a more direct semantic analysis of instructional language, as in [Dzifcak et al. 2009], although the formal representation extracted from instructions defines a sequence of possible groundings rather than a set of action and goal representations defined over a hand-annotated map. The semantic parsing in this dissertation is not limited to nouns and explicit direction terms, as it is trained over more general natural language.

Chen & Mooney [Chen and Mooney 2011; Chen 2012] perform parser learning over a body of route instructions through a complex indoor environment containing objects and landmarks with no prior linguistic knowledge. However, their work, as above, assumes initial knowledge of a map, and does not attempt to represent complex control structures such as 'while,' higher-order concepts such as ordering, and set operations.

The work on instruction following described in Chapter 4 and Chapter 5 presents several contributions to the state of the art. The relationships between world concepts and language is learned entirely from data, rather than relying on a pre-specified mapping from language to $\lambda$-calculus robot commands, or on any human-provided heuristics for interpreting language. More crucially, Chapter 5 describes a system in which the grounding context—here, a map of an indoor environment—is discovered during language understanding. This is a significant contribution, as it avoids the common simplifying assumption that the robot is provided with a fully specified world model in which to act. More recent work [Duvallet et al. 2013] takes a similar approach to following directions sequentially through partially known environment, learning from example human routes provided in an imitation learning framework.

### *2.3.2 Robotic Manipulation and Game Playing*

While mobile robotics is an active area of research, human-robot interaction, human-robot collaboration, and assistive technology all depend at least equally on manipu-

lation. To this end, Chapter 3 describes the development of a manipulation platform, which is used for subsequent object and HRI work. This platform, the Gambit small-scale manipulation platform, was developed and tested in a game-playing setting: specifically, playing chess with a human opponent.

While the game-play aspects of chess were long considered a major target of traditional AI and cognitive research,[1] the manipulation aspects of playing a game with arbitrary boards, pieces, and opponent movements proved to be at least as difficult, and in practice possibly much more so.

Nearly all autonomous board game playing systems use instrumented or special purpose game boards and pieces to simplify perception and manipulation. Thus, they do not handle the complexities introduced by using arbitrary game sets with arbitrary human interactions. To achieve unrestricted interaction, these assumptions must be relaxed. For example, chess playing robot arms were shown in 2010 at the Maker Faire,[2] and at several press events in Russia [Bratersky 2010]. They used instrumented chess sets such as the Digital Game Technology Sensory Chess Board[3] to eliminate the perception problem.

Game playing has the advantage that the interaction is limited to a known set of turn-taking actions within a physically defined board area. In the human-robot interaction literature, turn taking has been studied extensively in several contexts, including speech games [Chao et al. 2011], social speech interactions [Breazeal and

---

[1]The Mechanical Turk, first exhibited in 1770, is perhaps the first purported chess playing automaton. It in fact relied on a concealed human chess master for game logic and control of the Turk's manipulation hardware; magnetically instrumented chess pieces allowed sensing and manipulation of pieces via the motion of corresponding magnets in the operator's compartment [Schaffer 1999; Standage 2002].

[2]`http://www.chessplayingrobot.com`

[3]`http://www.digitalgametechnology.com`

Scassellati 2000], drumming [Kose-Bagci et al. 2008], and game play for autism therapy [Dautenhahn and Billard 2002]. The contribution described in Chapter 3 is a complete system that uses no instrumentation of board or pieces, autonomously detects human moves, can play with a wide variety of sets, and is generally capable of engaging in a robust game with fewer constraints than previous systems.

### *2.3.3 Perception and Robot Vision*

Current state of the art object recognition systems [Felzenszwalb et al. 2009] are based on local image descriptors, for example SIFT over images [Lowe 2004] and spin images over 3D point clouds [Johnson and Hebert 1999]. Visual attributes provide rich descriptions of objects, and have become a popular topic in the vision community [Farhadi et al. 2009; Parikh and Grauman 2011]; although very successful, we still lack a deep understanding of the design rules underlying them and how they measure similarity.

Recent work on kernel descriptors has demonstrated their usefulness in image recognition tasks. Classifiers based on Efficient Match Kernels (EMKs) [Bo and Sminchisescu 2009b], which form image-level features from low-level, hand-crafted SIFT features, demonstrated state of the art performance on object recognition benchmarks, performing similarly to sparse coding [Yang et al. 2009; Yu et al. 2009] and deep networks [Lee et al. 2009].

[Bo et al. 2010] subsequently demonstrate that low-level image features can be learned from scratch using kernel methods, rather than relying on hand-crafted features. These kernel descriptor approaches have proven to be appropriate in practice for object recognition [Bo et al. 2011b; Lai et al. 2011], feature learning, and attribute identification [Bo et al. 2012; Sun et al. 2013] tasks.

For RGB-D video and image collection, the widely available Microsoft Kinect sensor

is used almost exclusively. While low resolution, it provides enough information to extract features capable of successful object recognition support on a very large data set of objects [Lai et al. 2013]. Object identification and grounded understanding take advantage of state of the art work in the computer vision community. In particular, the contributions presented in Chapter 7 on analyzing attributes and gestures uses an expansion of Hierarchical Matching Pursuit features, which had previously shown state of the art performance on static problems such as image classification [Bo et al. 2011b].

## *2.4 Situated Language in Robotics*

Robotics is a natural source of non-simulated contextual grounding for language learning, and there has been significant work on using grounded learning more generally in the robotics and vision communities. Roy developed a series of techniques for grounding words in visual scenes [Mavridis and Roy 2006; Gorniak and Roy 2003; Reckman et al. 2010]. In computer vision, the grounding problem often relates to detecting objects and attributes in visual information (e.g., see [Barnard et al. 2003]); however, these approaches frequently focus on isolated word meaning, rather than compositional semantic analyses.

Semantic mapping of environments—the labeling of indoor spaces with human-meaningful concepts, such as "hallway" or "kitchen"—is an important aspect of understanding human commands [Zender et al. 2008]. Interpreting natural language utterances as descriptions of an environment, as in language-driven semantic mapping, integrates sensor data and linguistic information to improve the accuracy of a robot's understanding of an environment [Walter et al. 2013, 2014].

To address the uncertainty inherent in working with both language and vision, the work described in this dissertation combines learned probabilistic models of language

with probabilistic world models. Other work shows how parsed natural language can be grounded in a robot's world and action models, taking into account perceptual and grounding uncertainty [Tellex et al. 2011] or natural language ambiguity [Chen and Mooney 2011].

Another approach to grounding words in context is to focus on the meaning of language in a specific context rather than generally, such as learning specific referents in a tabletop setting [Mohan et al. 2013]. These terms can be incorporated into a cognitive architecture intended to support instruction following [Mohan and Laird 2014]; in this view, the symbol grounding problem is treated as one in which symbols are grounded in a shared human/robot context, rather than in a physical world which may or may not be viewed by a person.

A shared situated language allows for another important collaborative feature: the generation of language pertaining to an environment, which allows for behaviors such as using dialogue to negotiate a shared representation of the world [Chai et al. 2014]. A robot that can express information about its own world beliefs can negotiate such a representation despite the imbalance of detail about the world perceived by a robot and a human team-mate [Fang et al. 2014].

This approach allows for effective collaboration [Kiesler 2005; Kruijff 2013], especially in human-robot teaming applications [Kruijff et al. 2012]. In addition, a robot with the ability to describe a shared world can interactively clarify ambiguous commands about the world [Deits et al. 2013] and ask for help in a meaningful fashion [Tellex et al. 2014a].

## *2.5 Human-Robot Interaction using Language and Gesture*

Teaching robots about objects and attributes of objects in the world is an active target in human-robot interaction (HRI) research [Rouanet et al. 2013], and under-

standing how people refer to objects in the world is an important element of natural HRI [Sandygulova et al. 2012]. Meanwhile, natural language, while able to encode substantial complexity, is still an incomplete communication mechanism in a physical setting, where it is often natural to use gesture to focus attention [Topp et al. 2006]. (Compare, for example, "Put the mug that's on the table in the cupboard two to the left of the stove," versus "Put this mug in there.") As a result, there is long-term interest in interfaces that incorporate these modes of interaction [Bolt 1980; Jain et al. 2011], especially for human-robot interactions [Perzanowski et al. 2001].

The study of appropriate interfaces and feedback for interacting with robots has also received substantial study, both in the context of teaching interactions [Cakmak et al. 2010; Knox et al. 2013] and for tasks such as human-robot teaming [Harris and Barber 2014], providing lessons about the value of transparency and responsiveness in HRI. Human-robot collaboration and joint activity have been discussed in [Breazeal et al. 2004], which focuses on socially expressive and natural communication via expressive robots (i.e. robots with many facial and gestural degrees of freedom), rather than board game playing specifically or even co-manipulation generally.

Both gesture and language have been studied extensively as possible elements of human-robot interaction [Goodrich and Schultz 2007; Mitra and Acharya 2007]. Gesture has been found to be a comfortable, effective modality for teaching robots [Rouanet et al. 2011], and there is substantial work on gesture recognition in robotics [Wachs et al. 2011], including those using low-cost RGB-D sensors [Ramey et al. 2011] and those using temporal or trajectory-based recognition [Nickel and Stiefelhagen 2007; Yang et al. 2007]. Existing gesture interfaces have a user cost, in that they are primarily focused on gesture *recognition*—that is, focused on identifying a lexicon of gestures which the user must learn [Malizia and Bellucci 2012]. The work presented in Chapter 7) contributes a novel mechanism by which systems can learn to understand unscripted human gesture, avoiding the overhead of user training.

### *2.5.1 Cognitive Models, Human Language Acquisition, and Deixis*

While this dissertation focuses on language acquisition by way of building joint language and world models, there exists substantial work on learning situated meanings for symbols from a more emergent, cognitive approach. One way to study embodied acquisition of language is to consider how infants learn symbols and the meanings of gestures [McGregor et al. 2009; Vollmer et al. 2009]; another is to apply the same evolutionary concepts of language to a robotics platform and study the resulting behavior [Steels 2001; Steels and Loetzsch 2012]. While these approaches offer the potential of eventually reaching human-level language learning and understanding, they do not seek to address the near-term question of building a more special-purpose, explicitly robotic learner.

Teaching robots about objects and attributes of objects in the world is an active target in human-robot interaction (HRI) research. While the specific tasks targeted in this dissertation uses a limited set of objects and attributes, understanding how people refer to objects in the world is an important element of natural HRI [Sandygulova et al. 2012], and has been studied from the perspective of both human/robot dialogue [Peltason et al. 2012] and vision [Farhadi et al. 2009, 2010].

In terms of understanding human intention beyond descriptive language, the work in this dissertation focuses on *deictic* language and gesture—specifically, gestures that Clark [Clark 2003] calls *directing-to*, which are attention-focusing. While there exists work on *generating* understandable, appropriate gestures [Liu et al. 2013] and on integrating very constrained natural language with deictic gesture [Perzanowski et al. 2001] in robotics, none of this existing work focuses on sensor-based interpretation of unrestrained deictic gesture.

# Chapter 3

# A HUMAN-ROBOT INTERACTION PLATFORM

In grounded language acquisition, and the symbol grounding problem generally, there are a number of possibilities for what non-symbolic world symbols or language are grounded into. As described in the previous chapter, the choice of worlds by different researchers have ranged from conceptual terms in an ontology, to elements of a user interface, to simulated and real robots in the world.

Throughout this dissertation, language is grounded into real physical systems that can be perceived by non-simulated robots. This adds a definite element of complexity to the problem, requiring, as it does, interpretation of the noisy, probabilistic physical world. However, as described in the introduction, that complexity also offers opportunities to improve both robotics and language grounding by treating them as a jointly strengthening learning problem.

This chapter describes the development and testing of the Gambit robot platform, whose capabilities underlie the majority of the research into language grounding in a physical workspace/manipulation setting described in subsequent chapters. The small-scale manipulation task—playing a game of chess—that was used to establish its usefulness for human-robot interactions is described in detail.

---

The work described in this chapter was originally presented at ICRA 2011 [Matuszek et al. 2011], and was performed in collaboration with Brian Mayton, Roberto Aimi, Liefeng Bo, Marc Deisenroth, Robert Chu, Mike Chung, Louis LeGrand, Joshua R. Smith, and Dieter Fox, without all of whom an undertaking of this scale would not have been possible.

### *3.1 Motivation and Overview*

To deploy robots human-centric environments, it is necessary to relax many of the constraints that are often used in experimental settings. Robots in a home cannot expect to be surrounded by instrumented objects that conform to certain physical constraints, lit by carefully controlled lighting. The primary goal of the work described in this chapter was building and demonstrating the real-world flexibility of a small-scale manipulation robot.

Progress on tabletop manipulation paves the way for more general human-robot cooperation systems that assume less structure. For example, this line of work could lead eventually to a manipulator capable of helping a chemist as a lab assistant that cooperatively performs manipulation tasks on an unstructured laboratory bench-top, or of helping an elderly user manipulate small medication dispensers.

Section 3.2 presents Gambit, a custom, mid-cost 6-DoF robot manipulator system that can play physical board games against human opponents in non-idealized environments. Historically, unconstrained robotic manipulation in board games has often proven to be more challenging than the underlying game reasoning, making it an ideal testbed for small-scale manipulation. The Gambit system includes a low-cost Kinect-style visual sensor, a custom manipulator, and appropriate learning algorithms for automatic detection and recognition of the board and objects on it. Chess is used as a testbed, and subsequent sections demonstrate that Gambit can play chess quickly and accurately with arbitrary, uninstrumented boards and pieces.

Physical board games are a rich problem domain for human-robot cooperation research because such games have an intermediate and easily adjustable degree of structure. Playing board games involves perception of the board and game pieces, perception of the human, reasoning about the game and game state, and manipulation of the physical pieces while coordinating with the human opponent. In later chapters,

Figure 3.1: Gambit playing chess autonomously against a child.

substantial elements of this perception and manipulation are applied to more difficult tasks of learning about object attributes and interacting with objects as verbally directed, demonstrating how games can provide an extensible starting point for this type of human-robot interaction.

Gambit can play with arbitrary chess sets on a variety of boards, requiring no instrumentation or modeling of pieces. It monitors the board state continuously and detects when and what kind of move an opponent has made. Furthermore, it conveys its state to a human opponent using a natural (scripted) spoken-language interface.

The robot's perceptual system tracks the board pose and the human opponent in real time. The board is not fixed relative to the robot and is continuously calibrated during game play. and uses ordinary (even previously unseen) boards and pieces designed for human players. This allows for smoother, more natural gameplay.

### *3.2 Manipulator and Sensor Suite Design*

The Gambit system includes a completely new arm, the design of which is open source. The decision to create a new arm design was driven by several factors. Most

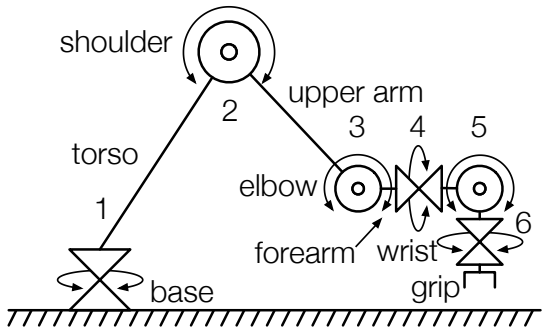



Figure 3.2: Schematic of the manipulator arm's degrees of freedom.

important, for purposes of the research described in this dissertation, was the need to be able to implement smooth, natural, sensor-driven motions and interactions (such as visual- or E-Field-servoing [Mayton et al. 2010]), or to implement programmable compliance. This required complete control of the robot, down to low-level firmware and short timescales.

### *3.2.1 Mechanical Design*

The Gambit manipulator consists of a 6-DoF arm with a parallel jaw gripper. The arm's DoFs are illustrated schematically in Figure 3.2. The three revolute DoFs (1, 2, and 3) provide position control, while DoFs 4, 5, and 6 provide orientation control via a series roll-pitch-roll spherical wrist. The "torso," the structure between the base and shoulder, is offset from the base rotation axis; this torso "lean" reduces the torque requirements of the shoulder joint. The tabletop working area is a circle with a radius of $\approx$ 60cm.

To construct a 6-DoF robot with reasonable cost, precision of position was prioritized over orientation. The three position DoFs use Harmonic Drive FHA series integrated motors, zero backlash gearboxes, and 0.0018-degree encoders, driven by Copley Con-

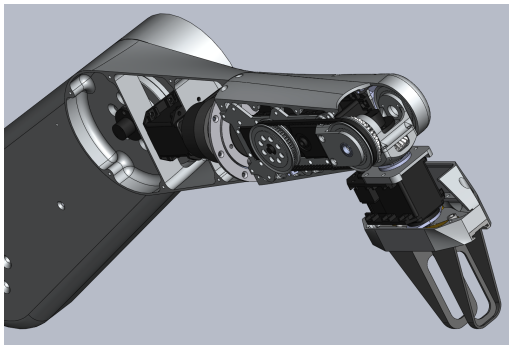


Figure 3.3: Cross sectional view of Gambit forearm with covers removed.

trols servo amplifiers. These actuators are hollow and use double needle bearings to support significant cantilever loads, simplifying joint design and cabling. The wrist uses three RX-28 Dynamixel actuators with a much lower resolution of 0.29 degrees. The Dynamixels have an integrated controller, yielding a much smaller overall package.

To achieve a spherical wrist, DoFs 5 and 6 use timing belts to offset the servo motor from the DoF, as illustrated in Figure 3.3. DoF 6 uses a single hollow miter gear pair to achieve a 90-degree turn. In order to hold DoF 6 still while rotating DoF 5, both motors need to rotate. A differential design was considered but rejected because of limited space for cabling and slip ring. Slip rings are located on all necessary joints in order to allow continuous rotation.

Twelve conductors are wired to the gripper, four for a USB camera centered in the gripper, four for the gripper servo, and four are reserved for future use. Gambit uses a simple parallel jaw gripper design with a Dynamixel RX-10 in a double crank mechanism. The gripper jaws are easily replaceable to adapt to specialized tasks, and the entire gripper assembly can be retrofitted if another type of gripper is required.

For picking up chess pieces, a hollow gripper jaw was designed design with a rubber

finger tip (normally used to aid people in collating papers) stretched over the frame, forming a compliant "opposing trampoline" structure that conforms to objects and has a slight centering effect.

The structure of the robot is milled from aluminum. The Copley Controls drivers use the structure as a heat sink, eliminating the need for a cooling fan.

### *3.2.2 Sensing*

In addition to joint encoders, Gambit has a shoulder-mounted PrimeSense depth camera (technologically extremely similar to an Xbox Kinect), and a small camera built into the gripper. The PrimeSense camera provides three color channels (RGB) plus depth for each pixel, and has a working range of approximately 0.5m to 5m. This large minimum range presented a design challenge for the system. Since the goal was to develop a standalone, integrated system, the camera is mounted on the arm rather than overhead.

The small camera in Gambit's gripper was originally designed to fit in the bezel of an Apple MacBook. The challenge in integrating this palm camera was to maintain the integrity of its high-speed USB data as it passes through all five of Gambit's slip rings and the electrically noisy environment inside the arm. A USB hub in the robot's forearm boosted the signal after it passed through the two wrist slip rings, and shielded cable was used wherever possible.

### *3.2.3 Driver Software*

The driver software for the Gambit arm is arranged hierarchically, targeting specific hardware at the low levels and providing broader abstraction at higher levels; thus, the drivers above the lowest layer are reusable for other robot arms.

As explained in Section 3.2.1, Gambit uses two different kinds of actuators. DoFs 1, 2, and 3 are addressed via CAN bus. DoFs 4, 5, and 6 use an RS-485 bus. The low level `gambit_driver`, which is built on top of ROS [Quigley et al. 2009], provides a uniform interface to the heterogeneous actuators.

The drivers run on a dedicated Intel Atom net-top PC equipped with CAN and RS-485 PCI cards. This dedicated control PC is not subject to variable load conditions that might interfere with smooth control of the arm. Applications running on a separate computer command the arm using a higher-level driver that provides an `Arm` object, which handles ROS communication with the lower level drivers.

There are two modes of controlling the arm, closed loop and open loop. In closed loop mode, the `Arm` object expects the client application to provide a somewhat regularly timed stream of joint targets, e.g. one sample approximately every 50ms. These targets are sent as ROS messages to the `gambit_driver` which does some smoothing, and commands the actuators to move. Closed loop mode is useful when the motion of the arm is not known in advance (e.g when the motion is sensor driven).

## *3.3 Perception and Manipulation*

This section describes details about the perception and manipulation required for autonomous chess playing. This includes real-time tracking of the location and the board state and detection of an opponent's move. In Section 3.3.2, the actual process of playing a game is described, including details about the manipulation and the interface Gambit uses to communicate.

*Locating the Chessboard*

Locating the chessboard and continuously updating its posture with respect to the camera involves: (1), finding the board itself, and (2), finding the transformation

of the board with respect to the camera. Ignoring the depth information from the camera, corner points on the 2D (RGB) image of the chess board are used to locate the board and the grid of squares, see Figure 3.4. Depth information of the corner points is then incorporated. A plane is fitted to the (now 3D) points using RANSAC [Fischler and Bolles 1981], which yields the plane upon which the board surface lies, but does not uniquely determine the exact board placemen; the best match of all 3D points on the board plane with a template of the $8 \times 8$ contiguous chessboard cells is used to localize the board. This approach is also robust to partial occlusions of corner points (e.g., by a hand or other pieces).

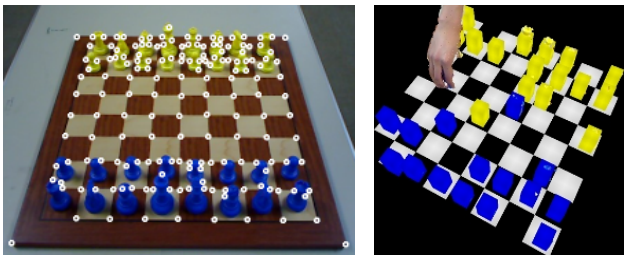


Figure 3.4: (left) Finding 2D board corner points, shown as white circles. (right) Detection of a hand while moving a chess piece.

*Board Occupancy*

To determine if chess squares are occupied, Gambit uses the point cloud consisting of all points above the surface of the board plane. An isometric projection of this point cloud onto the board returns a 2D projection of all pieces onto the board plane. The projected points are clustered, and individual clusters—corresponding to pieces—are assigned to a cell in the $8 \times 8$ *occupancy grid* defining the chess board. When an object (usually a hand) is detected in the area above the board, as in Figure 3.4, the determination of the occupancy grid is paused until the object disappears.

Piece *color* at occupied squares is assigned in two steps: At the beginning of the

game, “black” and “white” are defined as the 2-median colors, $c$, for the point clouds corresponding to all pieces on either side of the chessboard. During the game, Gambit determines the best assignment of the clustered projected point clouds to “black”/ “white”. This procedure returns the current occupancy of the board and the colors of the occupying pieces at 15Hz.

To track the piece *types* on the board, the *board state* is defined as the as a tuple $(s_i, c_i, p_i)_t$, $i = 1, \ldots, 64$, where $s \in \{0, 1\}$ is a binary square occupancy variable, $c \in \{\text{white, black}\}$ is the chess piece color, $t$ describes the time step, and $p \in \{\text{K, Q, P, R, B, N}\}$ for **K**ing, **Q**ueen, **P**awn, **R**ook, **B**ishop, and k**N**ight, respectively, defines the piece type.

Second, the board occupancy and color differences of two consecutive board states are computed. GNU chess uses these differences to check the validity of the change. If the change is invalid, the opponent is asked for correction.

### *3.3.1 Chess Piece Detection and Recognition*

The chess-piece detection and recognition system consists of four hierarchical classifiers: a square detector, a binary piece/background detector, a binary piece color (white/black) recognizer, and two piece-type recognizers, each with the six class labels {B,N,K,P,Q,R}. Figure 3.6 details the main steps for chess-piece recognition during game play given a 640×480 palm-camera image.

*Square Detector*

The square detector is used to find chess squares—independent of rotation and scale—in the 640×480 palm-camera image.

To be robust to rotations of the camera, 20 square templates are generated by rotating a $220 \times 220$ pixel square. From the templates, histograms of oriented gradients (HOG)

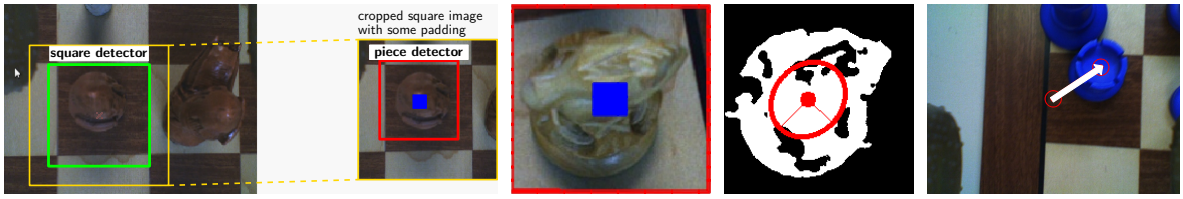



Figure 3.5: From a full image to rotational and translational corrections of the end effector illustrated across all considered chess sets. (left): Palm-camera image, showing the detected square; (center left): Detected chess piece. (center): An SVM training datum. Cropped image around the detected chess piece (enlarged box from center left image, but showing a different piece type). Positive (blue, center) and negative SVM training labels (red, image border) are shown. (right center): The corresponding segmented image used for visual servoing, showing the center of the blob and its two principal axes,from which the rotational correction is computed. (left): The global translational correction.

features [Dalal and Triggs 2005] are used to obtain 20 HOG square *training* templates. Using the HOG square templates turns out to be more robust than using Hough transformation, which only returns lines, but not squares. During *testing*, a standard sliding window approach is used, evaluate a score function for all positions in an image, and threshold the scores to obtain bounding boxes for the squares. The left panel in Figure 3.6 (left) shows an example of a detected square in the full image.

For robustness to scaling of squares, due to different chess boards or different hover heights of the robot arm, an image scale is chosen at *test time* to ensure that the square size in the image corresponds approximately to the size of square training templates. To find an appropriate scale, an image pyramid using ten pre-specified scales $0.8 \times (1.05)^i$, $i = 1, \ldots, 10$ is used. The square detector is applied to the image pyramid to find the best scale. This procedure is repeated 20 times to find the overall-most likely scale.

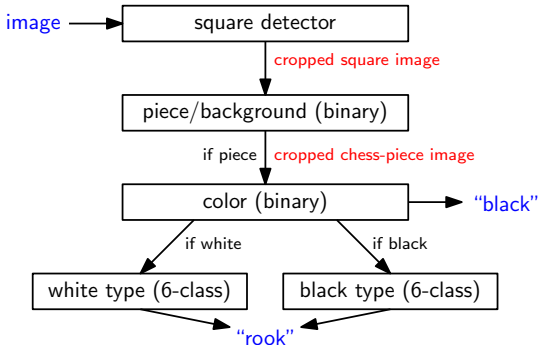



Figure 3.6: Classifier hierarchy for chess-piece recognition during game play.

*Piece/Background Detector*

For training the chess-piece detector, padded cropped images from the square detector (240×240–300×300 pixels) with chess pieces in their centers serve as positive examples (see right panel in Figure 3.6), equally sized background images serve as negative examples. On this training set, a binary linear SVM is trained using LIBLinear [Fan et al. 2008], which defines the piece/non-piece detector. Chess squares were padded (shown in yellow in Figure 3.6) to account for large chess pieces and camera tilt.

The training examples are generated automatically using the square detector described previously: The square detector typically yields up to three squares in the 640×480 image. The square closest to the center of the image is chosen: Optimally, the robot arm and the palm camera would be centered exactly at the target square). The left panel in Figure 3.6 shows the detected square closest to the image center. Finally, a rectangular image window centered at the target square is the training example for the SVM (see Figure 3.6, right panel).

*Color Detector*

The chess pieces have two colors ("black"/"white"). The chess-piece bounding box is used to train a color classifier. For example, in Figure 3.6, the blue pixels are training

data for “white” chess pieces.

*Chess Piece Classifier*

As illustrated in Figure 3.6, for each color, a chess piece classifier is trained to distinguish the piece types. The features of the respective chess-piece classifiers are concatenated SIFT [Lowe 2004] and kernel descriptors [Bo et al. 2010]. The features are extracted with $16 \times 16$ image patches over dense regular grids with spacing of 8 pixels. These local features are used to compute efficient match kernel (EMK) features [Bo and Sminchisescu 2009a] to obtain the final image level feature.

### *3.3.2 Game Play and Manipulation*

A non-simplified chess game is played from a known initial configuration of the board. Gambit is capable of making any legal chess move, including castling and piece capture. Furthermore, Gambit can identify these moves when made by an opponent.

At the beginning of the game, the system builds an initial occupancy grid, computes and saves models of piece color, and stores a table of the heights, obtained from the depth camera, of different piece types. This information is used through the rest of the game. Once the occupancy grid has been constructed, the current state of the board is tracked using depth and color information from the PrimeSense camera.

Playing is thereafter a cycle of perceiving the board state, checking for problems, deciding on and making a move, and checking again. Examples of problems that can be detected include a failure in making a move, an opponent making an illegal move, or a piece becoming occluded by another piece. In these cases, Gambit asks for help and game play is paused.

A complete move (from the end of an opponent placing a piece to the end of Gambit’s subsequent move) averages 22.5 seconds, which is well within human turn-taking time.

*User Interface*

Gambit communicates both the game state and encountered problems via a natural-language spoken interface. Every move taken during the game is repeated in spoken colloquial language. When an error is detected—either on the robot's part (e.g., dropping a piece), or on the human's part (e.g., making an illegal move)—Gambit verbally requests human intervention to make it possible to continue.

*Chess Piece Manipulation*

Depending on the move, a sequence of manipulations is required. For example, when to capture an opponent's piece, Gambit first removes the opponent's piece from the board and then moves the capturing piece to the desired location. Picking up a piece requires moving the end effector above the center of the board square containing the piece, optionally using visual servoing for local adjustments, lowering the end effector to a piece-dependent height, and closing the gripper. Depositing a piece is similar.

*Visual Servoing for Local End Effector Adjustment*

For asymmetric chess pieces or pieces not centered on a chess square, standard grasps can fail. To improve grasp success, Gambit performs visual servoing for local corrections. Visual servoing is based on images from the palm camera.

As in [Feddema and Mitchell 1989], low-dimensional image features obtained from the palm-camera image are used to correct both the 2D pose of the end effector in the hovering plane above the board and the roll angle of the manipulator. These features—the center and the orientation of the chess piece—are computed from the statistics of a binary image, which itself uses color and edge-based segmentation. Specifically, the local adjustments are computed with the loop:

1. Obtain a cropped image with a single chess piece from the square and piece detectors in Section 3.3.1 (Figure 3.6, rightmost panel).

2. To deal with changing lighting conditions, *online piece segmentation* using color and edge information is used to distinguish between the classes "chess piece" and "background" using a kernel SVM [Schölkopf and Smola 2002]. Positive training inputs for the SVM are the pixels in a $32 \times 32$ square in the center of the image, negative inputs are the same number of points along the image border. After training, the entire cropped image serves as the test input. The SVM returns a binary class label per pixel.

3. Compute the center and the orientation (longest principal axis) of the segmented chess piece.

4. Compute the global displacement and orientation of the piece in manipulator coordinates.

5. Align the end effector with the shorter principal axis and move above the piece center.

### *3.4 Experimental Evaluation*

The use of the depth camera and the palm camera enables Gambit to play a natural, uninterrupted game of chess on essentially arbitrary chess boards and uninstrumented pieces.

In 2010, Gambit participated in the AAAI Small Scale Manipulation Challenge [Anderson et al. 2010], where it played chess games against three other robots. The four participating chess robots were quite different in many respects, including basic mechanical architecture, system cost, and software architecture. In each of its three pairwise matches, Gambit scored higher than its competition, as well as attaining the highest cumulative score. The informal feedback on the subjective experience

Figure 3.7: Gambit can play using arbitrary chess sets. Shown are yellow and blue Grandmaster pieces (lower left), reproduction Lewis Chessmen pieces (lower right), and Magic Ball pieces (back row).

of playing a game with Gambit has been generally positive, with examples of people playing multiple games, games played successfully without the system developers being present, etc.

Experiments were structured in three parts. First, problems encountered with perception, manipulation, or logic during several *complete games* of chess encompassing several hundred turns were analyzed. Second, set up grasping experiments were set up to test the improvement due to visual servoing. Third, piece recognition accuracy was tested by setting up several mid-game boards and attempting to locate occupied squares and identify pieces.

### *3.4.1 Perception and Manipulation*

In order to test the baseline sensing, manipulation, and game logic, two complete games were conducted with each of the three chess sets shown in Figure 3.7. Games were played both with a human opponent, and with the robot playing both sides.

For baseline games, a standard chess opening setup with no handicaps was used. Games averaged 112.3 chess moves. For each game, the number of *interventions* re-

quested by Gambit and how many *failures* occurred (with no request for intervention) are reported.

As certain moves (castling and capturing) require multiple manipulation steps, interventions and errors are reported across all *manipulations* rather than across all moves. Six games provided 786 total manipulations. The following error types are distinguished:

- Manipulation: Failing to grasp a piece, dropping a piece, or failure to place a piece legally (see Figure 3.8).
- Perception: Failing to find a piece on the board (e.g., due to occlusion) or failing to detect illegal piece placement (see Figure 3.9).
- Miscellaneous: Gripper collision with pieces, collisions between pieces during placement, or failure to find an inverse kinematics solution for some motion (see Figure 3.9).

*Logic errors* (failure to correctly identify an opponent's move, illegal moves) are not reported because none occurred.

| Manipulations | Autonomous Successes | Interventions Requested | Failures |
|---|---|---|---|
| 786 | 720 | 38 | 28 |
| 100% | **91.6%** | 4.8% | **3.6%** |

Figure 3.8: Successes, requests for intervention, and failures of manipulation.

Generally, the results demonstrate the reliability and robustness of the Gambit system. Most errors encountered were manipulation errors (Figure 3.8), often caused by either a failure to pick up an off-center piece or illegal placement of a piece that was

| Perception | | Miscellaneous | |
|---|---|---|---|
| Interventions Requested | Failures | Interventions Requested | Failures |
| 9/786 | 12/786 | 3/786 | 7/786 |
| 1.1% | 1.6% | <0.1% | 0.9% |

Figure 3.9: Perceptual/miscellaneous failures and requests for intervention during 786 manipulations. The first row is total occurrences across all six games. The second row shows the corresponding percentages.

picked up awkwardly. Visual servoing can reduce these errors. Moreover, the Lewis Chessmen are difficult to grip because of their shape and relatively large for the chess boards used, resulting in a higher percentage of errors. Improved centering of the gripper over the piece can be expected to help with these difficulties.

### *3.4.2 Visual Servoing*

Because the total manipulation errors are still relatively low, playing the number of games necessary to demonstrate improvements from visual servoing would be significantly time-consuming due additional image-processing overhead. Thus, servoing was tested with a different experimental setup designed to concentrate on manipulation cases that are likely to fail without servoing, e.g., difficult pieces that are *deliberately* placed as badly as possible.

From each chess set, pieces were placed in the extreme corners of a board square—the worst possible placement that might still be considered legal—with other pieces on the surrounding squares to serve as visual distractors. Gambit then attempted to pick up each piece and put it down again several times, both with and without servoing. After each grasp, experimenters replaced the piece in the original position.

Regardless of how a piece was picked up, it was always put down by moving the gripper to the center of a chess square. Thus, an off-center grasp resulted in an off-center placement. Accordingly, experimenters also assigned a score describing how well-centered a piece was on the square after placement, as determined by the quality of the grasp. The five-point scale ranges from 1 (very poor grasp) to 4 (perfect grasp), with 0 assigned for illegal placement or a trial in which grasping failed. "Down-the-line" errors, where a poor but legal piece placement causes later grasping errors, are not captured by this experimental protocol. The results are summarized in Figure 3.10.

| | Num. Trials | Successful Grasps | | Grasp Quality |
|---|---|---|---|---|
| Servoing | 40 | 31 | **77.5%** | 3.1 |
| No Servoing | 40 | 7 | **17.5%** | 1.4 |

Figure 3.10: Grasping quality of deliberately poorly placed pieces with and without servoing. Whether the grasp was successful and how well-centered the piece is in the gripper is reported. Visual servoing provided a substantial improvement in success and grasp quality.

These results suggest that visual servoing can substantially reduce the number of grasping and placement failures encountered while playing games, particularly since this experiment was designed to reflect a worst-case scenario. However, found that visual servoing, particularly piece detection, somewhat slows down the game play and sensitive to the frequently changing lighting conditions, leading to an additional source of possible failures.

### *3.4.3 Piece Recognition*

Piece recognition was tested on the Lewis Chessmen, which are less stylized and lower contrast than the other chess sets. Supervised training data was collected from games,

and identification models were trained according to Section 3.3.1. Two chess games in different states of completion were used to test how accurately the system could recognize the pieces in use.

First, the depth camera was used to determine which chess squares were occupied. Second, the robot hand moved above occupied squares to take pictures of each piece from directly above. Third, these images were used as test images of the piece recognition procedure detailed in Section 3.3.1. The confusion matrix is shown in Figure 3.11. On a test set of 59 images, four piece types were misclassified, giving an overall accuracy of 93.22%.

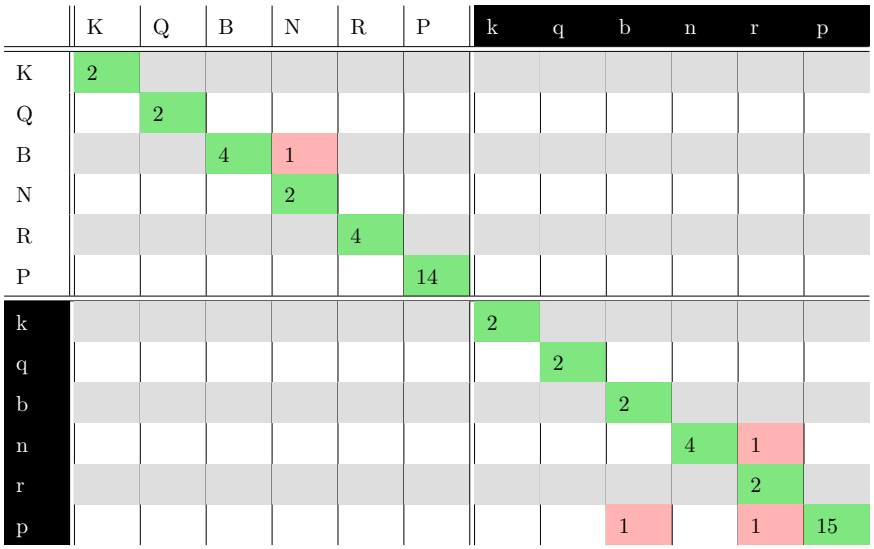



Figure 3.11: Confusion matrix from recognition of 59 pieces participating in two chess games. Pieces are labeled using standard FEN notation, e.g., "N" is a white knight. The *color* of the chess piece (white/black) was 100% accurately determined, the recognition accuracy of the piece *types* was about 93%.

Although the data set is small, the overall results suggest that close-range object recognition using the camera in the gripper has real promise in manipulation and interaction tasks.

### *3.5 Discussion*

In the chess domain, reliable piece recognition can provide a principled way of resolving errors, such as perception uncertainties, e.g., occlusions: The gripper camera could simply be sent to look at areas of the board where problems were encountered. This demonstrates the more general idea that a reasonably high-resolution, close-up image of the workspace can lead to improved grasp selection, better motion planning, and ambiguity resolution.

Comprehensive perception of the human, e.g., monitoring the human's facial and hand gestures, would enable improved human-robot cooperation, which could allow for more natural interactions, such as smoothly interleaving other activities with game play. Game play, with its easily varied degree of structure, represents an excellent toy domain for exploring the usefulness of a specific platform for human-robot interaction.

This work can generalized toward the ultimate goal of an intelligent manipulation assistant, and the robotic platform described here is appropriately capable for the wide variety of tasks that may be useful for language grounding. Subsequent chapters do not describe robotic platforms in any detail, but this chapter provides a firm grounding in the realities of physical perception and manipulation, which in turn drive many of the fundamentally probabilistic tasks addressed in the remainder of this dissertation.

# Chapter 4

# LEARNING TO FOLLOW DIRECTIONS USING MACHINE TRANSLATION

One of the key requirements for robots deployed in human-centric spaces is the ability to follow directions given in unconstrained natural language. Such instructions may contain novel language, and may pertain to actions and referents which are novel to a specific environment or task; as a result, such a robot will have to learn to follow directions at least partially on the fly, rather than pre-programmed before such robots are deployed. In this chapter and the next, two approaches are presented to learning how to follow instructions, using simulated mobile robots following route directions as a platform.

This chapter investigates how statistical machine translation techniques can be used to bridge the gap between natural language route instructions and a map of an environment built by a robot. The approach involves the use of training data to learn to translate from natural language instructions to a formal description of a path through an automatically labeled map. The complexity of the translation process is controlled by taking advantage of physical constraints imposed by the map. As a result, this technique can efficiently handle uncertainty in both map labeling and parsing.

---

The work described in this chapter was originally presented at HRI 2010 [Matuszek et al. 2010], and was performed in collaboration with Dieter Fox and Karl Koscher.

## *4.1 Motivation and Overview*

The problem of following instructions can be described as finding a way of going from the natural language description, or layer, to an underlying *map layer* that is grounded in a map built by a robot from sensor data. Conceptually, this is accomplished by applying statistical machine translation (SMT) tools to the task of translating from natural language into a formal intermediate *path description language*, designed to closely mimic what current robotic sensors and actuators can handle in a real-world environment. This reduces intermediate translation steps and dependence on heuristic handling of input, while minimizing assumptions about available information.

There exists a substantial body of work on robotic navigation, and specifically on direction-following in navigation and other tasks, which has been described in more detail in Section 2.3.1. This work contributes to that in the following ways.

First, it represents a demonstration of a system that takes full advantage of existing robot mapping and place-identification technology, by showing pathfinding in an environment where available information is limited to automatically obtained map segmentation and labels—neither human labeling nor manual landmark identification are required.

Second, uncertainty is taken into account in both mapping and the language parsing by combining the two into a single analysis of the most likely interpretation of a set of directions. Finally, map information is heuristically incorporated into the parsing process, so as to manage the combinatorics of a very non-restrictive formal grammar with an extraordinarily high branching factor.

To achieve the goal of limiting an agent to environments which can actually be detected by current mapping technology, existing topological place-labeling techniques [Friedman et al. 2007] are taken as a basis for mapping. This enables robots

to autonomously build maps that can be described in terms of *typed* areas (such as hallways, rooms, and intersections). A formal language is defined that describes movement through a sequence of such areas, and a Statistical Machine Translation (SMT) system is used to learn a probabilistic translation model between natural language and the formal description of paths.

The SMT system is trained on examples of natural language route directions and the corresponding paths through a map built by a mobile robot. Natural language directions are very incomplete with respect to areas being traversed, generally specifying only particular decision points and salient elements. As a result, a fully specified path through a map is typically only very ambiguously described by any given set of natural language instructions, yielding a large set of possible parses. In this work, the space of possible movements through the map is used to constrain the parsing process to physically relevant parses, rather than the infeasibly large set of possible parses. Map uncertainty and parse probabilities are combined to find a path that corresponds to the directions.

The next section provides background on statistical machine translation and a description of the technical approach taken.

## *4.2 Approach: Language, Semantic Mapping, and Search*

The approach may be summarized as using machine translation techniques to learn to transform natural language instructions into a sequence of instructions that can be executed by a robot equipped only with a map and a laser range-finder. Specifically, there are three representations of a route: statements in NL (the natural language route instructions), statements in a formal path description language (described in Section 4.2.1), and an actual path through a map at the map layer.

SMT learning trains a parser that transforms NL instructions into the path descrip-

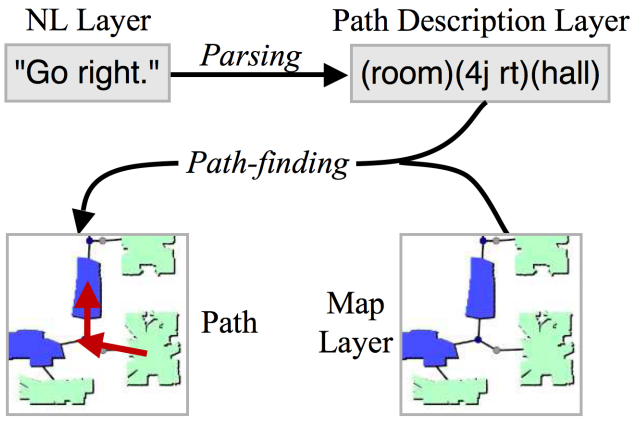



Figure 4.1: The stages of converting a statetement in natural language to a path through a map. Parsing is performed by a parser trained on example route instructions and map traces. A route is represented in three ways: (top left) in NL instructions; (top right) in a map-based path description language; (bottom left) the actual route traversed, shown in red.

tion layer (parsing), which can then be transformed into the map layer (path-finding). Both parsing and path-finding must handle uncertain information. The natural language instructions are constrained by the maps provided, which do not contain landmarks, and are further limited by the grammar's inability to express *context* (information about the nodes surrounding those that are actually traversed).

### *4.2.1 Labeled Maps and Formal Language*

*Labeled Topology Using Voronoi Random Fields*

The maps being used for training and testing are constructed from laser range-finder data, and have been automatically segmented and labeled using *Voronoi Random Fields* [Friedman et al. 2007]. In this approach, a Voronoi graph is initially extracted

from an occupancy grid generated with a laser range-finder; each point on the graph is represented as a node of a conditional random field, which is a discriminatively trained graphical model. The resulting VRF estimates the label of each node, integrating features from both the map and the Voronoi topology. The labels provide a segmentation of an environment, with the different segments corresponding to rooms, hallways, intersections, or doorways (as in Figure 4.2 (far left)).

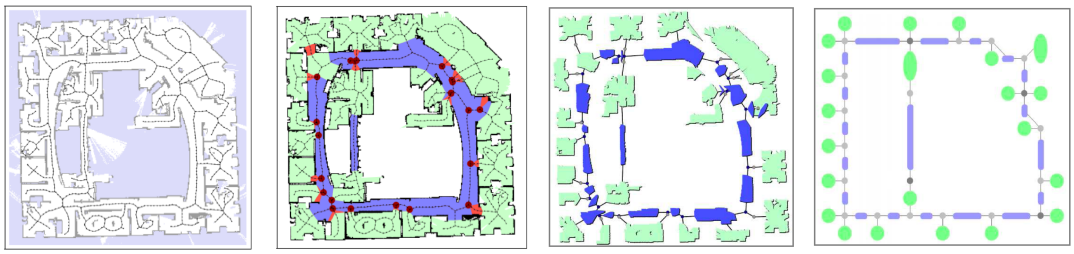


Figure 4.2: The test map used for evaluation. (far left) An occupancy map constructed from laser range-finder data, with a Voronoi-based route graph shown. (left) The labeled Voronoi graph defines a place type for each point on the map. Hallways are colored red, rooms are green, and junctions are indicated by blue circles. (right) A topological-metric map given by the segmentation of the labeled Voronoi graph. (far right) The topological metric map transformed into the typed-node representation, with junctions representing edges.

These maps can then be segmented to provide topological-metric maps of the labeled Voronoi graph. The spatial layout of rooms and hallways is retained, along with connectivity structure based on identification of intersections. This structure can be transformed to an idealized map of typed nodes with connecting edges; the stochastic classification is not perfect, but has an accuracy above 90% [Friedman et al. 2007].

*Formal Path Description Language*

The synchronous context-free grammar (SCFG) parsing model used (see Section 4.3)

requires that the target language (the path description layer) be defined by an unambiguous grammar, which is given in Figure 4.3. Additionally, path-finding—going from the path description layer to the map layer—must be robust against uncertainty in the segmenting and labeling of the map.

```
n:Statement → ({ ( go n:Action ) })
n:Action → ({ n:Vertex n:Movement })
n:Movement → ({ n:Edge n:Vertex })
n:Movement → ({ n:Edge n:Vertex n:Movement })
n:Vertex → ({ ( room ) })
n:Vertex → ({ ( hall ) })
n:Edge → ({ ( n:Junction ) })
n:Junction → ({ n:Jtype n:Jdir })
n:Junction → ({ n:Jtype n:Jorient n:Jdir })
n:Jtype → ({ j4 })
n:Jtype → ({ j3 })
n:Jorient → ({ t })
n:Jorient → ({ r })
n:Jorient → ({ l })
n:Jdir → ({ rt })
n:Jdir → ({ st })
n:Jdir → ({ lt })
```

Figure 4.3: The grammar of the path description language into which natural language directions are parsed. Hallways and rooms are treated as nodes of a map, connected by intersections, which are three-way or four-way junctions (`j3` and `j4`); the direction in which a junction is traversed is given as right, straight, or left (`rt`, `st`, and `lt`); and the orientation of three-way junctions is given (`t`, `r`, and `l` indicate which direction is traversible). The key "`n:`" is used to mark nonterminals.

The path description language defined by the grammar is quite simple: a path is defined as a series of nodes through the topological graph-map, connected by edges.

Junction nodes are parameterized with available exits and information about the orientation of the robot as it traverses the junction, which captures rotation. Nodes are described relative to the agent's point of view, rather than in absolute terms; for example, a 3-way junction may be a T-junction, meaning there are openings to the robot's right and left, but could not be described as having openings to the east and west.

The grammar contains both *terminals*—the lexicon of words in the path description language—and *nonterminals*, which can be decomposed into terminals or other non-terminals. Nonterminals in Figure 4.3 are explicitly marked with `n:`, such as n:Action, whereas terminals (such as `rt` or `room`) are not marked. While the names of the terminals are chosen to be human-readable, it should be noted that there is no semantic information encoded in the system at the beginning. The connection between the word "right" and the terminal `rt` is entirely learned.

This statistical approach allows this system to handle noise (such as previously unseen or irrelevant words), as well as avoiding any explicit definition of structure such as labeling parts of speech or pre-defining the words that correlate to actions such as turning.

As mentioned, there are two noteworthy simplifications to the language: there is no representation of *landmarks* (other than the nodes themselves), nor of *context*. Landmarks are any element of the environment that could be used to describe an area other than node type. Examples include sentences such as "go past the couch" or "the room with the blue rug."

In this use, context describes nodes on the map that surround the path, but are not actually traversed. Examples of contextual path descriptions might include sentences such as "Go to the end of the hall," which requires information about the step one past the steps actually taken, or "go past a room." In these cases, the spaces being

described are not on the path taken by the agent, and so are not represented in the path descriptions, making it impossible to learn a translation into the path description language.

### *4.3 Statistical Machine Translation*

The underlying approach used is Statistical Machine Translation, in which a large body of example sentence pairs are used to learn a translation system automatically [Lopez 2008]. A modified version of the Word Alignment-based Semantic Parser (WASP) developed by Wong & Mooney [Wong and Mooney 2006][1], which learns a parser that translates from novel strings in a source language to a formally defined target language, is used.

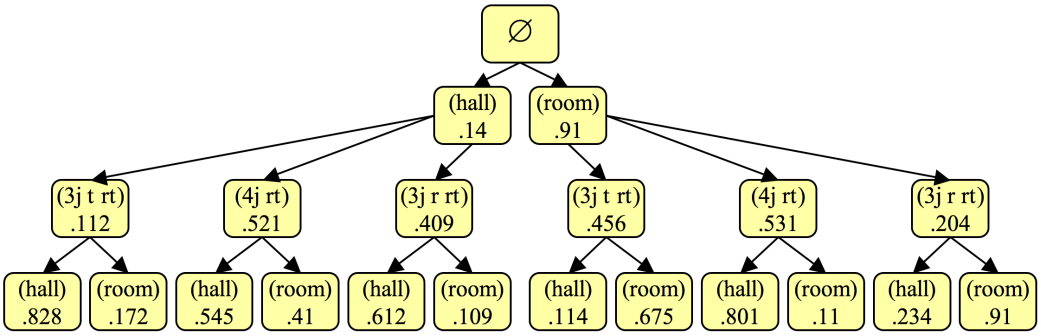



Figure 4.4: The complete set of possible correct parses of the command "go right," with weights provided by the agent's beliefs about the map. Because instruction-giving is often noisy, incorrect parses should also be considered with low weight, making the fan-out of the actual tree larger.

WASP implements a synchronous translation learning model, meaning that parse trees are constructed simultaneously for sentences in the source and target language

[1]Retrieved from `http://www.cs.utexas.edu/~ml/wasp`.

by applying a sequence of production rules to higher-level *nonterminals*. For example, in translating between two natural languages, the nonterminals `NP` and `V` might refer to noun phrases and verbs; a production would describe how `NP` could be synchronously replaced by the strings "I" in English and "*je*" in French.[2]

The learning process can be described at a high level as follows. First, a statistical word alignment model [Brown et al. 1990] is learned. In word alignment, pairs of corresponding strings between source and target languages are discovered, for example, *je*/I or "turn right"/`rt`. For this task the off-the-shelf word-alignment tool GIZA++ [Och and Ney 2003] is used. A lexicon of production rules is then generated by creating a rule for each aligned string pair in *each* training pair; for example, the paired training sentences "I will" and "*je vais*" provide evidence for the existence of the production rules "*je*←`NP`→I" and "*vais*←`V`→will."

If sufficient training data is present, the set of production rules learned will implicitly define the set of all possible derivations of target strings. Finally, a set of parameters is learned that define a probability distribution over derivations (a sequence of rule applications). This is a probabilistic extension of the synchronous context-free grammar (SCFG) parsing framework [Aho and Ullman 1972].

A probabilistic SCFG $\mathcal{G}$ is then defined as follows:

$$\mathcal{G} = < \mathcal{N}, \mathcal{T}_{in}, \mathcal{T}_{out}, \mathcal{L}, \mathcal{S}, \lambda >$$

In which $\mathcal{N}$ is a finite set of nonterminals, $\mathcal{T}_{in}$ and $\mathcal{T}_{out}$ are finite sets of terminals (words) in the source and target languages, $\mathcal{L}$ is a lexicon of production rules, $\mathcal{S}$ is a defined start symbol, and $\lambda$ is the set of parameters defining a probability distribution over derivations.

[2] [Chiang 2006] is recommended to the reader seeking a clear overview of SCFGs; the terminology of [Wong 2007] is followed in describing grammars, rules, and derivations.

WASP uses a log-linear probabilistic model described by a set of parameters $\lambda$. The probability of a *derivation* (sequence of translation steps) $\mathbf{d}$ given an input sentence $\mathbf{e}$ is:

$$\Pr_\lambda(\mathbf{d}|\mathbf{e}) = \frac{1}{Z_\lambda(\mathbf{e})}\exp\sum_i \lambda_i f_i(\mathbf{d})$$

Where $f_i$ is one of three types of features:

- For a rule, the number of times that rule is used in any derivation.
- For a word, the number of times that word is generated from a word gap in a derivation.
- The total number of words generated from word gaps.

However, only the results of derivations—that is, input/output sentence pairs $(\mathbf{f_j}, \mathbf{e_j})$—are provided as training data. The conditional probability of an output sentence being produced by an input sentence can be computed by summing over all the derivations that perform such a translation:

$$\Pr_\lambda(\mathbf{f_j}|\mathbf{e_j}) = \sum_{\mathbf{d}} \Pr_\lambda(\mathbf{d}|\mathbf{e})$$

WASP uses a variant of the Inside-Outside algorithm [Baker 1979] to avoid enumerating all possible derivations, since the number of derivations may be intractably large. An estimate of $\lambda$ can then be obtained by maximizing the conditional log-likelihood of the training set, i.e. maximizing the sum of the conditional log likelihoods of each sentence pair, using gradient-descent techniques (L-BFGS in WASP's case). A detailed description of this approach can be found in [Wong 2007].

The advantages of using this approach to parser learning are twofold. First, rules with nested non-terminals can readily represent commands that do not specify a fixed

number of actions, such as "take the second left," which may traverse an arbitrary number of map nodes. Second, it minimizes commitment to a specific domain. This approach can be applied to any domain for which a simple grammar can be defined and examples can be readily expressed using that grammar.

However, because the process of parsing from route instructions to path descriptions is not dependent on a specific map, and because the formal language being parsed into does not have complex or deeply nested structural constraints, the number of correct parses is very large compared to parsing problems which target more constrained formal languages such as [Wong 2007; Chen and Mooney 2008; Dzifcak et al. 2009] (see Figure 4.5 for an example). Management of this complexity is described in the next section.

## *4.4 Controlling Search Complexity*

The parsing process produces uncertain information; additionally, because the map is automatically labeled, there is uncertainty in the annotation of the underlying map graph. (For instance, the robot might not be sure if a certain node in the topological graph is a hallway or a room.) This combined uncertainty means that evaluation of parses against the map is not necessarily reliable.

In order to to be robust in the face of such uncertainty, the direction-following agent backtracks to lower-probability paths if the most probable path does not lead to the desired destination. As a result, all parses of a sequence of route instructions must be retained and considered when choosing the most probable path to take, leading to an exponential blowup in possible path traces for a given input description. The large fan-out of possible parses makes a brute-force approach to parsing intractable (see Figure 4.6 for example parses of a language utterance).

Complexity in the parsing stage and in the path-finding stage are addressed separately.

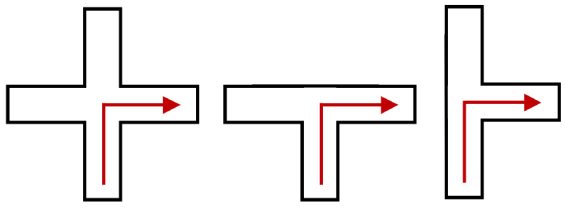


Figure 4.5: An example of parsing ambiguity. Without reference to a specific map, all of the following are correct parses of the phrase "turn right," with the distinct path descriptions `(j4 rt)`, `(j3 t rt)`, and `(j3 r rt)`. Combinations of such commands (such as "turn right, then turn right again") quickly become intractable.

*Parse Complexity*

Route instructions may be thought of as a sequence of semi-independent segments, each of which describes a separable set of steps between decision points. Because the parse grammar does not have much nested structure, it is rare for the parse of a segment to rely on other segments; as a result, segments of the natural language inputs can generally be parsed separately without harming correctness. For example, given the instructions "Go left, then turn right," the meaning of "turn right" is unlikely to depend on the meaning of "Go left"—unlike general natural language, in which a later phrase may refer back to the subject of a previous phrase.

Language segmentation can be done either automatically (e.g., by relying on phrase boundaries learned by the parser [Chen and Mooney 2008]), or by pre-processing route instructions in some way; in the experiments, described here, instructions are segmented by splitting on defined keywords.

In this terminology, the natural language route instructions "Go out of the room and go right, then go past two intersections, then turn left" contains four segments. The

| | |
|---|---|
| (room) (4j rt) (room) | (room) (4j rt) (hall) |
| (room) (3j r rt) (room) | (room) (3j r rt) (hall) |
| (room) (3j t rt) (room) | (room) (3j t rt) (hall) |
| (hall) (4j rt) (room) | (hall) (4j rt) (hall) |
| (hall) (3j r rt) (room) | (hall) (3j r rt) (hall) |
| (hall) (3j t rt) (room) | (hall) (3j t rt) (hall) |

Figure 4.6: All possible ways to describe an immediate right turn, as specified by the grammar of the path description language.

correct path description when these instructions are applied to Figure 4.2 is `(room) (j4 rt) (hall) (j3 r st) (hall) (j3 l st) (hall) (j3 t lt) (room)`, corresponding to a path going from the entryway down to the room on the lower right. However, without oracular knowledge of the correct parse and correct map labeling, this is only one many thousands of path descriptions that must be considered.

*Pathfinding Complexity*

While segmented parsing improves the complexity of the parsing step, every possible combination of segment parses must still be considered during path-finding. As well, each step has an associated uncertainty, produced by combining the parse weight with the current belief as to whether the step is correct with respect to the map.

An example helps clarify this description. A single parse of the phrase "go right" may be `(room) (j4 rt) (hall),` with an associated parse weight of 0.5. If this command is given while the robot is at node 418 on Figure 4.7 (b), the first step, `(room),` is believed to be consistent with the map, and so should be scored highly. The second step, however, describes a 4-way intersection, when it is in fact believed to be a 3-way intersection; that parse receives a lower score. These scores, when combined with

the parse score, must be treated individually. Sequences of nodes can be stored in a weighted tree (Figure 4.4 shows such a tree for the instruction "go right"); however, the size of that tree rapidly becomes intractable for longer sequences of commands.

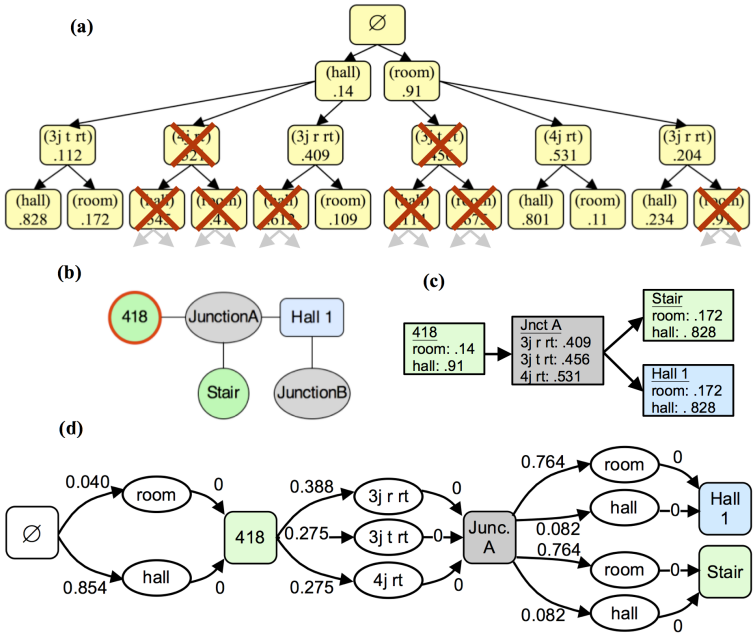



Figure 4.7: Developing a reduced navigation tree. A large, unbalanced navigation tree is shown in (a); the weight of *each node* is obtained by combining the parse weight with the current belief as to whether the node-type is correct with respect to the map. Nodes at depth $d \leq 5$ are not shown (edges downward are indicated). Given a very small sample map (b), the tree can be reduced to only those paths of length 4 or less (others in (a) are marked ×). (c) Nodes at each depth are combined into labeled nodes, which are associated with the possible map nodes to which they may correspond; the maximum weight from all nodes with the same label is retained. (d) A map/path tree derived from (c). Edge weights are the negated log of the original label weight; ordering of additive path cost is then identical the ordering of paths by optimality.

In order to search this space, a combined map/parse tree is introduced, in which the structure of the map is used to both constrain and efficiently store possible parses (see Figure 4.7).[3] A tree of nodes corresponding to regions of the map is constructed, rooted in the robot's starting position and disallowing cycles by terminating a branch when the only possible step is a node that already exists on that branch. Because there are a sharply limited number of map nodes that are $n$ steps from a known starting point, this tree is feasible to construct. The $n$-th nodes of possible parses are then stored as a list attached to each tree node (see Figure 4.7 (c)).

This simplification requires that the robot possess a complete map, as well as the ability to follow the path through the map and a known starting point. (A known destination is not required.) Experimentally, a test is also supplied for whether the destination has been reached once a path has been traversed. The goal of the combined system is then to produce an ordered list of paths through the map, representing a sequence of possible interpretations of the directions given. If the first proposed path does not reach the desired destination, the next highest-probability path is selected and the robot backtracks to the point where the next-most-likely path diverges from the path already traveled.

If a path fails, it is often not obvious at what step the problem occurred, meaning it is not obvious how to prune the search tree further when errors are encountered. The need to try several paths (due to map or parsing errors, or due to human errors in giving the directions) therefore means finding the first, second, ... $k$-th highest-probability walk through list entries on the map/parse tree. This task can be considered as a variant of the well-studied problem of finding the $k$-th best (lowest cost) path through a graph. The individual steps are treated as edges between nodes, with a cost proportional to their probability; Figure 4.7 (d) shows this new graph.

---

[3]This representation is tractable, but is not guaranteed to be optimal; see supplemental materials to [Matuszek et al. 2010] for a more detailed discussion: `http://tiny.cc/HRINavTreeSupplemental`

The canonical solution to this task is Yen's $K$th-shortest-path algorithm [Yen 1971], which is $O(kn(m + n \log n))$ for a graph with $n$ nodes and $m$ edges; [Martins et al. 2001] provides a more efficient variant which is used for this work, with some modifications. First, because probabilities are combined via multiplication and path costs are combined by addition, the log of the probabilities is used, negated in order to make higher probabilities consistent with lower costs. Second, the $K$th-shortest-path algorithm does not handle multiple edges between nodes, so dummy nodes (not shown in Figure 4.7) are inserted between each pair of map/parse tree nodes. Each dummy node contains a pointer to the actual step value (for example, `room`), which makes reconstructing a path description from a graph walk trivial.

Because the desired endpoint is not known, it is necessary to find the $k$ best paths from the start node to all other nodes in the graph, making the actual complexity $O(kn2(m + n \log n))$ in the worst case. When this graph has been constructed, the search process proceeds by creating a list of the best ($k$=1) path to every node, ordering the resulting $n - 1$ paths by total probability, and exploring the first. If the first fails, $k$ is incremented, $n - 1$ more paths are generated (the second-best path to every node except the starting point), and the process is repeated.

## *4.5 Experimental Evaluation*

As described in Section 4.4, the simulated agent starts with a complete (but possibly incorrectly labeled) map, the ability to follow a path through the map, a starting point, and a test to determine whether the goal has been reached. The evaluation is based on whether the robot reaches the desired destination, and if so, whether the path taken is the one described by the natural language instructions—since both testing and training maps contain loops, a destination can always be reached in multiple ways, but the primary question is how successful the robot is in following directions.

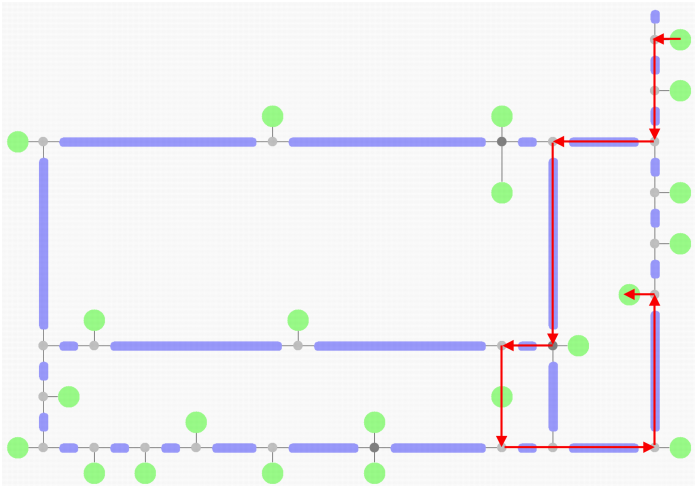


Figure 4.8: The training map, produced from the Allen Center computer science building. The red line shows one of the paths the route instructors were asked to describe in the training data.

### *4.5.1 Data Collection and Corpus*

Training and testing maps were generated from Voronoi graph maps of two buildings, the Allen Center and Intel Research (pictured in Figure 4.8 and Figure 4.2, respectively.) These maps were constructed from non-simulated laser range-finder data collected by a SICK scanner mounted on a small Pioneer robot base.

Paths can be generated randomly, and because the path description language is unambiguous with respect to movement through a map, it is feasible to generate the formally described component of training examples. The process of collecting training data is primarily constrained by obtaining natural human descriptions of paths.

To obtain training data, five random paths were generated on the training map. Paths were not always the shortest possible route between start and end points. For this work, we obtained a collection of 33 sets of route instructions describing five

paths from eight volunteers. The data collected varied in length from 4-9 instruction segments and 15-20 nodes traversed.

In order to increase the size of the training set, the training data was analyzed and phrases corresponding to individual actions (such as "turn left") were extracted and randomly combined to describe other paths through the map, producing a much larger synthetic data set, into which words were randomly introduced to provide noise. (This allows for the learning of word gap models, although those models would in this case be random.) Some examples of (noiseless) route instructions produced in this way:

```
Leave the room and go left, go past the next junction, and then
take the second right.
```

```
Go straight through the second junction and take a right, pass
the next junction, and go into the room ahead.
```

As noted in Section 4.2.1, neither landmarks nor context are considered in this work. The former do not appear in the natural-language training data collected. Because volunteers were shown only the final topological graph-map when giving instructions, no additional landmarks were described. However, context-based instructions, which are not representable in the current path description language, were used in some route instructions.

The testing corpus consisted of an additional five paths provided by four volunteers over the Intel building testing map, providing 20 sets of route instructions. Additional human volunteers evaluated the test instructions by attempting to interpret them as paths through the test map. 70% of test routes were followed successfully by human evaluators. This number, while low, is consistent with results reported by others [Macmahon et al. 2006; Wei et al. 2009], a testament to the fundamental difficulty of the problem. There were no cases in which the complete system successfully followed instructions that human evaluators could not follow.

### *4.5.2 Experiments*

Of the fourteen human-followable direction sets, the system was able to follow ten successfully, or 71% (50% of the total), getting to the correct destination along the specific route described by the route director. For a perfectly labeled map, the route selected to explore first was the correct one in all cases except those that involved counting (e.g., "Go through two intersections,"), which—because the path description language cannot represent such conditions—requires the agent to arbitrarily try different numbers of steps before the correct path is found.

For testing purposes, the experiment was re-run with the introduction of a random map-labeling error of 90% (higher than that produced by the actual Voronoi Random Field map labeling). Given an erroneous map, the robot frequently made errors specific to that map; for example, if a four-way intersection is incorrectly identified as being a three-way intersection, then a robot instructed to "take the left" might pass through that intersection rather than turning, preferring the parse that matched the incorrect map labels. Because the robot backtracks when its explorations fail, this error is recoverable; the correct parse will be explored and the goal will be reached, although not as efficiently.

The system successfully generalizes from a few examples to unfamiliar cases, making it able to handle not only input with terms that were not previously encountered, but also those requiring semantic grouping, e.g., the similar uses of the terms "right" and "left." Functionally, such higher-level semantic generalizations can be learned from incomplete training data, based on the fact that the two terms appear in otherwise identical examples some of the time. Such concepts are learned automatically from examples; no information on the *meaning* of left, as a motion, was explicitly encoded in the map or in the path description.

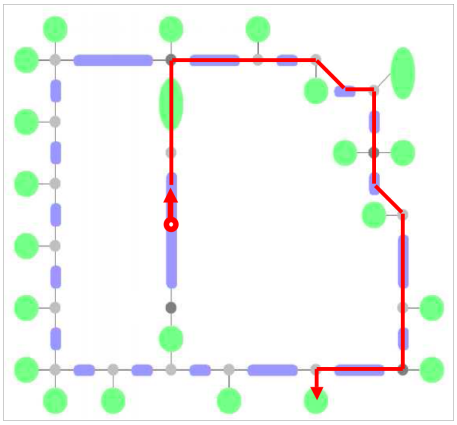



Figure 4.9: One of the routes used for testing, with human-provided directions that could be followed successfully. This set of navigation instructions demonstrates the system's learned capability to follow directions which include counting information (e.g., "take the third right").

### *4.5.3 Discussion*

This system described in this chapter allows simulated agents to learn, entirely from examples, how to use statistical machine translation to follow natural language directions tractably. The search mechanism feasibly combines management of uncertainty in parsing and in map labeling; the approach is robust enough to demonstrate successful navigation of automatically labeled maps based on natural language route instructions with an approximately 70% success rate—comparable to other work performed at the same time, without making use of human-provided heuristics.

As erroneous map labeling increases, so does the likelihood of exploring an incorrect path based on poor information. This result is unsurprising, but as robotic mapping of indoor environments using low-cost sensors improves (e.g., [Henry et al. 2012, 2013]), failures of this sort will become less relevant to overall performance. It is

noteworthy that the robot's behavior in this case—exploring the "best" (i.e., most likely) interpretation and backtracking if the goal is not reached—mimics human direction following.

Given the nature of statistical machine translation, it is very likely that these results would improve with additional training data and less reliance on synthetic data. However, the system described in this chapter still requires access to a map—the world model is provided in advance of the actual grounding of language into actions. The next chapter describes a system that uses a more semantically accurate grammar of language. This allows for the representation of more complex control structures and, more importantly, for the simultaneous discovery of a world model.

# Chapter 5

# LEARNING TO FOLLOW DIRECTIONS USING SEMANTIC PARSING

Human route instructions include complex language constructs which robots must be able to execute in a variety of settings. One requirement for reaching that goal is to induce a semantic parser that produces correct, robot-executable commands for such instructions without being given a fully specified world model such as a map. Here, language grounding is treated as as a problem of parsing from a natural language to a formal control language capable of representing a robot's operation in an environment at a higher level.

In the previous chapter, the problem of learning to follow natural language directions through a map was addressed. Route directions were compared to a map in order to produce a formal description of a robot's intended path, expressed in a path direction language. While results were comparable to the state of the art, this approach also has weaknesses that made it potentially difficult to scale to other problem domains; these are addressed in this chapter.

The work described here again focuses on learning to follow directions, using indoor navigation as a testbed. The approach taken is one of semantic parsing: taking natural language inputs paired with interpretations drawn from the physical world, and learning to interpret novel instructions into a formal control language for a physical

---

The work described in this chapter was originally presented at ISER 2012 [Matuszek et al. 2012b], and was performed in collaboration with Evan Herbst, Luke Zettlemoyer, and Dieter Fox.

agent. The key difference lies in targeting a formal representation that captures the semantic intent of English statements, rather than the final physical interpretation.

## *5.1 Motivation and Overview*

Tying the interpretation of route instructions to a map, as in the previous chapter, has two key disadvantages. The first is that instructions can only be parsed in the context of a map. Whether the map is known in advance or discovered simultaneously with the parsing process, this makes the ultimate formal interpretation of instructions relevant only to a specific world model. Second, as a result of the focus on a grammar that describes a path, at no point are the instructions transformed into a formalism that captures the intermediate semantic content of the language. This lack of abstraction, while useful for controlling complexity, makes it difficult to transfer the learning approach to other domains or to reason about intent, context, and correctness.

This chapter describes training a parser to transform natural language commands to semantically correct actions and control structures, which can then be readily implemented in a robot execution system. Learning is based on example pairs of English commands and corresponding control language expressions. Specifically, a parsing model is trained that defines, for any natural language sentence, a distribution over possible robot control sequences in a LISP-like control language called Robot Control Language, or RCL.

Evaluation demonstrates that the resulting system can learn to translate English commands into sequences of desired actions, while correctly capturing the semantic intent of statements involving complex control structures. The procedural nature of the formal representation allows a robot to interpret route instructions online while moving through an unfamiliar environment. The world model—in this case, an indoor map—is not required for language parsing.

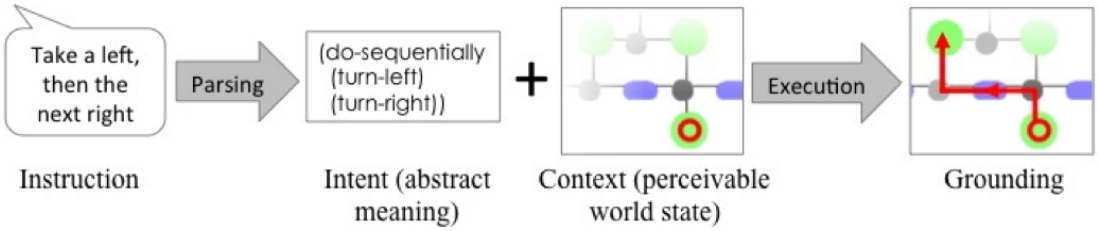



Figure 5.1: The task: Going from NL to Robot Control Language. First, the natural language command is parsed into a formal, procedural description representing the intent of the person. The robot control commands are then used by the executor, along with the local state of the world, to control the robot, thereby grounding the NL commands into actions while exploring the environment.

Grounding relations are learned from data (rather than predefining a mapping of natural language to actions) and executed against a previously unseen world model (in this case, a map of an environment), as illustrated in Figure 5.1. Training is performed on English commands annotated with the corresponding robot commands. This parser can then be used to transform new route instructions to execution system inputs in an unfamiliar map. The resulting system can represent control structures and higher-order concepts. The approach is tested using a simulator executing the commands produced.

## *5.2 Approach: Robot Control Language and Parser Induction*

Much of the work on learned grounded language assumes previous knowledge of the world in which a robot is operating. In contrast, the approach taken in this work is to parse natural language into a high level *specification* of robot behavior, which can then be executed in the *context* of an arbitrary environment; the meaning of terms in the formal grammar is still learned entirely from grounded examples, rather than pre-

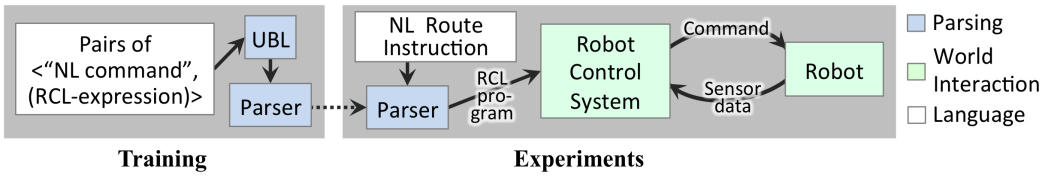



Figure 5.2: The high-level architecture of the end-to-end system. Training is performed by learning a parser from English instructions to RCL. In the experimental phase, the learned parser maps NL instructions to an RCL program that is executed by the robot.

specified. Differentiating context and meaning in this way lends itself well to tasks in robotics, where full knowledge of the environment is rarely available. This approach also separates the parsing problem from execution, providing a more appropriate layer of abstraction for robot control. This architecture is shown in Figure 5.2.

### *5.2.1 RCL*

Control structures are expressed in in a logic-based Robot Control Language (RCL), inspired by task execution systems such as PRS [Ingrand et al. 1996], RPL [Beetz et al. 2001], and GOLEX [Hähnel et al. 1998]. For a given set of route instructions, RCL represents the high-level execution semantics captured in the original NL route instructions. Figure 5.3 illustrates RCL programs for paths through an automatically generated, semantically labeled map [Friedman et al. 2007], along with the instructions corresponding to that path. (RCL is a subset of the typed lambda calculus, and can be represented in a LISP-like format.)

The language is intended to be high-level enough that different execution systems can be used for various tasks, rather than specifying low-level actuation and perception tasks. As such, RCL expressions can encode actions with nontrivial execution subtasks, which may combine perception and action. For example, `(exists left-loc)`

| locations | |
|---|---|
| `current-loc:loc` | robot's current position |
| `forward-loc:loc` | location ahead of robot |
| `left-loc:loc` | to robot's left |
| `right-loc:loc` | to robot's right |
| `exists:t [loc]` | does `[loc]` exist? |
| **movement** | |
| `move-to:t [loc]` | move to `[loc]` |
| `turn-left:t` | take next available left |
| `turn-right:t` | take next available right |
| **logic** | |
| `and:t [t] [t]` | boolean 'and' |
| `or:t [t] [t]` | boolean 'or' |
| `not:t [t]` | boolean 'not' |
| **loops** | |
| `do-until:t [t] [e]` | do `[e]` until `[t]` is true |
| `do-n-times:t [n] [e]` | do `[e]` `[n]` times |
| **querying the type of a node** | |
| `room:t [loc]` | is `[loc]` a room? |
| `junction:t [loc]` | is `[loc]` a junction? |
| `junction3:t [loc]` | is `[loc]` a 3-way junction? |
| `junction4:t [loc]` | is `[loc]` a 4-way junction? |
| `hall:t [loc]` | is `[loc]` a hallway? |
| **mid-level perception** | |
| `turn-uniq-corner:t` | take available turn |
| `take-uniq-exit:t` | leave a room with one exit |
| **other** | |
| `<#>:n` | integers |
| `do-sequentially:t e+` | do each thing in turn |
| `verify:t t` | error if arg is false |

**"Go left to the end of the hall."**

```
(do-sequentially
  (turn-left
  (do-until
    (or
      (not
        (exists forward-loc))
      (room forward-loc))
    (move-to forward-loc)))
```

**"Go to the third junction and take a right."**

```
(do-sequentially
  (do-n-times 3
    (do-sequentially
      (move-to forward-loc
      (do-until
        (junction current-loc
        (move-to
          forward-loc))))
  (turn-right))
```

**"Go straight down the hallway past a bunch of rooms until you reach an intersection with a hallway on your left."**

```
(do-sequentially
  (do-until
    (and
      (exists left-loc)
      (hall left-loc))
    (move-to forward-loc))
  (turn-left))
```

Figure 5.3: (left) Terms in Robot Control Language; hallways, rooms and intersections are treated as nodes of a map. The return type of each term is given after its name, followed by parameter types: `e` (entity), `t` (boolean), `n` (number), `loc` (map location). (right) Examples of sentences from the test corpus and their RCL interpretations.

is a high-level command to the execution system to determine whether there is an opening to the left, while `(move-to forward-loc)` tells the robot to move one node forward in the semantic map. This also allows the language to be quite compact; a full list of RCL commands is given in Figure 5.3(left), including terms for logical expressions, queries about local state, and actions to be performed.

### *5.2.2 Parsing*

RCL is a formal language defined by a grammar; here, parsing is the process of producing a semantically equivalent expression in that grammar from some input. In this case, the input is natural language route instructions, and the parsing target is a statement in RCL that can be passed to a robot controller for planning and execution.

For this work, parsing is performed using an extended version of the Unification-Based Learner, UBL [Zettlemoyer and Collins 2005; Kwiatkowksi et al. 2010].[1] The grammatical formalism used by UBL is a probabilistic version of *combinatory categorial grammars*, or CCGs [Steedman 2000], a type of phrase structure grammar. CCGs model both the syntax (language constructs such as NP for noun phrase) and the semantics (expressions in $\lambda$-calculus) of a sentence. UBL creates a parser by inducing a set of weighted derivation rules and lexical triplets from a set of training examples.

PCCG-based algorithms have several characteristics that make them a good choice for parsing NL instructions. They are robust against noise found in language, and they are able to efficiently generate $n$-best parses using CKY-style parsing [Tsuruoka and Tsujii 2005; Kwiatkowski et al. 2011], allowing for jointly considering a parse model and a world model derived from sensor data when interpreting instructions into grounded action. Next-best parse search can also be used for "fallback" exploration, e.g., when performing local error correction. Additionally, the probabilistic nature

[1] A slightly later version of this system is described in detail in the next chapter.

of PCCGs offers a clear objective for learning, that is, maximizing the conditional likelihood of training data.

Importantly, UBL can learn a parser solely from training data of the form $\{(x_i, z_i)\}$, where $x_i$ is a natural-language sentence and $z_i$ is a corresponding semantic-language sentence. In brief, UBL learns a model for $p(z_i, y_i|x_i; \theta)$, where $\theta$ parameterizes the learned grammar $G$ and $y_i$ is a derivation in $G$ ($z_i$ is completely specified by $y_i$). UBL uses a log-linear model:

$$p(z_i, y_i|x_i; \theta) \propto e^{\theta \cdot \phi(x_i, y_i, z_i)}$$

UBL first generates a set of possibly useful *lexical items*, made up of natural language words, a $\lambda$-calculus expression, and a syntactic category. (An example lexical item might be <"left", `turn-left`, $S$>.) The algorithm then alternates between increasing the size of this *lexicon* and estimating the parameters of $G$ via stochastic gradient descent [LeCun et al. 1998].

Two types of features are used. *Lexical* features fire when the associated lexical items are found in a training example (an example lexical item might be <"left", `turn-left`, $S$>). *Semantic* features are functions of the logical RCL expression $z_i$. These are binary features that indicate the co-occurrence of different types of terms in $z_i$: predicates and their arguments, argument types, predicate co-occurrences, and shared variables. Once a parser is trained, parses are produced via *derivations*, using the learned lexical items and a small set of fixed production rules. Figure 5.4 gives an example derivation of an RCL program (last line) from an input sentence (first line).

### *5.2.3 Initialization of Lexicon and Parameters*

In [Kwiatkowksi et al. 2010], the lexicon—the set of lexical entries and their weights—was initialized with entries covering the entirety of each training example: for each pair of terms found in $(x_i, z_i)$, one initial lexical entry was created. The model defined

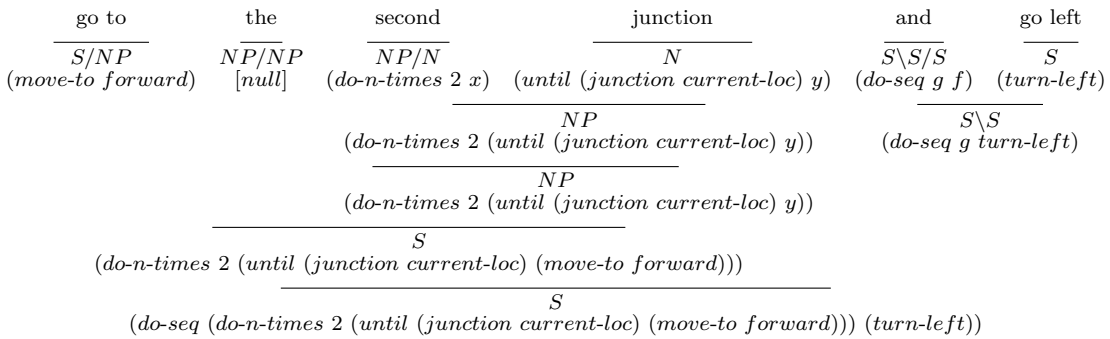



Figure 5.4: CCG parse of a test sentence performed by the learned parser. Natural language input is first, followed by alternating CCG syntactic categorization and $\lambda$-calculus logical forms. The bottom row shows the RCL program to be executed. Some syntax has been changed for readability: some commands are abbreviated, and the $\lambda$-calculus notation has been collapsed to a more LISP-like syntax. For example, *(do-seq g f)* may be equivalently shown as $\lambda g.\lambda f.do\text{-}sequentially(g, f)$.

by $\theta$ contained a parameter corresponding to each lexical item, whose weights were set using cooccurrence statistics of single words in the natural language with constants in the semantic language. For example, given the training example

$$\frac{\text{exit the room and go left}}{(do\text{-}sequentially\ (take\text{-}unique\text{-}exit)\ (turn\text{-}left))}$$

the algorithm would count one cooccurrence for each of ('exit', *do-sequentially*), ('exit', *take-unique-exit*), ('exit', *turn-left*), and each other (NL-word, logical-term) pair. The more often such a pair cooccurred, the more important it was considered for parsing and so the higher its weight in the initial model.

In this work a new initialization process that is better able to handle the execution-

oriented Robot Control Language was implemented. Intuitively, rather than generating lexical items from each word, arbitrary subsequences of natural-language words are considered, along with semantic subexpressions as defined by the splitting procedure of [Kwiatkowksi et al. 2010]. As before, this favors (NL, semantics) pairs that are very predictive of each other, while remaining tractable.

More specifically: for each training example $i$, let $W(x_i)$ be all subsequences of words in the NL sentence $x_i$ (up to a fixed length, $N = 4$ in these experiments). $L_i$ is defined as the initial *sentential CCG category* of the sentence $x_i$, having syntactic category $S$ (sentence) and meaning $z_i$. Splits of $L_i$ into CCG categories are recursively explored to depth two, yielding a set of possible syntactic sub-categorizations $R$. Three count functions are then defined: $C(w \in W)$, instances of phrase $w$ in the training data; $C(r \in R)$, occurrences of each syntactic category in the training data; and $C(w, r)$ as the cooccurrence count. The score for lexical entry $(w, r)$ is then defined to be $p(w|r) + p(r|w)$, where probabilities are computed using counts.

Two other improvements involve synonyms and ordinals. During both training and evaluation, if the induced parser is unable to find a parse for a sentence, it tries to substitute known synonym lists from a standard dictionary for individual words. In addition, numbers are special-cased: when an ordinal (numbers such as 'three', or counting terms such as 'second') is encountered in a training sentence, it is replaced by a standard symbol, turning that training sentence into a *template* into which other ordinals can be substituted freely. As a result, the training data need not contain redundant examples of different ordinals in distinct contexts.

### *5.3 Dataset and Maps*

The primary goal, as in Chapter 4, is to demonstrate the ability of the system to generate RCL command sequences that allow a robot to navigate through a labeled map

according to NL instructions. Maps are labeled according to area type (room, hallway, etc.), but are not known to the robot in advance. Instead, the robot explores the map simultaneously with executing a parser-produced RCL program. These experiments are performed in simulation.

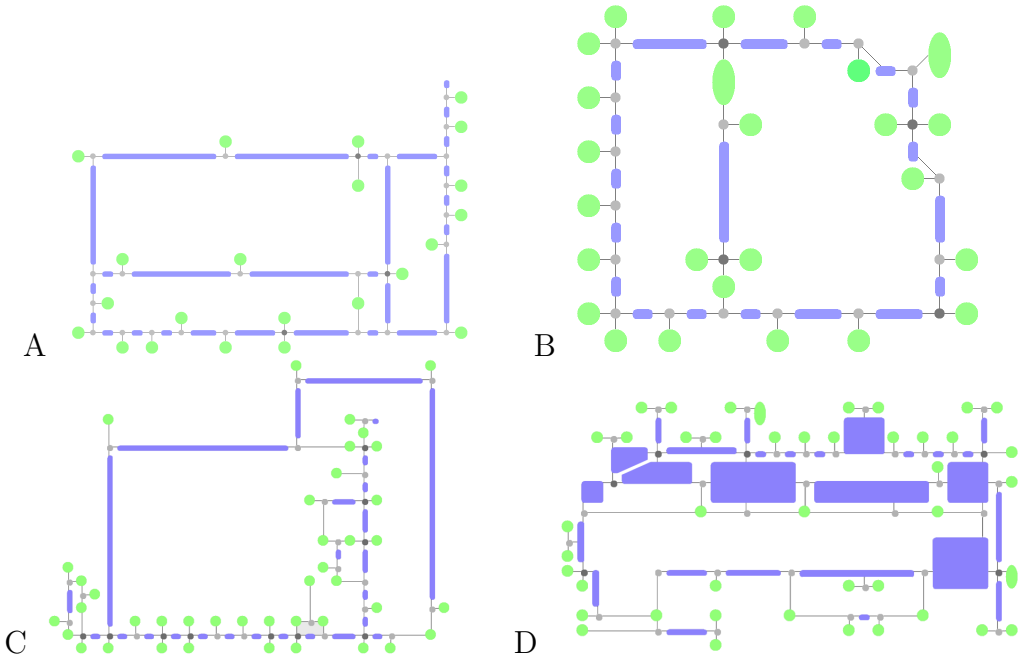



Figure 5.5: Four maps were used for experimental evaluation. A, B, C, and D are respectively a computer science building, an industry research lab, an apartment building, and an academic support building. Buildings C and D were selected for their relatively complex structure. Areas are color-coded according to type: green areas are rooms, blue areas are hallways, and dark and light gray circles are 4-way and 3-way intersections.

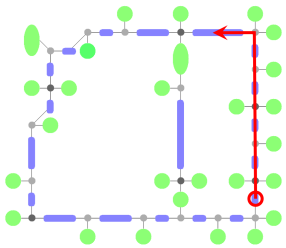
Experiments were conducted on four maps, two each for training and testing (see Figure 5.5). Each map has an associated set of routes through the map that have been described in English; for training and test purposes, each English route description is also paired with an associated RCL annotation. Maps A and B—the training and

testing maps from earlier work—were automatically generated using Voronoi Random Fields [Friedman et al. 2007] on data gathered by a SICK laser range-finder mounted on a Pioneer robot. Because these original maps were fairly simple, two additional manually constructed maps, C and D, were introduced for training and testing, in order to increase experimental complexity. All navigation experiments were performed in simulated execution of RCL commands in these maps.

Language datasets were generated as follows. First, English training and testing data from earlier work [Matuszek et al. 2010] was re-annotated in RCL; as in [Matuszek et al. 2010] and [Chen and Mooney 2011], all English instructions are segmented into individual movement phrases. This training set, $\mathbf{S}_{\text{orig}}$, contained 189 unique sentences generated by non-experts. This was then supplemented with additional sentences taken from descriptions of four more complex routes in map C. Twelve non-experts supplied NL directions for those routes. The resulting *enriched* dataset, $\mathbf{S}_{\text{new}}$, contains 418 NL route instructions along with corresponding RCL command sequences, including structures requiring loops and counting—for example, "Go until you reach a four-way intersection", or "Go into the ninth door on the right". Figure 5.6 shows an example of an NL route instruction set with RCL annotation.

## *5.4 Experimental Evaluation*

The question evaluated in these trials is whether the robot reaches the desired destination. A trial is considered successful only if the robot reached the correct destination along the route intended by the instructor, as the goal is to test instruction-following rather than navigation. Since all maps contain loops, any destination can be reached via an incorrect path; this considered a failed trial. Local knowledge of the map (adjacent and line-of-sight areas) is updated continuously as the robot progresses.



"Go straight down the hallway past a bunch of rooms until you reach an intersection with a hallway on your left; turn left there."

```
(do-sequentially
  (do-until
    (and
      (exists left-loc)
      (hall left-loc))
    (move-to forward-loc))
  (turn-left current-loc))
```

Figure 5.6: An example training/testing triplet. (left) The red line shows the path through map B; (middle) an English description of the path; (right) the language's RCL annotation. In testing, the annotation is compared to the annotation produced by parsing.

### *5.4.1 Baseline, Parser, and Route-Following Experiments*

Three different experiments were used to test the capabilities of the system: two which evaluate its ability to successfully guide a robot through a map, and one which evaluates the parsing component directly. Because the complete number of routes is small and does not necessarily capture map complexity, a large number of test routes of various lengths were generated in the test map, and NL descriptions were generated using route instructions drawn from $\mathbf{S}_{\text{orig}}$ and $\mathbf{S}_{\text{new}}$ (see Figure 5.9 for an example test route and description).

*Baseline*

As a baseline, the robot control system was tested on the data used in Chapter 4. For this experiment, data was collected on Map A, and tested against Map B. Performance was virtually identical to that work, with the simulated robot successfully

| data set | precision | recall | F1 score |
|---|---|---|---|
| enriched | 71.0% | 72.6% | 71.8% |

Figure 5.7: Precision, recall, and F1 on cross-validation tests of the parser.

following 71.4% of route instructions. It is not surprising that the new approach did not significantly improve over the performance of the prior approach, as that data set did not contain complex control structures requiring the additional capabilities provided by RCL. This accuracy was achieved using only collected training data, without amplifying the training set by adding a large number of hypothetical map routes to each natural language instruction as previously.

*Parser Cross-Validation*

In order to evaluate the parser induction system independently, ten-fold cross-validations were run on the enriched dataset $\mathbf{S}_{\text{new}}$, which contain more complex instructions. This test compares the learned parser output against the expert-created RCL annotation, rather than testing the full system against a map, and evaluates performance on individual sentences rather than sets of route instructions. Table 5.7 shows precision, recall, and F1 score. A parse is reported to be correct if it matches the human-supplied RCL program for that phrase, so this test shows the frequency with which the parsing system recovers the exact RCL program of unseen commands rather than allowing for different programs that capture the same semantic intent.

*Route Following with Complex Language*

The primary experiment is a test of following more difficult route instructions through the complex map D. To determine directly whether this framework can be used to

produce executable Robot Control Language, end-to-end tests were performed on a large number of paths through map D in Figure 5.5. For each test, $\mathbf{S}_{\text{new}}$ is split into training and test data by *participant:* all English instructions from some number of non-expert instruction givers ($N = 2$ in our experiments) is withheld from training and used to create a test set $\mathbf{S}_{\text{test}}$.

For each of 10 different test sets (every combination of five different participants), 1,200 paths through map D were randomly generated: 1,000 short paths described by single NL sentences, and 200 paths that required multiple sentences to express (5, on average). An example test path through the map with NL instructions is shown in Figure 5.9. The learned parser is asked to interpret these novel route instructions into RCL commands, which were executed by the simulated robot. Table 5.8 gives a summary of the results.

| data set | success (short) | success (long) |
|---|---|---|
| enriched | $66.3 \pm 10.7\%$ | 48.9% |

Figure 5.8: Testing the end-to-end system on 1,000 short and 200 longer sets of route instruction. Longer routes average five sentences/route; examples of long and short routes and their associated RCL programs can be seen in Figure 5.9 and Figure 5.3(b).

### *5.4.2 Discussion*

Execution of complex language instructions is successful in approximately 66% of short paths, and 49% of complex paths. In general, longer paths are more likely to fail than shorter ones: the simulated control system does not attempt any local error recovery if it encounters an unexpected situation, such as discovering that the

current map location is not of the expected type or failing to find an opening to the left after a `(turn-left)` instruction. Intuitively, this metric does not give any credit for partially correct parses of route instructions, e.g., cases where the robot *almost* reaches the destination.

The high variability of results is likely a product of the experimental setup, in which $\mathbf{S}_{\text{test}}$ is generated from statements by people who did not contribute to training data, making the learned parser vulnerable to forms of instruction that are idiosyncratic to an individual. Given that each person contributed route instructions for only four routes (five sentences each, on average), this is not unlikely.

Performance could be improved by either gathering training data from a wider range of people (for example, using Amazon Mechanical Turk), or by splitting test sets by *path*—allowing an individual to contribute route instructions to both training and testing data, but describing different paths or different maps. This is consistent with the goal of having people use natural language to train their own robots.

Figure 5.4 demonstrates the successful derivation of an RCL command from a short test route instruction using the learned parser. As shown, the parser is able to correctly express concepts such as counting ('second') and can generate differing, contextually correct interpretations of the word 'go' (move vs. turn). Figure 5.9 shows an example of a long path description that the induced parser handles successfully.

### *5.4.3 Discussion and Insights*

Several conclusions can be drawn from these experiments. It is possible to induce a parser that is able to handle complex, procedural natural language commands for robot instruction. This makes it possible to target a rich robot control language capable of handling abstract notions such as counting, loops, and conditionals—semantic constructs that occur in even comparatively simple direction-giving language.

"Go down the long hallway past three intersections, turn left, take the hallway to your left, go through two intersections, turn right, and go forward until you reach the end of the hall."

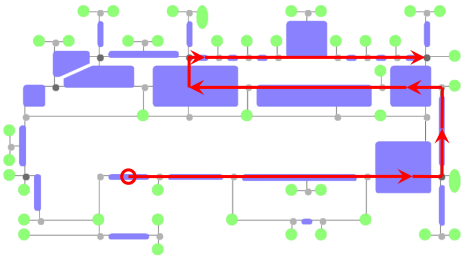


Figure 5.9: (left) Example directions and (right) the corresponding path through map D. The system correctly parses 62.5% of such complex instructions into RCL programs, guiding a simulated robot all the way from the start to the end of such a path.

This chapter describes a system that transforms complex natural-language instructions into a language suitable for use by a robot control system, demonstrating that it is possible to combine advanced natural language processing with robotic control systems, and complements work in mapping such representations to action spaces [Kress-Gazit et al. 2008]. Subsequent chapters focus on the grounding of natural language into robot *perception:* learning about things in the world from human descriptions. This allows for the possibility of reasoning over instructions that include previously unfamiliar context (for example, novel landmarks).

While the effort involved in expert annotation of natural language is far less than that of writing control programs directly, it still requires experts to be involved in the teaching process. Learning solely from execution traces, as in [Chen 2012] and [Artzi and Zettlemoyer 2013], has the potential to increase the efficiency of learning. This is revisited in the next chapter, where (after an initialization phase) annotation of training data is avoided in learning to ground descriptions of object attributes.

# Chapter 6

# LEARNING ABOUT OBJECT ATTRIBUTES FROM NATURAL LANGUAGE

The last two chapters have presented systems that let a simulated agent learn to follow instructions by transforming natural language into different formal control languages. In Chapter 5, the robot learns to parse directions to their semantic representations without using a model of the world in which tasks are performed. Both of these chapters tie into a strong body of work on learning to ground language (Section 2.1) in a variety of domains and world models.

Nonetheless, the formal control language used, both in the relevant literature and in the first half of this dissertation, creates strong assumptions about the nature of that model—for example, by providing a `room` terminal that can be tied to the underlying recognition system, even though the meaning of that terminal must be learned.

The work in this chapter explores the perceptual grounding necessary for building new world models. Intuitively, a physical system can ground language by extending its own underlying world model, creating novel formal representations of novel things. This involves jointly learning models of language and perception, here focused on grounded attribute induction. In addition, this work moves from a simulated perceived map to the Kinect sensors mounted on the robot described in Chapter 3, in order to better

---

The work described in this chapter was originally presented at ICML 2012 [Matuszek et al. 2012a], and was performed in collaboration with Liefeng Bo, Luke Zettlemoyer, Dieter Fox, and especially Nicholas FitzGerald, without whom it would have been neither possible nor enjoyable.

account for the complexity of real-world perception.

## *6.1 Motivation and Overview*

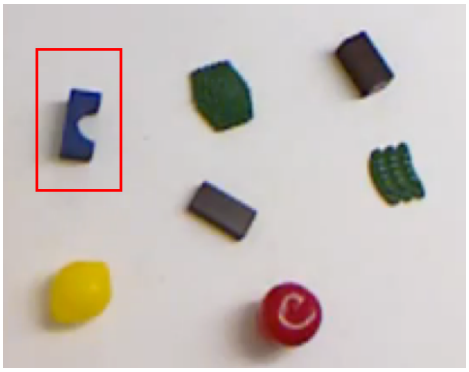
Having access to the physical sensors and systems of a robot makes novel kinds of learning possible. It is now reasonable to imagine that a person might wish and be able to teach a robot about objects in their shared environment. However, to do this, a robot must jointly reason about the different modalities encountered (for example language and vision), and induce rich associations with as little guidance as possible.

Consider a simple sentence such as "These are the yellow blocks," uttered in a setting where there is a physical workspace that contains a number of objects that vary in shape and color. A robot can be said to 'understand' such a sentence if it can solve the associated *grounded object selection task*: picking out those objects being described in language based on data from its visual systems.

In order to accomplish that, it must realize that words such as "yellow" and "blocks" refer to object attributes, and *ground* the meaning of such words by mapping them to a perceptual system that will enable it to identify the specific physical objects referred to. To do so robustly, even in cases where words or attributes are new, such a robot must learn (1) visual classifiers that identify the appropriate object properties, (2) representations of the meaning of words or phrases that incorporate these classifiers, and (3) a model of compositional semantics that allow it to analyze complete sentences.

This chapter describes an effective approach for jointly learning these components, building on existing work on visual attribute classification [Bo et al. 2011a] and probabilistic categorial grammar induction for semantic parsing [Zettlemoyer and Collins 2005; Kwiatkowski et al. 2011]. The perception model includes classifiers for physical characteristics and a language model based on a probabilistic categorial grammar

that enables the construction of compositional meaning representations. Specifically, the system induces new grounded concepts (words or groups of words along with the parameters of the attribute classifier they are paired with) from a set of *scenes* containing sentences, images, and indications of what objects are being referred to (see Figure 6.1 for an example).



"This is a blue thing shaped like a bridge or an arch."

"The blue one."

Figure 6.1: Examples of training data. (left) A training scene, with a red rectangle indicating the language referent. (right) Two different descriptions collected via crowd sourcing.

As a result, it can be taught to recognize previously unknown object attributes by someone describing objects while pointing out the relevant objects in a set of training scenes. Learning is online, adding one scene at a time, and EM-like, in that the parameters are updated to maximize the expected marginal likelihood of the latent language and visual components of the model. This integrated approach allows for effective model updates with no explicit labeling of logical meaning representations or attribute classifier outputs.

Training and testing data was gathered on Amazon Mechanical Turk, in a task in which people describe subsets of objects on a table. Experiments demonstrate that the joint learning approach can effectively extend the set of grounded concepts in

an incomplete model initialized with supervised training on a small dataset. This provides a simple mechanism for semisupervised vocabulary learning with respect to physical objects. The evaluation task is the interpretation of sentences that describe sets of objects in a physical workspace; in this task, the learning system demonstrates accurate task performance and effective latent-variable concept induction in physically grounded scenes.

## *6.2 Approach: Joint Model Learning*

The system described here is able to analyze a sentence, produce a rich, logical representation of its meaning, associate individual parts of the meaning with specific attribute classifiers, and, finally, combine all of these pieces to compute the likelihood of the person referring to any specific set of objects in the world. In this section, the task is defined more formally, followed by descriptions of the individual and jointly learnt models.

More formally, the goal is to learn a joint language and perception model for the object selection task. The goal is to automatically map any natural language sentence $x$ and a set of scene objects $O$ to the subset $G \subseteq O$ of objects described by $x$. The left panel of Figure 6.2 shows an example scene. Here, $O$ is the set of objects present in this scene. The individual objects $o \in O$ are extracted from the scene via visual segmentation (the right panel of Figure 6.2 shows example segments). Given the sentence $x =$"Here are the yellow ones," the goal is to select the five yellow objects for the named set $G$.

### *6.2.1 Model Components*

Given a sentence and segmented scene objects, the goal is to learn a distribution $P(G \mid x, O)$ over the selected set. This can be done by combining existing state of

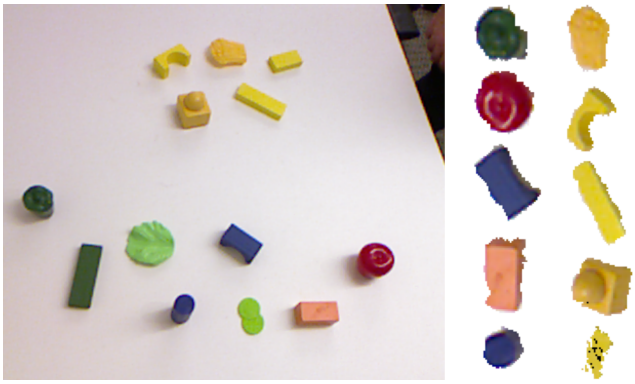


Figure 6.2: An example of an RGB-D object identification scene. Columns on the right show segments, identified as positive (right) and negative (center).

the art models of language and vision, including:

(1) A *semantic parsing* model that defines $P(z|x)$, a distribution over logical meaning representations $z$ for each sentence $x$. In the example, the desired representation $z = \lambda x.color(x, yellow)$ is a lambda-calculus expression that defines a set of objects that are yellow. For this task, the same semantic parsing model [Kwiatkowski et al. 2011] as in Chapter 5 is used, demonstrating its versatility for grounding tasks.

(2) A set of *visual attribute classifiers* $C$, where each classifier $c \in C$ defines a distribution $P(c = true|o)$ of the classifier returning true for each possible object $o \in O$ in the scene—for example, there would be a unique classifier $c \in C$ for each possible color or shape an object can have. Logistic regression is used to train classifiers on color and shape features extracted from object segments recorded using a Kinect depth camera.

### *6.2.2 Joint Model*

The language and vision models are combined in two ways. First, an explicit model of alignment is introduced between the logical constants in the logical form $z$ and

classifiers in the set $C$. This alignment would, for example, enable learning that the logical constant *yellow* should be paired with a classifier $c \in C$ that fires on (gives a positive value for) yellow objects.

The second model introduced is an execution model that determines what scene objects in $O$ would be selected by a logical expression $z$, given the classifiers in $C$. This allows, for example, execution of the statement $\lambda x.color(x, green) \wedge shape(x, triangle)$ by testing all of the objects with the appropriate classifiers (for *green* and *triangle*), then selecting objects on which both classifiers return true. This execution model includes uncertainty from the semantic parser $P(z|x)$, classifier confidences $P(c = true|o)$, and a deterministic ground-truth constraint that encodes what objects are actually intended to be selected. Full details are in Section 6.4.

### *6.2.3 Model Learning*

This system learns the meaning of new words from a dataset $D = \{(x_i, O_i, G_i) \mid i = 1 \ldots n\}$, where each example $i$ contains a sentence $x_i$, the objects $O_i$, and the selected set $G_i$. This is an abstraction of the situation where a teacher mentions $x_i$ while pointing to the objects $G_i \subseteq O_i$ being described. As described in detail in Section 6.5, learning proceeds in an online, EM-like fashion by repeatedly estimating expectations over the latent logical forms $z_i$ and the outputs of the classifiers $c \in C$, then using these expectations to update the parameters of the component models for language $P(z|x)$ and visual classification $P(c|o)$.

To bootstrap the learning approach, a limited language and perception system is first trained in a fully supervised way: at this stage, each example additionally contains annotated logical meaning expressions and classifier outputs, as described in Section 6.5. This approach provides an efficient way to learn from simple demonstrations in a grounded world.

### *6.3 Semantic Parsing Background*

The grounded language learning described in this and the previous chapters incorporates a state of the art model for semantic parsing, which is reviewed in this section. FUBL [Kwiatkowski et al. 2011] is an algorithm for learning factored Combinatory Categorial Grammar (CCG) lexicons for semantic parsing. Given a dataset $\{(x_i, z_i) \mid i = 1...n\}$ of natural language sentences $x_i$, which are paired with logical forms $z_i$ that represent their meaning, UBL learns a factored lexicon $\Lambda$ made up of a set of lexemes $L$ and a set of lexical templates $T$. Lexemes combine with templates in order to form lexical items, which can be used by a semantic parser to parse natural language sentences into logical forms.

For example, given the sentence $x =$ "this red block is in the shape of a half-pipe" and the logical form $z_i = \lambda x.color(x, red) \wedge shape(x, arch)$, FUBL learns a parse like the example in figure 6.3. In this parse, the lexeme (half-pipe, $[arch]$) has combined with the template $\lambda(\omega, \vec{v}).[\omega \vdash NP : \vec{v}_1]$ to yield the lexical item $half\text{-}pipe \vdash NP : arch$.

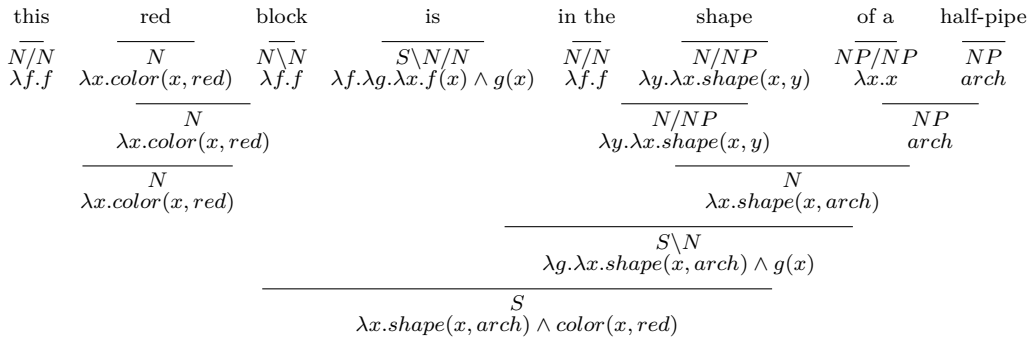



Figure 6.3: An example semantic analysis for a sentence in the dataset.

FUBL also learns a log-linear model which produces the probability of a parse $y$ that yields logical form $z$ given the sentence $x$:

$$P(y, z \mid x; \Theta^L, \Lambda) = \frac{e^{\Theta^L \cdot \phi(x,y,z)}}{\sum_{(y',z')} e^{\Theta^L \cdot \phi(x,y',z')}} \tag{6.1}$$

where $\phi(x, y, z)$ is a feature vector encompassing the lexemes and lexical templates used to generate $y$, amongst other things.

The parse model is initialized using the standard FUBL approach, rather than the modified initialization described in Section 5.2.2. Lexemes are then automatically induced and paired with new visual attributes not present in the initial training set, a process described in the next section.

## *6.4 Joint Language/Perception Model*

The object selection task described in Section 6.2 is to identify a subset of objects, $G$, given a scene $O$ and an NL sentence $x$. A *possible world* $w$ is defined to be a set of classifier outputs, where $w_{o,c} \in \{T, F\}$ specifies the boolean output of classifier $c$ for object $o$. The joint probabilistic model is then:

$$P(G \mid x, O) = \sum_z \sum_w P(G, z, w \mid x, O) \tag{6.2}$$

where the latent variable $z$ over logical forms models linguistic uncertainty and the latent $w$ over possible worlds models perceptual uncertainty.

(6.2) is then further decomposed into a product of models for language, vision, and grounded execution. This final model selects the named objects $G$, motivated in Section 6.2 and described below; the final decomposition is:

$$P(G, z, w \mid x, O) = P(z \mid x) P(w \mid O) P(G \mid z, w) \tag{6.3}$$

The language model $P(z|x)$ and vision model $P(w|O)$ are held in agreement by the conditional probability term $P(G|z,w)$. Let $z(w)$ be the set of objects that are selected, under the assignment in $w$, when $z$ is applied to them. (For example, the expression $z = \lambda x.shape(x, cube) \wedge color(x, red)$ returns true when applied to the objects in $w$ for which the *cube* and *red* logical constant classifiers return true.) $P(G|z,w)$ then forces agreement and models object selection by putting all of its probability mass on the set $G$ that equals $z(w)$.

In this formulation, the language and vision distributions are conditionally independent given this agreement. The semantic parsing model $P(z|x)$ builds on the previous work described in Equation 6.1.

The perceptual classification $P(w|O)$ is defined as follows. Assume each perceptual classifier is applied independently, decomposing this term into:

$$P(w \mid O) = \prod_{o \in O} \prod_{c \in C} P(w_{o,c}|o) \tag{6.4}$$

where the probability of a world is the product of the probabilities of the individual classifier assignments for all of the objects.

Each classifier is a logistic regression model, where the probability of a classifier $c$ on a given object $o$ is:

$$P(w_{o,c} = 1|o; \Theta^P) = \frac{e^{\Theta_c^P \cdot \phi(o)}}{1 + e^{\Theta_c^P \cdot \phi(o)}} \tag{6.5}$$

where $\Theta_c^P$ is the parameters in $\Theta^P$ for classifier $c$. This approach provides a simple, direct way to couple the individual language and vision components to model the

object selection task.

There are two key inference problems in this model. During learning, the marginal distribution $P(z, w|x, O, G)$ over latent logical forms $z$ and perceptual assignments $w$ must be computed (see Section 6.5, next). Then, at test time, the set of objects being referred to can be retrieved by calculating $\arg\max_G P(G|x, O)$.

Computing this probability distribution requires summing the total probability of all world/logical form pairs that name $G$. For each possible world $w$, determining if $z$ names $G$ is equivalent to a SAT problem, as $z$ can theoretically encode an arbitrary logical expression that will name the appropriate $G$ only when satisfied. Computing the marginal probability is then a weighted model counting problem, which is in #-P. However, the logical expressions allowed by the current grammar—conjunctions of unary attribute descriptors—admit efficient exact computation, described below.

## *6.5 Model Learning*

The joint grounded learning problem is to induce a model $P(G|x, O)$, given data of the form $D = \{(x_i, O_i, G_i) \mid i = 1 \ldots n\}$, where each example $i$ contains a sentence $x_i$, the objects $O_i$, and the selected set $G_i$. (Figure 6.1 gives an example of such training data.) The goal is to automatically extend the model to induce new classifiers that are tied to new words in the semantic parser.

In the description of model learning, it is assumed that the learner already has a partial model, including a CCG parser with a small vocabulary and a small existing set of attribute classifiers, provided during the initialization stage. Initialization, in which decoupled models are learned from small datasets with more extensive annotations, is described in Section 6.5.4.

### *6.5.1 Aligning Words to Classifiers*

One key challenge is to learn to create new attribute classifiers associated with unseen words in the sentences $x_i$ in the data $D$. A simple, exhaustive approach is to create a set of $k$ new classifiers, initialized to uniform distributions. Each classifier is additionally paired with a newly generated logical constant in the FUBL lambda-calculus language.

A new lexeme is then created by pairing each previously unknown word in a sentence in $D$ with either one of these new classifier constants, or the logical expression of an existing lexeme in the lexicon. The parsing weights for the indicator features for each of these additions are set to 0. This approach learns, through the probabilistic updates described below, to jointly reestimate the parameters of both the new classifiers and the expanded semantic parsing model.

### *6.5.2 Parameter Estimation*

The language parameters $\Theta^L$ and perception parameters $\Theta^P$ are estimated from training data $D = \{(x_i, O_i, G_i) \mid i = 1 \ldots n\}$, as defined above. The goal is to find parameter settings that maximize the marginal log likelihood of $D$:

$$LL(D; \Theta^L, \Theta^P) = \sum_{i=1 \ldots n} \ln P(G_i | x_i, O_i; \Theta^L, \Theta^P) \tag{6.6}$$

This objective is non-convex due to the sum over latent assignments for the logical form $z$ and attribute classifier outputs $w$ in the definition of $P(G_i|x_i, O_i; \Theta^L, \Theta^P)$ from Equation 6.2. However, if $z$ and $w$ are labeled, the overall algorithm reduces to simply training the log-linear models for the semantic parser $P(z|x_i; \Theta^L)$ and attribute classifiers $P(w|O_i; \Theta^P)$, both well-studied problems. In this situation, it is possible

to use an EM algorithm to first estimate the marginal $P(z, w \mid x_i, O_i, G_i; \Theta^L, \Theta^P)$, then maximize the expected likelihood according to the distribution, with a weighted version of the familiar log-linear model parameter updates. The final system uses an online version of this approach, with updates computed one example at a time.

*Computing Expectations:* For each example $i$, it is necessary to compute the marginal over latent variables given by:

$$P(z, w \mid x_i, O_i, G_i; \Theta^L, \Theta^P) = \frac{P(z \mid x_i; \Theta^L) P(w \mid O_i; \Theta^P) P(G_i | z, w)}{\sum_{z'} \sum_{w'} P(z' \mid x_i; \Theta^L) P(w' \mid O_i; \Theta^P) P(G_i | z', w')} \tag{6.7}$$

Since computing all possible parses $z$ is exponential in the length of the sentence, beam search is used to find the top-N parses. This exact inference could be replaced with an approximate method, such as MC-SAT, to accommodate a more permissive grammar.

*Conditional Expected Gradient:* For each training datum presented, the parameters are updated with the expected gradient, according to the marginal distribution above. For the language parameters $\Theta^L$, the gradient is

$$\begin{aligned} \Delta^L = \sum_{z'} \sum_{w'} & P(z', w' \mid x_i, O_i, G_i; \Theta^L, \Theta^P) * \\ & \left(E_{P(y | x_i, z'; \Theta^L)} \left[\phi_j^L(x_i, y, z')\right] - \right. \\ & \left. E_{P(y, z | x_i; \Theta^L)} \left[\phi_j^L(x_i, y, z)\right]\right) \end{aligned} \tag{6.8}$$

where the inner difference of expectations is the familiar gradient of a log-linear model for conditional random fields with hidden variables [Quattoni et al. 2007; Kwiatkowksi et al. 2010], and is weighted according to the expectation.

Similarly, for the perception parameters $\Theta^P$, the gradient is:

$$\Delta_c^P = \sum_{z'} \sum_{w'} P(z', w' \mid x_i, O_i, G_i; \Theta^L, \Theta^P) * \sum_{o \in O_i} \left[ w'_{o,c} - P(w'_{o,c} = 1 \mid \phi(o); \Theta^P) \right] \phi(o) \tag{6.9}$$

where the inner sum ranges over the objects and adds in the familiar gradient for logistic regression binary-classification models. [Bishop 2007].

*Online Updates:* A simple, online parameter estimation scheme loops over the data $K = 10$ (picked on validation set) times. For each data point $i$ consisting of the tuple $(x_i, O_i, G_i)$, an update is performed in which a step is taken according to the above expected gradient over the latent variables. A learning rate of 0.1 with a constant decay of .00001 per update performs well for all experiments.

### *6.5.3 Model Learning Performance*

This complete learning approach provides an efficient online algorithm that closely matches the style of interactive, grounded language learning being pursued in this work. Given the decayed learning rate, the algorithm is guaranteed to converge, but little can be said about the optimality of the solution. However, the approach works well in practice for the object set selection task defined in Section 6.6.

### *6.5.4 Model Initialization*

To construct the initial limited language and perceptual models, supervised learning is performed over a small data set $D_{sup} = \{(x_i, z_i, w_i, O_i, G_i) \mid i = 1 \ldots m\}$, which matches the training data previous setup but additionally labels the latent logical

form $z_i$ and classifier outputs $w_i$. Learning in this setting is completely decoupled, and it is possible to estimate the semantic parsing distribution $P(z_i|x_i;\Theta^L)$ with the FUBL learning algorithm [Kwiatkowski et al. 2011] and the attribute classifiers $P(w_i|O_i;\Theta^P)$ with gradient ascent for logistic regression.

Experimentally, $D_{sup}$ can often be quite small, and in general does not contain many of the words and attributes that must be additionally learned during training. Nonetheless, this initialization represents a step that cannot be performed without technical expertise. Later chapters explore aspects of human interaction that would be involved in replacing $D_{sup}$ with interactive dialog with a human teacher.

## *6.6 Experimental Evaluation*

### *6.6.1 Experimental Setup*

Vision and language data was collected using a selection of toys, including wooden blocks, plastic food, and building bricks. Short RGB-D videos taken with the robot's Kinect depth camera were collected, each showing a person gesturing to a subset of the objects (see Figure 6.4).

Natural language annotations were gathered using Mechanical Turk; workers were asked to describe the objects being pointed to in the video. The referenced objects were then marked as belonging to $G$, the positive set of objects for that scene. A total of 142 scenes were shown, eliciting descriptions of 12 attributes, divided evenly into shapes and colors. In total, there were 1003 sentence/annotation pairs.

To automatically segment objects from each scene, RANSAC plane fitting was performed on the Kinect depth values to find the table plane, then connected components (segments) of points more than a minimum distance above that plane were extracted. After segmenting objects, features for every object were extracted using kernel de-

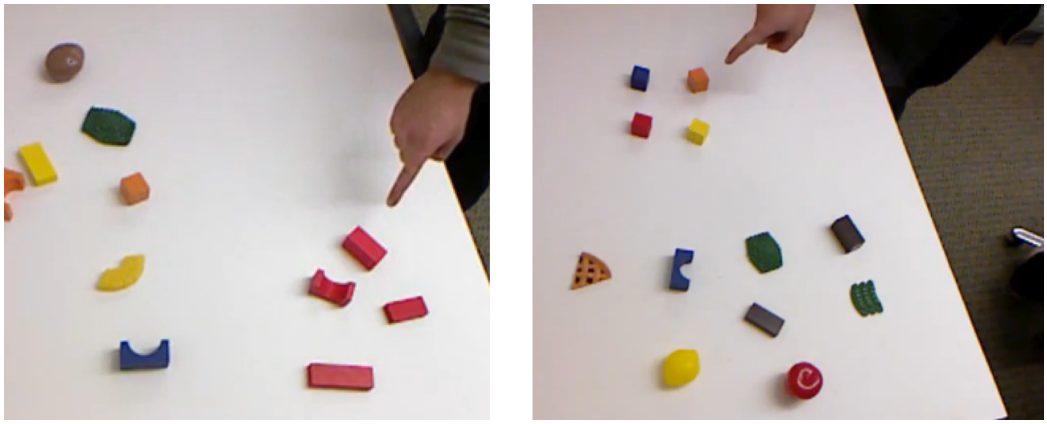


Figure 6.4: Example scenes presented on Mechanical Turk. (left) A scene that elicited the descriptions "here are some red things" and "these are various types of red colored objects", both formally annotated $\lambda x.color(x, red)$. (right) A scene associated with sentence/representation pairs such as $\langle$"this toy is orange cube", $\lambda x.color(x, orange) \wedge shape(x, cube)\rangle$.

scriptors [Bo et al. 2011a]. Two types of features, for depth values and RGB values, were extracted; these correspond to shape and color attributes, respectively. During training, the system learns logistic regression classifiers using these features. In the initialization phase used to bootstrap the model, the annotation additionally provides information about which language attributes relate to shape or color. However, this information is not provided in the training phase.

A standard set of binary indicator features was to define the log-linear model $P(z|x; \Theta^L)$ over logical forms, given sentences, as in the original FUBL algorithm analysis [Kwiatkowski et al. 2011]. This includes indicators for what lexical entries were used and properties of the logical forms that are constructed. These features allow the joint learning approach to weight lexical selection against evidence provided by the compositional analysis and the visual model components.

### *6.6.2 Results*

This section presents results and a discussion of the experimental evaluation. Effective learning is demonstrated in the full model for the object set selection task; ablation studies and examples of learned models are described briefly.

#### *Object Set Selection*

To measure set selection task performance, the data was divided according to attribute. To initialize the model, six of the attributes were used to train supervised classifiers, and logical forms for the corresponding sentences were used to train the initial semantic parsing model. Data for the remaining six attributes were used for evaluation, with 80% allocated for training and 20% held out for testing. During training, all visual scenes are previously unseen, the words in the sentences describing the new attributes are unknown, and the only available labels are the output object set $G$.

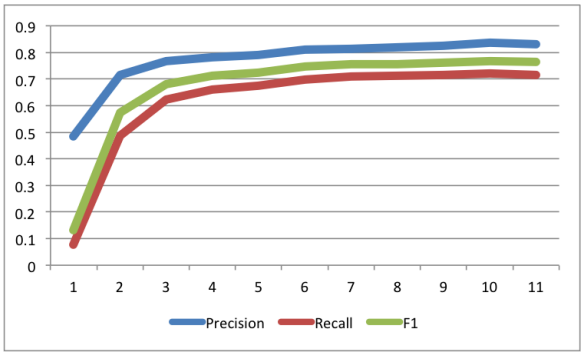



Figure 6.5: Performance on the set selection task, averaged over three different concept splits. Here, the $y$-axis is precision, recall, and F-measure for the joint model, and the $x$-axis is number of training epochs.

Figure 6.5 shows precision, recall, and F1-score on the set selection task. Results are averaged over 10 different runs with the training data presented in different randomized orders. The system performs well, achieving an average precision of 82%, recall of 71%, and a 76% F1-score. This level of performance is achieved relatively quickly; performance generally converges within five passes over the training data.

### *6.6.3 Ablation Studies*

To examine the need for a joint model, two models are evaluated in which either the language or the visual component is sharply limited. In each case, performance significantly degrades. These results are summarized in Figure 6.6.

*Language Ablation:* In order to measure how a set of classifiers would perform on the set selection task with only a simple language model, the joint model was tested with a language component consisting of a thesaurus of words used in the dataset to refer to target attributes. To learn the unsupervised concepts for this baseline, a list of all words appearing in the training corpus but not in the initialization data was extracted. Words which appear in the thesaurus are grouped into *synonym sets*.

To train classifiers, objects were collected from scenes in which only terms from the given synonym set appear. Any synonym set which does not occur in at least 2 distinct scenes is discarded. The resulting positive and negative objects are used to train classifiers. To generate a predicted set of objects at test time, all synonym sets which occur in the sentence $x$ are found and used to determine whether the classifiers associated with those words successfully identify the object.

Averaged across trials, the results are as follows: Precision=0.92; Recall=0.41; F1-score=0.55. These results are, on average, notably worse than the performance of the jointly trained model.

*Vision Ablation:* As a baseline for testing how well a pure parsing model will perform when the perception model is ablated, the parsing model obtained during initialization was run directly on the test set, training no new classifiers. Since the parser is capable of generating parses by skipping unknown words, this baseline is equivalent to treating the unknown concept words as if they are semantically empty.

Averaged across trials, the results are as follows: Precision=0.52; Recall=0.09; F1-score=0.14. Not surprisingly, a substantial number of parses selected no objects, as the parser has no way of determining the meaning of new attribute words.

| | Precision | Recall | F1-Score |
|---|---|---|---|
| Language Ablation | **0.92** | 0.41 | 0.55 |
| Vision Ablation | 0.52 | 0.09 | 0.14 |
| **Full Joint Model** | 0.82 | **0.71** | **0.76** |

Figure 6.6: A summary of precision, recall, and F1 for ablated and joint models.

### *6.6.4 Discussion and Examples*

This section discusses typical training runs and data requirements. Examples of learned models area presented, highlighting what is learned and typical errors, and then simple experiments are described investigating the amount of supervised data required for initialization.

Classifier performance after training effects the system's ability to perform the set selection task. During a sample trial, average accuracy of color and shape classifiers for newly learned concepts are 97% and 74%, respectively. Although these values are sufficient for reasonable task performance, there are some failures—for example, the shape attributes "cube" and "cylinder" are sometimes challenging to differentiate.

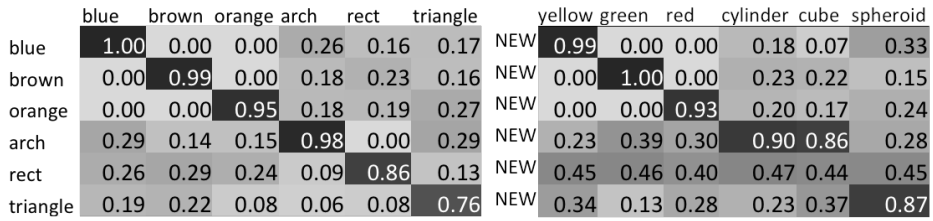



Figure 6.7: Confusion matrices showing the accuracy of initialized classifiers (left) and newly learned classifiers (right) on test objects. Each entry is the F1 score of the learned classifier (listed on y-axis) when tested on sets of hand-labeled objects with specific attributes (listed on x-axis).

Failures of the system can be attributed to several causes. Classification errors can lead to incorrect outputs. This was particularly common where two attributes were not well distinguished by the learned classifier. For example, Figure 6.7 shows an example where the learned 'cylinder' and 'cube' classifiers were not well differentiated. This can lead to seemingly nonsensical output parses where two shapes are asserted (ie. $\lambda x$. (shape($x$, cylinder) $\wedge$ shape($x$,cube))) because the classifiers can simultaneously fire. These cases are rare, since in almost all cases the inclusion of a second shape will decrease the overall likelihood of the possible world defined by the parse.

As described in the overview of parsing (Section 6.3), the semantic parser contains lexemes that pair words with learned classifiers, and features that indicate lexeme use during parsing. Figure 6.8 shows selected word/classifier pairs, along with the weight for their associated feature (each trial produces a large number of such lexemes).

The classifiers NEW0 through NEW2 and NEW3 through NEW5 are color and shape classifiers, respectively. As can be seen, each of the novel attributes is most strongly associated with a single newly created classifier, while irrelevant words such as "thing" tend to parse to null; this is precisely the desired behavior. The system must learn from training examples what classifiers and classifier to associate with novel words.

Additional tests demonstrate the system's ability to learn synonyms. The data was

| | NEW0 | NEW1 | NEW2 | NEW3 | NEW4 | NEW5 | null |
|---|---|---|---|---|---|---|---|
| “red” | 3.27 | -0.34 | -0.37 | -0.16 | -0.16 | -0.17 | 0.00 |
| “green | -0.39 | -0.30 | 3.47 | -0.19 | -0.19 | -0.19 | 0.00 |
| “blue” | -0.34 | 2.97 | -0.31 | -0.16 | -0.16 | -0.16 | 0.00 |
| “thing” | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.29 |
| “cube” | -0.43 | 0.31 | -0.37 | -0.23 | 0.00 | 2.78 | 0.00 |
| “that” | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.42 |
| “arch” | -0.01 | -0.01 | 0.09 | -0.14 | 0.6 | -0.15 | 0.00 |
| “triangle” | 0.34 | -0.30 | 0.04 | 1.92 | -0.18 | -0.19 | 0.00 |
| “toys’ | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.38 |

Figure 6.8: Feature weights for hypothesized lexemes pairing natural language words (rows) with newly created terms referring to novel classifiers (columns), as well as the special null token. Each weight serves as an unnormalized indicator of which associations are preferred.

split so that the training set contains attributes learned during initialization that are referred to by new, synonymous words. These runs performed comparably to those reported above; the approach easily learns lexemes that pair these new words with the appropriate classifiers.

Finally, experiments were performed on the effects of reducing the amount of annotated data used to initialize the language and perception model (see Figure 6.9). As can be seen, with fewer than 150 sentences, the learned grammar does not appear to have sufficient coverage to model unknown words in joint learning; however, beyond that, performance is quite stable. This relatively small number increases the likelihood that it will be possible to replace initialization with interactive training.

## *6.7 Discussion and Conclusions*

This system is able to learn accurate language and attribute models, given data containing only language, raw percepts, and the target objects, by jointly learning language and perception models—a powerful demonstration of the effectiveness of

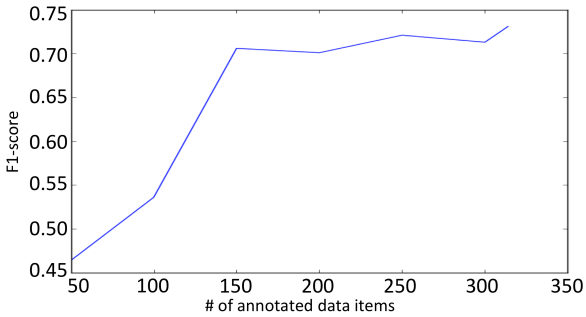



Figure 6.9: Example F1-score on object recognition from models initialized with reduced amounts of labeled data, reported over one particular data split. The F1-score for this split peaks at roughly 73%.

grounded language acquisition. In the first half of this thesis, natural language instructions were being grounded by learning their relation to an existing formal execution language; the system described in this chapter demonstrates grounding language directly into objects that are perceived in the physical world, using real-world sensors mounted on an non-simulated robot.

Extending the robot's formal representation of the world based on language provided by non-experts is a significant contribution, demonstrating as it does the potential for robots to learn concepts that were not considered by their programmers. The next two chapters further investigate this concept, exploring questions raised by the process of gathering data from human teachers.

# Chapter 7

# LEARNING FROM UNSCRIPTED DEICTIC GESTURE AND LANGUAGE

Statistical approaches to parser induction, and to learned natural language processing generally, rely heavily on having corpora of training data available. The work described in this dissertation is no exception. In Chapter 4 and Chapter 5, natural language describing paths through a map was collected from a large number of volunteers; in the former, it was also remixed to create an artificially amplified training corpus. In Chapter 6, NL descriptions of objects in scenes were collected via crowdsourcing. In all cases, the language collected for these tasks was written or typed.

While convenient, this style of interaction is not necessarily consistent with how individuals will most effectively communicate with robots. In the physical world, people are more likely to provide instructions or descriptions using face-to-face modalities such as spoken language, with associated side channels such as gaze and gesture.

This chapter investigates how people refer to objects in the world during unstructured, real-world communication. A corpus of deictic interactions was collected from users describing objects and used to train language and gesture models. The learned models are combined into a system that interprets what objects are being referred to and how. The task setting—describing blocks on a table—is the same as in the previous chapter, but the communication is more realistic and complex.

---

The work described in this chapter was originally presented at AAAI 2014 [Matuszek et al. 2014], and was performed in collaboration with Liefeng Bo, Luke Zettlemoyer, and Dieter Fox.

## *7.1 Motivation and Overview*

Much of the work described in this dissertation is motivated by the goal of building natural, intuitive human-robot interfaces. While it is possible to teach someone how to control a robot for specified tasks, a more intuitive robot will learn how users naturally use speech and gesture. In turn, having access to the physical world allows for learning to understand natural language pertaining to that world, such as descriptions of visual percepts. This is the core of the grounded language acquisition problem.

In practice, people do not use language in a vacuum, but rather incorporate information from gaze, gesture, body language, and so on. This chapter presents a new RGB-D corpus of interactions from users identifying objects in a space (described in detail in Section 7.2.1), which was used to train models of interactions. A proof of concept system in which a robot interacts with objects according to user instructions was implemented (see Figure 7.1).

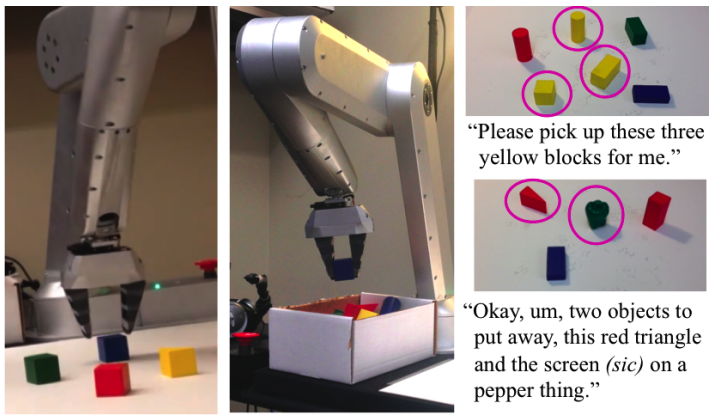



Figure 7.1: The integrated system. (left) and (center): the Gambit robot picking up a toy and placing it in the bin; (right) examples of the target objects in test scenes, with language used by test participants (including errors).

In the previous chapter, workers were asked to provide descriptions of objects. A component of the training data provided for learning about attributes was a specification of what objects were being described ($G_i$). In the new corpus, this data is generally provided using gesture—specifically, deictic gesture of the sort Clark [Clark 2003] calls *directing-to*, or attention-focusing: language and gesture in which the user is drawing attention to specific objects.

This corpus of gestures is used to train models that allow the robot to determine what objects are being indicated. A temporal extension to state of the art hierarchical matching pursuit features [Bo et al. 2011b] to support gesture understanding is described, and it is experimentally demonstrated that combining multiple communication modalities more effectively capture user intent than relying on a single type of input. Initial interactions with a robot that uses the learned models to follow commands are presented, with error analysis.

## *7.2 Problem Statement and Data Collection*

The task is to build a system that understands someone who is indicating objects in a workspace, by verbally describing any combination of object attributes, by gesturing, or both—a system that learns models of language, deictic gesture, and visual attributes from examples. This combination of modalities performs well at capturing users' intent, as demonstrated on a robot that understands the intentions of test participants as well as in evaluations of the effectiveness of individual classifiers on the set selection task.

The language and gesture in this corpus are unscripted, i.e., not predefined by experimenters; people did what came naturally during training and evaluation. Training a classifier intended to interpret any arbitrary deictic gesture in a tabletop manipulation setting is qualitatively different from gesture recognition, in which a set of known

gestures must be disambiguated. This is, to the best of our knowledge, the first body of work on using machine learning to interpret unscripted deictic gesture, and is a contribution of this chapter.

### *7.2.1 Data Collection and Corpus*

Data collection is similar to that in Chapter 6: a group of people were asked to describe a subset of objects in a scenario. The primary difference is that data collection was performed in a shared physical workspace. For each scenario, the first goal is to identify positive (indicated) objects in the scene. In order to collect information about how people refer to objects when given few constraints, a Kinect sensor was used to record RGB-D videos of people describing objects.

Participants were instructed to distinguish objects from a scene, using language and gesture "as if they were describing those objects to a robot," but not given a predefined set of gestures or instructions to use. Experimenters specified what objects to indicate (see Figure 7.2) in both training and testing. The data set contained 234 scenes in which two or more objects were to be indicated.

Language from the training corpus was hand-transcribed, although in the experimental evaluation speech recognition is used. Language is not temporally aligned with gesture; instead the entire time-series is analyzed for deictic motions. This presents a noisier learning problem, but is consistent with the collected data, in which gestures do not always correspond temporally to language.

Scenes were designed to collect data for objects with different spatial relations and combinations of attributes. The length of responses from participants ranged from very brief (around three seconds) to more than a minute. Thirteen participants described each scene. The resulting data set contains examples of language used without gesture, gesture paired with non-descriptive language (e.g., "These objects"),

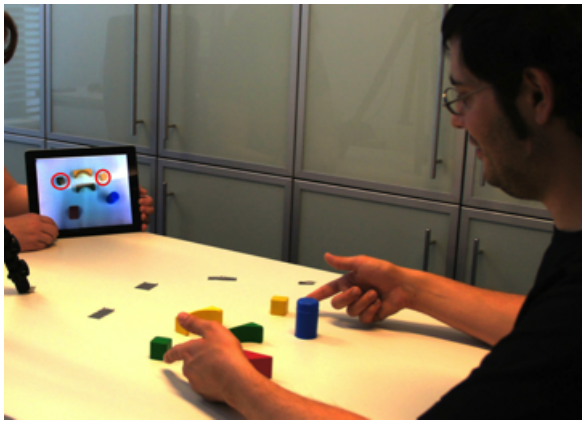


Figure 7.2: A data collection participant indicating objects. The experimenter shows an iPad with target objects circled in order to avoid linguistic or gestural priming.

and gesture and language used together (examples can be seen at: `http://tiny.cc/Gambit14`).

## *7.3 Approach: Vision, Gesture and Language Models*

To accomplish the goal of identifying objects from whatever interactions a user offers, vision and language classifiers were trained separately on the training corpus. (Users might choose refer to objects by verbally describing any combination of color and shape attributes, by gesturing, or both.)

Accordingly, for each object, whether it is gestured to and whether it is referred to verbally are combined into a final per-object score, which is used to determine what objects in a scene a person wishes to indicate. The pipeline is as follows:

1. Raw sensor input is processed to identify the workspace, objects, and hands, and language is processed into text.

2. If hands are present, a gesture classifier is used to obtain a probability that each object was gestured to.

3. Vision classifiers return probabilities for the color and shape of each object.

4. Language is combined with the output of the visual attribute classifiers to determine whether each object is referred to in speech.

5. The results of gesture and language analysis are combined into a final *object score* for each object, representing whether the system believes it is a positive object (that is, was somehow indicated by the user).

The individual classifiers used are defined, followed by discussion of how their outputs are synthesized into an evaluation of user intent.

### *7.3.1 Point-Cloud Processing*

In this first step, the system extracts the user's hands and the objects from each frame of the RGB-D video. To automatically segment individual objects $o \in O$ from each scene, RANSAC plane fitting is used to remove the table plane, then connected components (segments) of points above that plane are extracted. Remaining points are segmented into clusters (Figure 7.5 shows an example of a segmented object).

Gestures are identified using spatial relationships between the user's hands and the objects over time. For each frame in a scene, the leading edge of a hand is identified (the point on the hand that is furthest from the user's body, from the viewpoint of the interlocutor, see Figure 7.3). Distance between this point and an object is then computed along each of the three coordinate axes and as a single Euclidean distance, resulting in a four-dimensional distance vector at each timestep.

For this work, the hierarchical matching pursuit (HMP) approach of [Bo et al. 2011b]

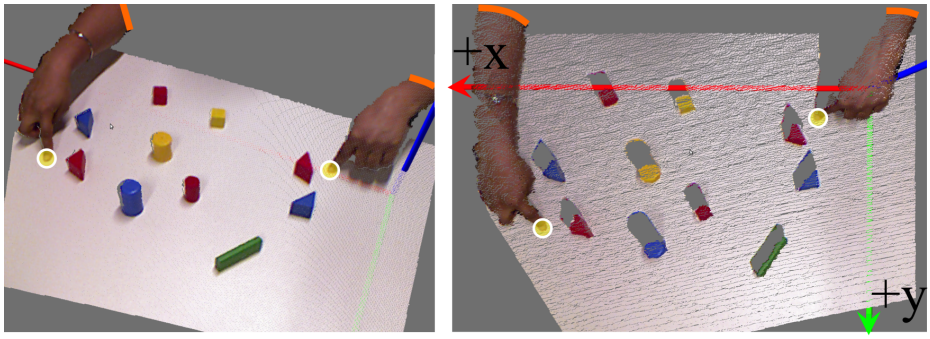


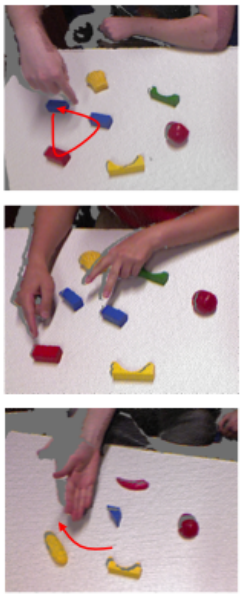

Figure 7.3: A frame from the data corpus, cropped to the workspace (left), then ***x,y*** axis-aligned to the table (right). Moving clusters of points that extend beyond the workspace are hands (boundaries in orange). The leading-edge point on each hand (yellow) is used to calculate gesture features.

is used, which uses the matching pursuit encoder to build a feature hierarchy layer by layer, rather than relying on hand-crafted features. HMP has proven to be on par with state of the art for object recognition, scene recognition, and static event recognition. The experiments described in this chapter demonstrate that it can be extended to dynamic classification problems.

### *7.3.2 Gesture Classification*

Once the user's hands and the objects on the table are extracted for the complete video sequence, the system tries to determine for each object whether the user is gesturing to that object. This task turns out to be quite complex, due to the substantial variability in how people use deictic gesture [Clark 2005]; Figure 7.4 shows some examples.

For temporal reasoning, the HMP architecture is adapted to perform pooling over the temporal sequence of the gesture, rather than the 2D grid of an image as in [Bo et al. 2011b]. The key novelty is in introducing a binary encoding that maintains important scale information. As described above, for each object, a sequence of spatial features



"Three rectangles. These three. Two of them are blue. Two blue red rectangles and one red rectangle."

"Okay now I have three objects... One two three. They both they all three have the same… shape. Like little thick sticks with some rounded corners. But those two are blue and this one is red."

"The yellow objects."

Figure 7.4: Examples of unscripted gesture and language. (top left) A circular pointing motion looping around the objects; (middle left) pointing with multiple fingers and both hands; (bottom left) an open-handed sweep above objects. (right) Examples of collected language for the two scenarios shown. Errors ("two blue red rectangles") are typical of spoken language.

is extracted from the video: the distance of the hand's leading edge from that object, and the hand's direction relative to the centers of objects in $x, y$, and $z$, giving a four dimensional vector at each timestep. Rather than applying a classifier directly to this sequence, *sparse coding* is used. This has become a popular tool in several fields, including signal processing and object recognition [Yang et al. 2009], to learn rich features that are more suitable for the classification task.

Sparse coding is an approach to finding rich representations of an input signal $Y$; from unlabeled input data, functions are discovered that capture higher-level features in the data, making it unnecessary to manually model those features. More specifically, sparse coding models data as sparse linear combinations of codewords selected from

a codebook $D$, which is trained using the efficient dictionary learning algorithm K-SVD. The key idea is to learn the codebook: a set of vectors, or *codewords*. Data can then be represented by a sparse, linear combination of these codebook entries.

Since the gesture recognition task entails time-series data of no fixed length, it is necessary to operate over the collection of four-dimensional vectors from the entire time-series $T$, the sequence of frames in which a person was gesturing. Here, the efficient dictionary learning algorithm K-SVD and orthogonal matching pursuit are used to build high-level features.

K-SVD finds the codebook $D = [d_1, \cdots, d_M] \in R^{H \times M}$ and the associated sparse codes $X = [x_1, \cdots, x_N] \in R^{M \times N}$ from a matrix $Y = [y_1, \cdots, y_N] \in R^{H \times N}$ of observed data by minimizing the reconstruction error:

$$\begin{aligned} &\min_{D,X} \|Y - DX\|_F^2 \\ &s.t. \ \ \forall m, \ \ \|d_m\|_2 = 1 \ \text{ and } \ \forall n, \ \ \|x_n\|_0 \leq K \end{aligned} \tag{7.1}$$

where $H$, $M$, and $N$ are the dimensionality of codewords, the size of codebook, and the number of training samples, respectively. $\|\cdot\|_F$ denotes the Frobenius norm, while the zero-norm $\|\cdot\|_0$ counts non-zero entries in the sparse codes $x_n$. $K$ is the sparsity level controlling the number of non-zero entries.

Classic K-SVD normalizes the $L_2$ norm of each codeword to be 1, so learned codebooks don't capture magnitude information of input data. This is a useful property for image recognition, as spatial pooling and contrast normalization over sparse codes can generate features robust to lighting condition changes [Bo et al. 2011b]. However, magnitude information is critical for gesture recognition. To allow codewords to encode magnitude information, we remove normalization constraints and limit sparse codes to be binary values:

$$\min_{D,X} \|Y - DX\|_F^2 \tag{7.2}$$
$$s.t.\ \ \forall n,\ \ x_n \in \{0,1\}^M \ \text{ and } \ \|x_n\|_0 \leq K$$

We design a K-SVD-like algorithm to decompose the above optimization problem into two subproblems, encoding and codebook updating, which are solved in an alternating manner. ENCODING: At each iteration, the current codebook $D$ is used to encode the data input $Y$ by computing the sparse code matrix $X$, using matching pursuit [Mallat and Zhang 1993]. UPDATE: Then, the codewords of the codebook are updated one at a time by gradient descent optimization, giving a new codebook. The new codebook is used in the next iteration to recompute the sparse code matrix followed by another round of codebook updating, repeated to maximum iterations.

With the learned codebook, we are able to generate features representing the whole gesture sequence. Since a gesture that points to a specific object could occur in any timestep, we perform temporal max-pooling by maximizing each component of binary sparse codes of the four dimensional vectors over all timesteps, thereby generating features robust to temporal changes:

$$f_G = \left[\max_{j \in T} |x_{j1}|, \cdots, \max_{j \in T} |x_{jM}|\right] \tag{7.3}$$

We run logistic regression over these features to train a classifier, which can be run over a new video sequence of gestures and objects. For each object $o$, this classifier gives $h_o$, the probability that object $o$ is being gestured to:

$$P(h_o|\Theta^P) = \frac{e^{\Theta_g^P}}{1 + e^{\Theta_g^P}} \tag{7.4}$$

where $\Theta_g^P$ are the parameters in $\Theta^P$ for the gesture classifier.

### *7.3.3 Color and Shape Classification*

The goal of the visual classifier described here is to determine what attributes each object in a scene has. More formally, for each color and shape present in the corpus, the goal is to return a probability that each object has that attribute. Let $C$ and $S$ be sets of discrete symbols which denote known colors and shapes, respectively:

$C = \{blue, green, red, yellow\}$
$S = \{arch, ball, cube, cylinder, rectangle, triangle, other\}$

Because the focus of this work is on how gesture can be interpreted and used with language, the same simple set of attributes is used as in the previous chapter. However, note the use of a lower resolution RGB-D camera and the need to learn classifiers that correspond to noisy language (for example, an apple, a potato, and a pepper all referred to as "round"). The relatively standard assumption applies that the objects we are working with are separated on a planar surface detectable using RANSAC.

To perform attribute classification, a similar method is used as for gesture classification, without temporal extensions. At a high level, the goal is to find a set of higher-level features from the raw camera inputs that give good predictive power for the target attributes. This means training a binary classifier for each attribute. On each object, HMP features are extracted, and the training inputs $Y$ are tiled sub-squares of the images.

Here, spatial max pooling instead of temporal max pooling is applied with the learned codebooks to aggregate the sparse codes. These features, drawn from the training scenes in the corpus, are used to train binary classifiers for each attribute. Figure 7.5 gives example inputs and outputs.

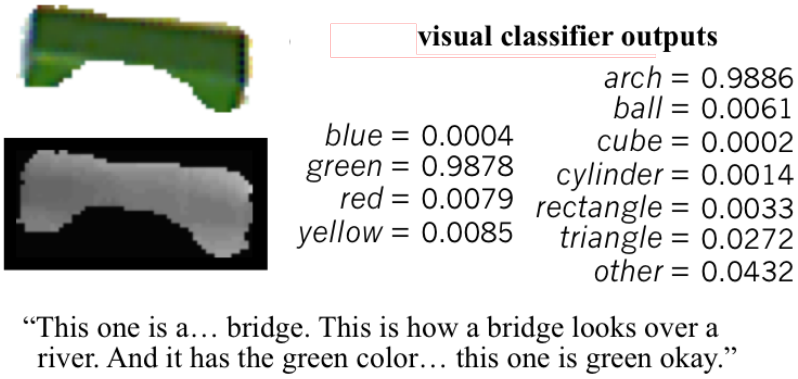



Figure 7.5: Data for one object in a scene. (top left) and (middle left): Images showing the comparatively low-resolution RGB and depth signal from the camera; (right) the output of the visual shape and color classifiers. (bottom) the spoken description.

K-SVD is applied as described in Section 7.3.2 to learn features for RGB-D images. Input data are now collections of 8×8×3 image patches and 8×8×1 depth patches, rather than four-dimensional distance vectors. An object image is tiled into into cells, and the features of each cell $E$ are the max pooled sparse codes, the component-wise maxima over all sparse codes within a cell:

$$f_I(E) = \left[\max_{j \in E} |x_{j1}|, \cdots, \max_{j \in E} |x_{jM}|\right] \tag{7.5}$$

Here, $j$ ranges over all entries in the cell, and $x_{jm}$ is the $m$-th component of the sparse code vector $x_j$ of entry $j$. The feature $f_I$ describing an image $I$ is the concatenation of aggregated sparse codes in each spatial cell:

$$f_I = \left[F(E^1), \cdots, F(E^P)\right] \tag{7.6}$$

where $E^p$ is a spatial cell generated by spatial partitions, and $P$ is the total number of spatial cells.

Once these features are calculated, they are again used to train standard logistic regression classifiers for each attribute, which give probabilities for whether an object possesses each attribute. The outputs of individual classifiers are denoted $V \in \mathbb{R}$, where $v_{o,c}, c \in C$ and $v_{o,s}, s \in S$ are the outputs of appropriate color or shape classifiers for object $o$.

### *7.3.4 Language Model*

At a high level, the goal of the language analysis step is to determine, for each object in a scene, whether the user is indicating that object. Intuitively, the likelihood $r_o$ of an object $o$ having been "referred to" depends on whether it has attributes the user mentions. The inputs for the language classifier are the language $L$, containing the user's description of the scene, plus the output of the attribute classifiers (for example, if $L =$ "The blue one" and $v_{o_1,blue}$ has a high value, $r_{o_1}$ should be high).

$$r_o = P(o|v_{i\in\{C,S\},o}, L) \tag{7.7}$$

In keeping with the goal of allowing free-form user input, the system does not use *a priori* information about what words correspond to what attributes; instead, such correspondences are learned from training data. This allows the system to learn interpretations of unexpected words, such as the use of "parcel" to describe cube-shaped blocks (Figure 7.6).

From the language $L$ for all training scenes, a bag-of-words feature vector is extracted from $W$, the set of all words found in the corpus. (While this relatively simple language model introduces some errors, in general it works well for this data set, per the failure analysis in Section 7.4; in future a more complex model may be called

"Okay… now I will describe the two green objects. One of them is this and one of them is this. And… this one is the same object has the same shape than this. And this is the same in green like this. And the form should be just known."

"This is a… the color is green. And it's a cuboid. It looks like a small parcel or a stone. And this is also green. It looks like a bridge. So it's the same color like the cuboid."

Figure 7.6: An image taken during data collection with examples of language used to describe the scene. An association is learned between words such as "cuboid" and appropriate classifiers.

for.) Each word $w \in W$ is a boolean feature $l_w \in \{1, 0\}$, whose value is its presence or absence in $L$; these features are analogous to the unordered codewords used for visual analysis. The scene description $L$ can then be represented as a vector of these features. These word features are then combined with the output of the visual classifiers for each object (in practice, this is performed by training the language model using logistic regression with a polynomial kernel), to produce features for the pairwise co-occurrence of words and attributes:

$$\begin{aligned} \gamma_o &= \left[ l_{w1} \times v_{o,i1}, \ldots, l_{w|W|} \times v_{o,i|\{C,S\}|} \right], \\ w &\in W, i \in \{C, S\} \end{aligned} \tag{7.8}$$

Intuitively, this means that features encoding "good" correspondences will have high values when found with positive examples in the training data (e.g., $\langle l_{\text{"}bridge\text{"}} \times \ v_{o,arch} \rangle$ in Figure 7.5), and can be weighted up accordingly.

The language $L$ from each scene is taken as a positive exemplar for attributes of

all positive objects. This introduces noise into the training data (for example, the first description in Figure 7.6 might cause the classifier to learn a low value for the feature $\langle l_{\text{“}cuboid\text{”}} \times v_{o,cube} \rangle$, since there are no words describing shape), but avoids hand-labeling of the spoken language, in line with the long-term goal of having users train a robot without expert assistance.

### *7.3.5 Integration*

Given a combination of language, visual attributes, and gesture, the object selection task is to automatically map a natural language scene description $L$, a (possible) gesture $h$, and a set of scene objects to the subset of objects $G$ indicated by $L$ and $h$, that is, $P(G \mid L, h)$. Rather than considering object subsets explicitly, language and gesture classification are factored directly into individual classifications over objects.

As described above, $r_o$ and $h_o$ are output by two classifiers: a logistic regression classifier that takes as input the language used and the detected visual attributes of object $o$, and outputs an estimate of the probability that $o$ is being *referred to* in language (Section 7.3.4); and a logistic regression classifier that takes the features described in Section 7.3.2 and outputs an estimate of the probability that object $o$ is being *gestured to*. These values must then be combined to provide an interpretation of the user's intent.

A user may choose to use only gesture or only language; to avoid penalizing single-mode interactions, if either the language or gesture signal is below some threshold $\epsilon$, a small minimum value is used, yielding the modified, positive language and gesture scores of $r'_o$ and $h'_o$. Experimentally, $\epsilon = 0.2$ worked well. (Because gesture is used more inconsistently than language, using different minimum values for $h_o$ and $r_o$ might be appropriate in future, interpretable as a prior over whether an arbitrary object is being gestured to/referred to.) $r'_o$ and $h'_o$ are then summed and normalized across the

data set to produce $k_o$, an unweighted 0-1 score used to determine whether the object $o$ is being indicated.

A different approach to integration is to simply use outputs of individual components as inputs to a linear SVM trained with LIBLinear, which gives a final probability, for each object, that that object is being referred to (see Figure 7.7). This approach works equally well, achieving an F1 score of 82%.

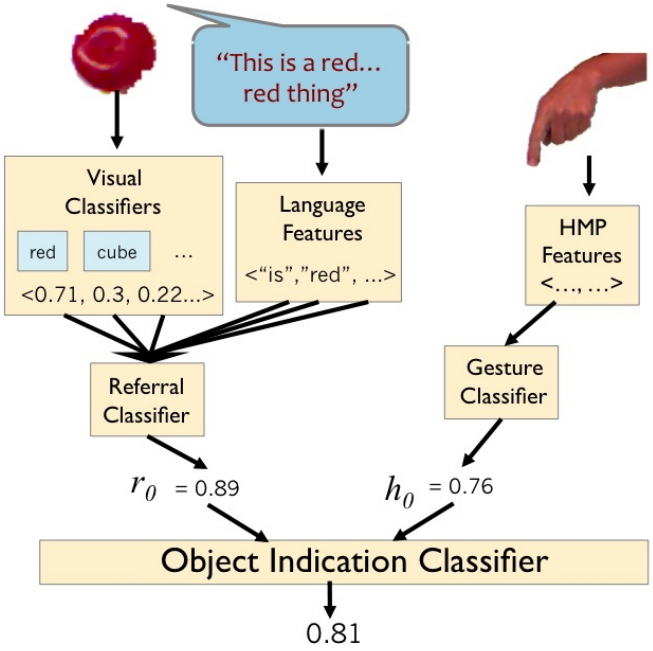



Figure 7.7: The integration architecture. $r_o$ represents the output of the referral classifier (*i.e.*, whether an object is being referred to in language, Section 7.3.4), while $h_o$ is output of the gestured-to classifier (Section 7.3.2).

## *7.4 Experimental Evaluation*

The corpus consists of data collected with 13 participants, each describing 28 scenes, yielding 364 language/video pairs. Testing was performed on a held-out set of 20% of these pairs, containing a total of 520 objects. Additional testing was performed

by asking 5 new participants to each instruct the robot to perform tasks on 5 scenes (Section 7.4.5). In these trials, we use the participants' speech and gesture to label objects, and no additional training of the robot is performed.

### *7.4.1 Per-Object Accuracy*

In this evaluation, objects are evaluated independently. To determine whether an object is being indicated, a cutoff is applied to the object score $k_o$, gesture score $h'_o$, or language+attribute score $r_o$; an object is positive (indicated) if its score is above the cutoff. This allows us to produce a precision/recall curve for the task (Figure 7.8). As expected, the combination of gesture and language outperforms either.

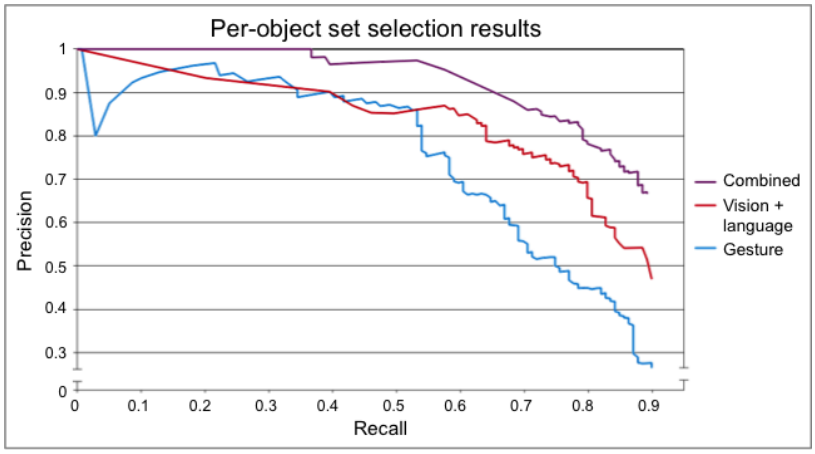



Figure 7.8: Precision (y-axis) and recall (x-axis), obtained from varying the cutoff above which object $o$ is selected. The cutoff is applied to $r'_o$ (for language+vision, red), $h'_o$ (gesture only, blue), or $k_o$, which combines language and gesture (purple).

The system achieves a peak F1 score of 80.7%. As the cutoff increases, the system increasingly returns only correct objects, but misses more objects as well; depending

on the context, different trade-offs of such false positives vs. false negatives might be appropriate.

### *7.4.2 Scene-Based Accuracy*

In *scene-based* accuracy analysis, object scores are again used to determine whether objects are indicated, but a trial is treated as a success only if all objects in the scene are correctly classified as positive or negative. This metric is strictly more difficult than the per-object analysis. At a naïve score cutoff of 0.5, the system performs perfectly on 53.9% of scenes; when the cutoff is tuned to best performance, which can be done using training data, the success rate rises to 57.9%. (For comparison, if the robot randomly selected some number of blocks, success would average 2.2%.)

### *7.4.3 Evaluation of Feasibility*

Human evaluators were asked to perform the same task in order to test how well the collected corpus supports the task. Evaluators were asked to decide what objects are being indicated from a (silent) video, the transcribed language, or both. Each scene was shown to three evaluators; a successful trial is one in which *all* objects in a scene were identified correctly (scene-based). In cases of disagreement, consensus majority-vote agreement (2 of 3) was also calculated, and success of the consensus opinion is reported.

Results are shown in Table 7.9. These results are consistent with the intuition that the combination of gesture and language is better for object selection than either in isolation. In addition, even the best case (consensus evaluation with both modalities), the success rate of human evaluators was approximately 93% (bold), providing an upper bound on probable system performance.

| | | Inter-annotator agreement | Success Rate |
|---|---|---|---|
| **Individual** | Vision+Language | 0.799 | 0.866 |
| | Gesture Only | 0.780 | 0.803 |
| | Given Both | 0.747 | 0.873 |
| **Consensus** | Vision+Language | 0.961 | 0.888 |
| | Gesture Only | 0.964 | 0.830 |
| | Given Both | 0.967 | **0.926** |

Figure 7.9: A trial is a "success" when *all* objects in a scene are labeled accurately. "Inter-annotator agreement" is percentage of trials where at least 2 testers agreed; the "consensus" case reflects the correctness of that consensus opinion. (top) Accuracy and inter-evaluator agreement across all evaluators; (bottom) accuracy of majority-voting results. When all three evaluators disagree, there is no consensus result reported.

### *7.4.4 Gesture Classification and Novel Features*

Different approaches for determining which objects a person is gesturing to were evaluated. As a baseline, a simple closest-approach and closest leading-edge approach (as described in Section 7.3.1) is considered, with consider any object below a certain distance threshold to be "indicated".

For a slightly less naïve baseline, a logistic regression classifier was trained over the minimum of the 4D leading-edge distances, computed over an entire interaction, for each object. This leading-edge distance performs better than simple distance-to-object; this makes sense from a geometric standpoint, as when someone points to a block on the far edge of an arrangement, some part of the hand may be quite close to blocks nearer the speaker (see Figure 7.10).

The novel hierarchical matching pursuit features described in this chapter were compared against these baselines, using a logistic regression classifier. Figure 7.11 shows

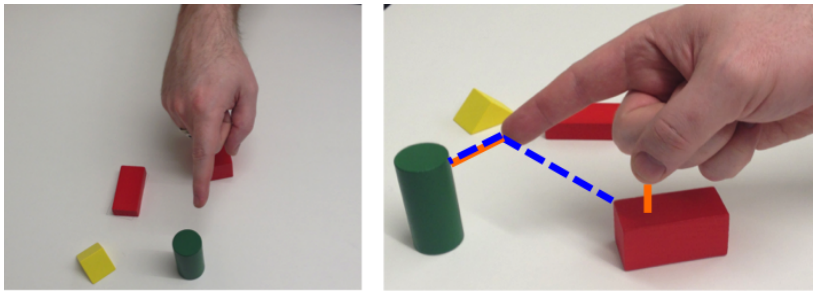


Figure 7.10: Orange lines show closest-approach distance to each object, while blue lines give leading-edge distance; the green cylinder is the target.

precision/recall for all approaches.

The novel high-dimensional hierarchical matching pursuit features provide significantly better results than the simple leading-edge vector. The difference is particularly striking in areas with high recall. For instance, this approach achieves 91% and 65% precision at 80% and 90% recall, whereas logistic regression over the closest approach only reaches 63% and 34% precision for the same recall values, respectively. This is due to the fact that people tend to move their hands in complex patterns, often moving closely over objects they do not want to indicate. While minimal distance is not able to capture such complex relationships, the HMP features developed are rich enough to provide good classification results.

### *7.4.5 Evaluation of Prototype System*

A prototype of a system that makes use of these classifiers was implemented on the Gambit manipulator platform [Matuszek et al. 2012a], using the models of language and gesture described above to identify and interact with objects being indicated by a user. Automatic speech recognition is performed using the Google Speech API. In an initial, small evaluation of the system, five new participants were asked to indicate

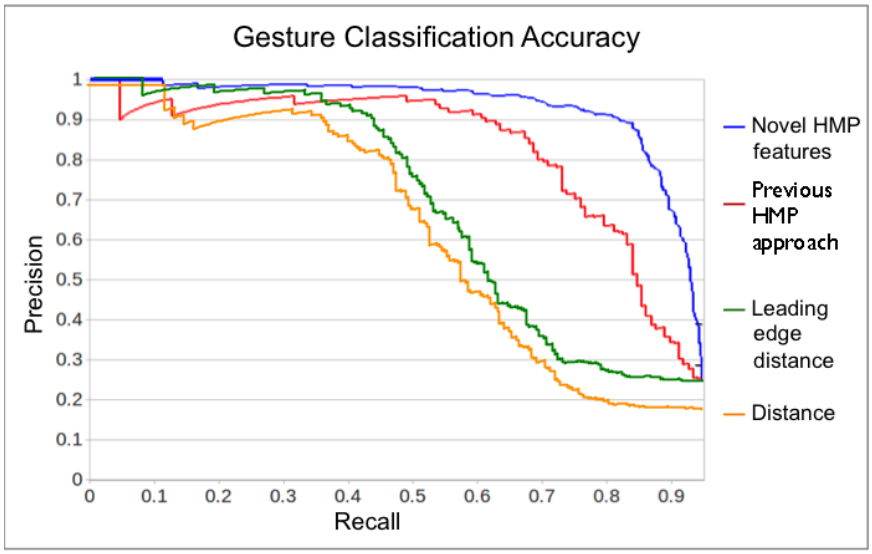



Figure 7.11: Precision (y-axis) and recall (x-axis) showing performance of different baselines. Yellow and green use the closest-approach distance and closest leading-edge distance, respectively. The red line shows the performance of a logistic regression classifier trained using traditional sparse codewords learned from the collection of four-dimensional vectors from the series, and the blue line uses the novel binary-value sparse codes presented in Section 7.3.2.

objects they would like the robot to put away in a bin. Even though many elements of the system (camera placement, specific objects, user) are different from the training data, the system does reasonably well. Per-object accuracy of the results is reported in Table 7.12.

The overall precision and recall of the combined system are 79% and 90%, overall consistent with results from evaluation of the individual classification models. Results vary substantially by participant, reinforcing the belief that additional training of a robot by individual users may be beneficial. In this trial, the combination of modalities is competitive, but not a clear improvement as a result of the weak language signal.

| | Lang+attr | | | Gesture | | | Combined | | |
|---|---|---|---|---|---|---|---|---|---|
| **user** | ***P*** | ***R*** | ***F*** | ***P*** | ***R*** | ***F*** | ***P*** | ***R*** | ***F*** |
| 1 | 0.91 | 0.71 | 0.8 | 0.91 | 0.83 | 0.87 | 0.83 | 1.0 | 0.91 |
| 2 | 0.91 | 0.91 | 0.91 | 1.0 | 0.91 | 0.95 | 0.91 | 1.0 | 0.91 |
| 3 | 0.83 | 0.71 | 0.77 | 0.91 | 0.77 | 0.83 | 0.71 | 1.0 | 0.83 |
| 4 | 0.71 | 0.91 | 0.8 | 0.91 | 0.91 | 0.91 | 0.67 | 0.91 | 0.77 |
| 5 | 0.77 | 0.59 | 0.67 | 1.0 | 0.5 | 0.67 | 0.71 | 0.59 | 0.64 |
| *mean* | 0.83 | 0.77 | 0.8 | 0.95 | 0.75 | 0.84 | 0.79 | 0.9 | 0.84 |

Figure 7.12: Performance for each participant across 5 scenes, containing 22 objects total. Precision and recall are reported for each source of instruction and for the combined results.

At a high level, the a prototype system demonstrates the feasibility of applying the learned models to a system which performs simple tasks. In addition, identifying the target of unconstrained gesture is evaluated, demonstrating that the novel temporal HMP features developed in Section 7.3.2 can handle this complex problem.

The largest sources of errors in language interpretation were failures of voice recognition (46%), followed by overfitting to the training data resulting in poor shape classification (25%). The largest source of error in gesture (43%) was gestures made too far back from the robot, outside the expected workspace.

These failures can be partially addressed by providing more transparency to the user regarding the robot's perceptions and state. However, while reflecting speech recognition results is relatively easy, it is not sufficient to naïvely show the output of the robot's camera. This approach and its associated interfaces are the subject of the next chapter.

## *7.5 Discussion*

In this chapter, language acquisition is addressed in an unambiguously grounded context. A corpus of people using unconstrained gesture and speech is used as the basis for learning the association of language with visual color and shape attributes, as perceived through a robot's sensors. Deictic gestures are interpreted using novel features that perform well for determining whether a participant is indicating objects, and the combination of speech and gesture understanding is tested on a prototype system. In future, using the learned model of gesture to provide world context for language and vision—that is, using the output $h'_0$ as the training input $G$ in the learning approach described in Chapter 6—will lead to a more automated, less supervised learning system.

The failure modes encountered speak to the difficulty of moving to a real, fully grounded system, with all the uncertainty inherent in the noisy physical world. While better sensors, more sophisticated language models, and a more comprehensive gesture model would all improve the knowledge acquisition to some extent, another element of the system exists which has been considered only minimally so far: the interaction with the person teaching the system.

The next chapter describes a system implemented on an additional robot platform, the Willow Garage PR2. The focus of that work is squarely on evaluating and improving the human-robot interaction element of the overall system.

# Chapter 8

# INTERFACES SUPPORTING HUMAN-ROBOT TEACHING INTERACTIONS

This dissertation has thus far focused on the learning and classification problems associated with grounded language acquisition. The motivation for this research is the need for intuitive, natural interfaces for teaching robots about tasks and objects in the world. The previous chapter described work on a robot that understands unscripted directives from end-users and performs operations based on that understanding. It learns, particularly, to ground language in visual percepts with the aid of gesture—a powerful step towards the task of building robots that users can teach about the world.

That understanding relied on an RGB-D corpus of teaching interactions from users. While this corpus was effective at training language and gesture models, the process of collecting it was time-consuming, and the need to annotate the collected data is inconsistent with the goal of learning entirely from end users. Evaluation of the collected data and the behavior of participants during testing also suggest that users would benefit from interactions with the robot that are more transparent, focused, and understandable.

It should be noted that interactions with a robot that are that are comfortable and intuitive do not necessarily mimic how humans interact with one another. Natural

---

The work described in this chapter was undertaken with substantial advice from Matthew Kay and Maya Cakmak, to whom I am grateful for the advice and the chance to play with Rosie.

language feedback provided by a robot may not be "natural" in the sense of resembling human interlocution, while still allowing the human user to provide direction in an unscripted, human-like way. The goal is then to develop a robot that behaves in a way users find easy to understand and adapt to, with no guidance from an experimenter (who would not be present in an eventual deployment context).

This final chapter focuses directly on the human side of the human-robot interaction, looking at the question of how to make the process of teaching a robot easier. While preliminary, the question of appropriate feedback in interactive/active learning and teaching will have a significant effect on the ultimate acceptance of a robot that learns from end users. The system presented is an initial implementation of various forms of feedback on the robot's side, developed and implemented on a PR2 robot as part of a feasibility study designed to examine what kinds of feedback a human user will find easy to use, helpful, and comfortable.

## *8.1 Motivation and Overview*

The previous chapter demonstrated progress towards a robot that can learn to understand unscripted directives from end-users and perform operations based on that understanding. That system learns to interpret user intent in a physical workspace, and particularly, learns to understand the subset of deictic interactions called directing-to, which are attention-focusing: language and gesture in which the user is drawing attention to specific objects. This learning was based on a corpus of 364 teaching interactions performed by volunteers in a physical workspace.

While this corpus was effective at training language and gesture models, the data collection itself was not without drawbacks. Participants in the study found the task (performing 28 teaching interactions, recorded and guided by a human experimenter) both too long, and, anecdotally, distasteful. This anecdotal data is backed up by

analysis of known signals of verbal discomfort, e.g., increased use of filler words, longer pauses between words, and frequency of conversational back-off/restarts. This user discomfort is in direct opposition to the goal of creating a robot with which a non-specialist can naturally and easily interact.

The focus of this chapter is on the results of implementing a first iteration of an interface intended to explore ways of making the same teaching interactions more pleasant to engage in, specifically by making the robot's internal state more transparent to the user (Figure 8.1). Results drawn from the post-study questionnaires are primarily either on a five-point Likert scale, or free responses to fairly general questions. For the latter, qualitative analysis was performed, looking for common answer themes and similar statements among responses from different users; these results are largely presented as representative quotes.

The results of a feasibility study are described. The goals of the study are: (1) to answer the question of whether interfaces of this sort are useful and appropriate for the task of teaching a robot; and (2) to gain insight into the quality and usability of the specific interfaces implemented. The interface elements presented here do not represent a complete solution; rather, asking the participants questions about their experience yields lessons and suggestions for continuing iteration on such interfaces.

## *8.2 Approach: User Study Design and Experimental Interfaces*

Choosing interface elements to implement initially was driven by attempts to identify elements that might have contributed to user discomfort during data collection. A pilot training session was performed with a human-computer interaction expert, who offered specific analyses of possible weaknesses. Possible sources of discomfort identified, and the interface elements intended to address them, included:

Figure 8.1: A data collection participant indicates objects to the PR2 robot "Rosie2." The experimenter places an iPad with target objects circled in order to avoid linguistic or gestural priming. Part of the experimental user interface is shown (top right of image).

- **Not having a clear idea at what level of granularity to supply information.** Proposed solutions:
  - Make the instructions substantially clearer as to the intent of the task.
  - Provide an interface that allows the users to see the view from the relevant robot sensors (a head-mounted Kinect), allowing them to constrain detail by the useful resolution of robot vision.
- **Talking "into a vacuum" with no responses,** which is not conversationally natural. Proposed solution:
  - Design the interaction such that the robot (a PR2 from Willow Garage, now maintained by ClearPath) makes conversational "affirmative sounds" at appropriate points.
- **Not being clear on whether the robot "understood" the teaching,** e.g., whether the participant is speaking at the correct level, clearly, etc. Proposed

solution:

- Add an interface that provides transparency of speech recognition, such that the user has a clear idea what the robot is "hearing."

The design of the user study is presented, followed by details of the components described above.

### *8.2.1 Experiment Design*

To evaluate the overall impact of the robot's feedback, a within-participants study was performed, allowing the participants to compare using the interfaces to the previous data collection situation. In the first case, the participant has no interfaces, and is performing a task identical to that described in Section 7.2.1. This is the "No Feedback" case. In the second, the interfaces described above are available; this is the "Interface" case. The order in which these cases are presented was randomized. After a participant performed ten teaching interactions in each case, they filled out a questionnaire about the teaching experience; at the end of the study, they filled out an additional questionnaire comparing the two different cases.

The teaching task is identical to that described in the previous chapter, in which participants are asked to distinguish objects from a scene, using language and gesture. The experimenter specified what objects to indicate (see Figure 8.1) in both training and testing. However, rather than being asked merely to "distinguish objects," examples were provided as part of the instructions, as shown in Figure 8.2.

In both cases, the participants were seated in front of the robot, rather than in front an unnaturally non-responsive human experimenter. Scenes were designed to collect data for objects with different spatial relations and combinations of attributes.

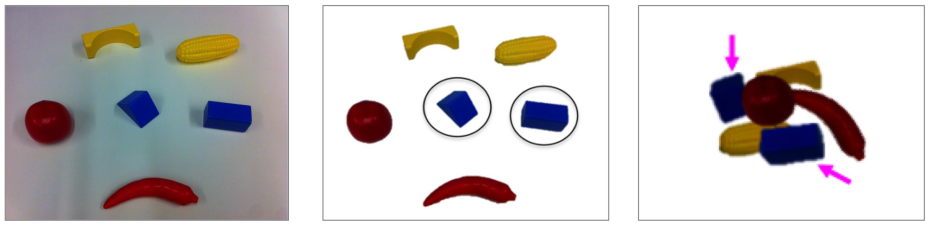


Figure 8.2: Excerpts from the experimental instructions given to participants. (left) The objects in front of the participant, and (middle) an iPad screen, showing the objects circled that are to be described. The robot should learn enough to select those objects from a different arrangement (right).

### *8.2.2 Visual Feedback*

The interface for providing transparency into the limits of the robot's vision was a simple real-time visual stream of data from the PR2's Kinect sensor (see Figure 8.3). This component was intended to allow the participant to be more concise by limiting their descriptions to physical detail appropriate to the camera's resolution, rather than feeling compelled to describe additional fine detail which the classifiers cannot see or make use of.

### *8.2.3 Speech Recognition*

In the previous chapter, the language used by participants was transcribed by human annotators. In the interest of moving towards a system that can be used for teaching a robot without experimenter intervention, this experiment instead used automatic speech recognition, performed using Google Speech web recognition. The web API provides a continuous update for speech recognition; as the participant continues speaking, the recognized language is updated to reflect changing analyses, which are dependent on context.

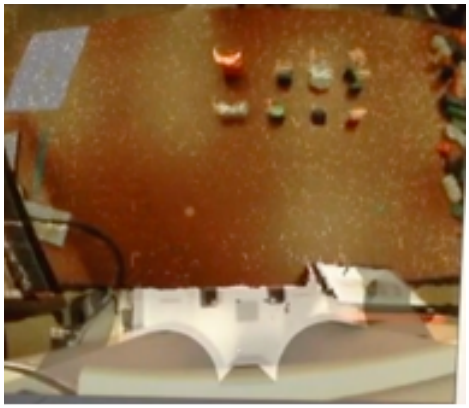


Figure 8.3: A frame from a scene as viewed through the PR2's Kinect camera, demonstrating the generally rough RGB-D resolution.

Participants were asked to wear a head-mounted microphone, which makes speech recognition substantially more accurate, and were shown a monitor on which the continuing recognition was displayed (see Figure 8.4).

Human hearing and speech recognition make it possible for a teacher in a two-sided interaction to generally assume the listener is capable of hearing and disambiguating spoken instructions. This interface does not attempt to mimic human conversational interactions; instead, the user is provided with feedback designed to allow him or her to know how well or poorly the robot understands spoken descriptions and instructions, somewhat mitigating the challenges presented by modern speech recognition.

### *8.2.4 Spoken Responses*

While automatic speech recognition is continuous, a final interpretation is output when there is a long enough pause to conclude that the speaker has finished speaking a sentence, thought, etc. This capability is used to control the timing of conversational affirmative noises (such as "Go on," "Uh huh," "Okay," and so on), to give the illusion


Figure 8.4: A frame from a scene showing the progression of real-time automatic speech recognition. In this case, while the language is not captured perfectly, the most important words are recognized. The original sentence was "These are two cubes, a blue one and a green one."

that the robot is an active participant in the learning process. After the speaker has finished a sentence and paused for long enough to trigger a final speech analysis, a randomly chosen affirmative is spoken aloud by the robot.

## *8.3 Results*

The intent of this study was to establish the quality of the different interfaces implemented, and to answer the question of whether interfaces of this sort are useful and appropriate for this teaching task. Results drawn from the post-study questionnaires are primarily either on a five-point Likert scale, or free responses to fairly general questions. For the latter, qualitative analysis was performed, looking for common answer themes and similar statements among responses from different users; these results are largely presented as representative quotes.

While the results presented in this chapter are preliminary, and additional analysis will

be performed when more participants have completed the experiment, the responses obtained so far provide insight into appropriate interface changes and the relative value of the components.

### *8.3.1 General Reactions*

One of the primary questions of this study was how users felt about teaching the robot with some form of feedback. A poorly designed interface might, in fact, be worse than none; it is also possible that a lack of transparency does not address the issues of the teaching task, making the implemented interfaces irrelevant. Accordingly, one of the most informative results is a comparison of the No Feedback case vs. the Interface case, as shown in Figure 8.5.

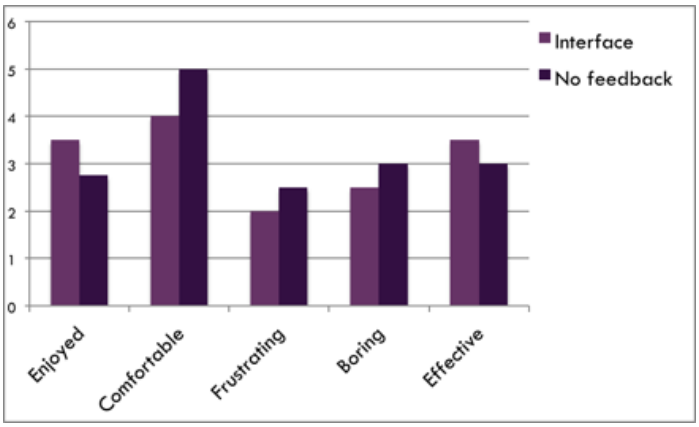



Figure 8.5: Average responses on a five-point scale to questions of whether the participant found the teaching interaction enjoyable, boring, and so on, as compared between the two cases.

As hoped, participants found that the presence of an interface made the teaching task more enjoyable, less frustrating, and less boring. In general, they were more engaged in the task. In a separate question, asking explicitly which case was preferred,

responses ranged from "No preference" to "Prefer the interface," with no participants explicitly preferring the case in which no feedback is present.

Participants also rated whether they thought the teaching was "effective," as well as answering the questions, *"After teaching the robot, what sorts of instructions would you expect it to be able to understand? What tasks should it now be able to do?"* Although the participants thought teaching with the interface was slightly more effective, answers to the question of expected capability were essentially the same:

> "I still think it can only learn about shapes and colors based on our interaction."

> "It should be able to interact with the objects I specify."

One interesting outcome is shown in the second cluster in Figure 8.5: participants found the process of teaching the robot *less* comfortable in the Interface case. This is likely a product of the fact that an increase in transparency exposed the weakness of the learning signal:

> "Without the language I felt I went faster but probably the robot performed worse. With the teacher I went slower and tried to correct errors, but also got more annoyed."

> "I found the teaching [with an interface] frustrating in some ways and less so in others. It was nice when it figured out what I was saying and annoying when it didn't, and I just gave up after a few failed tries."

### *8.3.2 Reactions to Interface Components*

Participants were also asked questions designed to gauge their evaluation of the specific components of the interface. In qualitative data, the spoken responses were considered a positive element in theory, but poor dialogue timing caused them to be considered frustrating; the speech recognition feedback was considered the most useful; and the interface conveying the robot's visual capabilities was generally misunderstood and unused.

#### *Reactions to Conversational Affirmatives*

The overall reaction to the spoken affirmatives was one of frustration. This was primarily a matter of dialog timing. The robot frequently interrupted the teacher:

> "The spoken feedback was annoying because Rosie interrupted me several times..."

> "The experience was frustrating because the robot interrupted me a lot, which is not normal."

There are two factors affecting the timing. One, both speech recognition and generating spoken responses took some time, so unless the participant waited between sentences, the robot often started speaking after the pause was over. With the correct infrastructure, this could be corrected. However, dialog timing is a more difficult problem; timing dependent on speech recognition does not match natural dialogue.

Users would also prefer real-time feedback about whether the robot was learning from the conversation, rather than general listening noises:

> "I couldn't understand if the robot really understood."

> "I would like the robot to give me indications that its [sic] understanding what I mean."

Because information given to the robot is used to continually train and augment existing language and gesture models, it is difficult to actually say whether any given interaction is "understood." However, some feedback as to the effectiveness of the teaching clearly should be devised, possibly as part of followup work in which the robot responds to teaching statements with appropriate questions.

*Reactions to Speech Recognition*

Despite its limitations, speech recognition was generally considered the most useful interface component. Users expressed some frustration with the generally poor quality of the recognition:

> "It made me try to make sure it got the language right, but I also got more frustrated when it wasn't able to get the language after just a few tries"

However, this was the interface marked most useful and least in need of improvement. This is an interesting result, given that speech recognition was the largest source of error in previous work, and in this work often resulted in the loss of key attribute words. Users evidently found feedback on when things were going wrong significantly useful, as well as frustrating.

*Reactions to Visual Feedback*

Almost universally, users failed to parse this feedback as information about the robot's limited visual capabilities. The display was generally considered the least useful:

> "I mostly focused on the language display and ignored the visual display."

> "The visual feedback was also not that necessary because I can see the table and the objects. I saw in the beginning that she was looking in the right place..."

This component of the interface needs to be redesigned to be useful. Possible changes might be: have the visuals change occasionally (to attract attention), or, more radically, to limit the users' view of the objects to the Kinect feed. Providing more instruction on the purpose of various components—that is, instructing users to consider the robot's visual limitations as shown by the camera—might also, in itself, be a substantial improvement. That said, this problem can only become less relevant as RGB-D hardware improves.

### *8.3.3 Comparisons*

In the comparative analysis, users' preference ranged from neutral to strongly preferring the Interface case, despite the limitations described above. Users who were shown the No Feedback case first expressed a desire for feedback, and one user complained when it was removed in the second case. When asked explicitly which elements should be considered for inclusion in future interfaces, only the visual interface was considered extraneous by some users (Figure 8.6).

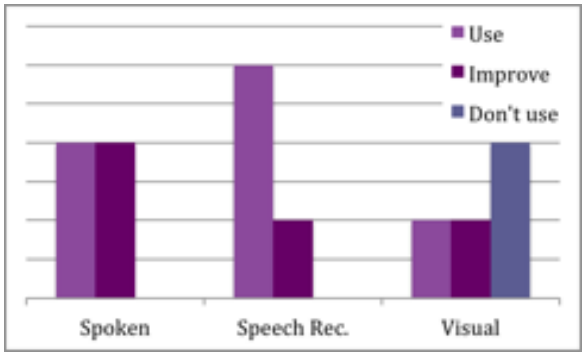



Figure 8.6: Responses to the question *"Which of the following interface elements would you suggest using in future teaching tasks?"* Possible answers were Use, Don't Use, and Use With Improvements.

In order to understand whether this work makes the teaching task easier to perform, the questionnaire additionally asked for a qualitative estimate of the number of additional teaching interactions that each participant would perform in order to teach the robot to perform a useful task: *"You engaged in ten teaching tasks in each case. If you imagine teaching the robot about something that was useful for you (for example, you want it to pull blue objects off an assembly line), how many additional teaching interactions would you be willing to engage in?"* The most common response for the Interface case was "Several More," as opposed to "A Few More" in the No Feedback case. This suggests that having an interface, even a discomfiting one, does in fact make users more willing to continue teaching the robot.

## *8.4 Discussion*

Based on data collected so far, the interfaces designed for this study were considered useful, and overall satisfaction increased when they are used. As expected, providing interfaces that give the human teacher feedback and provide transparency is beneficial

to the human/robot teaching scenario. The speech recognition interface also led to slightly improved training data.

This work points the way towards providing users with the necessary tools to make providing a robot with information painless and effective. However, it is also clear that the interfaces presented are only a first iteration. The specific design choices discussed in this chapter provide a jumping-off point for the more general study of improving the teaching experience.

# Chapter 9

# CONCLUSION

This dissertation describes a system for using a synthesis of natural language processing and real-world sensing to ground language—that is, to connect words and symbols to their referents in a non-symbolic context. The contexts used range from simulated maps, to vision provided by an RGB-D camera, to human gesture, but the theme throughout remains the same: learning the meaning of language by tying it to the world, and learning to interpret the rich and uncertain world guided by language.

## *9.1 Future Work*

Grounded language acquisition is a challenging problem with many real-world applications. Much of this work is motivated by the need for robots that operate in human environments and interact well with people, and this thesis addresses aspects of that problem and provides a base on which to continue to develop systems that meet those needs. Key areas for further work include better human-robot teaching interactions, improved incorporation of context, comprehension of more complex language, and the continuing development of end-to-end systems in which to test the effectiveness and usefulness of the techniques being developed.

The biggest challenge that lies ahead is extending beyond descriptive language to more general vocabulary and higher-level representations of meanings, including more abstract and intent-driven language interpretations. This will include adding richer representations of contextual world information, including models of the people in the

environment, as well as continuing to develop improved models for joint learning and improved interfaces to support efficient, natural interaction. To that end, this work will support further research in several key areas.

*Human-Robot Dialog for Active Learning*

To go beyond learning passively by listening to a human teacher, it is necessary for a robot to ask questions or otherwise help direct its own learning; such active learning allows for more efficient learning and better performance as well as more natural interaction. Using active learning in human-robot interactions presents its own challenges, raising issues not only of what questions to ask, but when and how to ask them: It is necessary to balance the improvement in the robot's acquisition of concepts with minimizing demands on users. The goal is then to incorporate only as much questioning as is useful to drive efficient understanding of human utterances.

Much of the work on active learning in robotics settings revolves around dialog systems, and, in particular, around an architecture in which a robot has a goal or goals that are integrated into a central architecture that may include other goals such as accomplishing tasks. Exploring the utility of making a robot that uses language acquisition to be a more active participant in its own learning is an obvious next step for this area of research. This includes taking advantage of the robot's ability to incorporate nonverbal elements of queries (such as pointing to an object while asking about it, or using tone to convey uncertainty), leading to more transparent and therefore hopefully more natural interactions.

*Additional Learning Mechanisms to Supplement Human Teaching*

Robots should be able to learn new concepts from their users, but that mechanism can and should should be supplemental to other sources of information gathering. This includes automatically extending formal representations of the world in order to help robots gather information about new concepts, especially in circumstances where

the transaction cost of asking a user for help is too high, or simply unnecessary. (For example, if a user asks a robot to retrieve an unfamiliar object such as a Coke can from another room, an online image search provides enough information to attempt a solution without asking for additional help.) This information still needs to be tied into a linguistic framework.

Crowdsourcing, in particular, has the potential to address a number of otherwise cumbersome problems. Large datasets can be gathered in order to provide a solid basis for language learning, and more focused uses will allow deployed robots to solve specific problems. Pre-emptive information gathering, such as asking for labels when an unrecognized object is in view, can be continuously conducted; higher-cost, real-time crowdsourcing can help with time-critical problems such as finding an elevator button in a camera view. The addition of these mechanisms to existing systems will allow deployed robots to continuously improve performance without constantly requesting user assistance.

*Assistive Technology*

One of the most immediate and crucial areas in which robots can be deployed productively is in the developing areas of assistive technology and elder care. Systems that can supplement human capabilities—such as providing descriptions of the environment to visually impaired persons or performing shared manipulations with a user with limited grip strength—have the potential to be useful as soon as user interfaces and control mechanisms catch up to the physical abilities of existing technology. Robots that increase autonomy for populations such as aging-in-place elders have the potential to have broad impact, but also require careful user studies and sensitivity to the costs and benefits of the systems being developed. As a driving application, this is an area of future work that stands to benefit enormously from application and further development of the technologies described in this dissertation.

## *9.2 In Conclusion*

This dissertation describes significant contributions to the long-standing question of how to ground language into physically situated systems. The work described herein uses and improves on state of the art approaches to semantically parsing language and interpreting visual signals, using them jointly to learn a shared language and world model; the result is robust, effective systems capable of following directions, identifying and learning about objects in the world, and responding to multimodal human instructions.

The primary contribution of this work is a solid base for continuing research on bringing together natural language, robotics, and human-robot interaction, with the long-term goal of producing systems that are easy to teach, intuitive to use, and useful in a human-centric world. I end this document where I began, by quoting the final line of *Computing Machinery and Intelligence:* "We can only see a short distance ahead, but we can see plenty there that needs to be done."

# BIBLIOGRAPHY

A. V. Aho and J. D. Ullman. *The Theory of Parsing, Translation, and Compiling*. Prentice Hall Professional Technical Reference, 1972. ISBN 0139145567.

M. D. Anderson, S. Chernova, Z. Dodds, A. L. Thomaz, and D. S. Touretzky. Report on the AAAI 2010 Robot Exhibits. *AAAI Magazine, in press*, 2010.

Y. Artzi and L. Zettlemoyer. Bootstrapping semantic parsers from conversations. In *Proc. of the Conf. on Empirical Methods in Natural Language Processing*, 2011.

Y. Artzi and L. Zettlemoyer. Weakly supervised learning of semantic parsers for mapping instructions to actions. *Transactions of the Association for Computational Linguistics*, 1(1):49–62, 2013.

J. K. Baker. Trainable grammars for speech recognition. *The Journal of the Acoustical Society of America*, 65(S1):S132–S132, 1979.

K. Barnard, P. Duygulu, D. Forsyth, N. De Freitas, D. Blei, and M. Jordan. Matching words and pictures. *The Journal of Machine Learning Research*, 3:1107–1135, 2003.

M. Beetz, T. Arbuckle, T. Belker, M. Bennewitz, W. Burgard, A. Cremers, D. Fox, H. Grosskreutz, D. Hähnel, and D. Schulz. Integrated plan-based control of autonomous service robots in human environments. *IEEE Intelligent Systems*, 16(5), 2001.

I. Beltagy, K. Erk, and R. Mooney. Semantic parsing using distributional semantics and probabilistic logic. In *Proceedings of ACL 2014 Workshop on Semantic Parsing (SP-2014)*, pages 7–11, Baltimore, MD, June 2014.

J. Berant and P. Liang. Semantic parsing via paraphrasing. In *Association for Computational Linguistics (ACL)*, 2014.

J. Berant, A. Chou, R. Frostig, and P. Liang. Semantic parsing on freebase from question-answer pairs. In *Proceedings of EMNLP*, 2013.

C. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer, 2007.

L. Bo and C. Sminchisescu. Efficient Match Kernel between Sets of Features for Visual Recognition. In *Neural Information Processing System*, 2009a.

L. Bo and C. Sminchisescu. Efficient Match Kernel between Sets of Features for Visual Recognition. In *Advances in Neural Information Processing Systems (NIPS)*, December 2009b.

L. Bo, X. Ren, and D. Fox. Kernel descriptors for visual recognition. In *Neural Information Processing Systems (NIPS)*, 2010.

L. Bo, X. Ren, and D. Fox. Depth kernel descriptors for object recognition. In *Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2011a.

L. Bo, X. Ren, and D. Fox. Hierarchical Matching Pursuit for image classification: architecture and fast algorithms. In *Advances in Neural Information Processing Systems*, December 2011b.

L. Bo, X. Ren, and D. Fox. Unsupervised feature learning for RGB-D based object recognition. *International Symposium on Experimental Robotics (ISER), June*, 2012.

R. A. Bolt. *"Put-that-there": Voice and gesture at the graphics interface*, volume 14. ACM, 1980.

C. Boutilier, R. Reiter, M. Soutchanski, and S. Thrun. Decision-theoretic, high-level agent programming in the situation calculus. In *Proc. of the National Conference on Artificial Intelligence (AAAI)*, 2000.

S. Branavan, L. Zettlemoyer, and R. Barzilay. Reading between the lines: Learning to map high-level instructions to commands. In *ACL*, pages 1268–1277, 2010.

A. Bratersky. Dvorkovich, Chess Robot Go 1-1. `http://www.themoscowtimes.com/news/article/dvorkovich-chess-robot-go-1-1/408089.html`, June 2010.

C. Breazeal and B. Scassellati. Infant-like Social Interactions between a Robot and a Human Caregiver. *Adaptive Behavior*, pages 49–74, 2000. doi: 10.1177/105971230000800104.

C. Breazeal, A. Brooks, D. Chilongo, J. Gray, G. Hoffman, C. Kidd, H. Lee, J. Lieberman, and A. Lockerd. Working Collaboratively with Humanoid Robots. In *Proceedings of 4th IEEE/RAS International Conference of Humanoid Robots*, 2004.

P. F. Brown, J. Cocke, S. A. D. Pietra, V. J. D. Pietra, F. Jelinek, J. D. Lafferty, R. L. Mercer, and P. S. Roossin. A statistical approach to machine translation. *Comput. Linguist.*, 16(2):79–85, 1990. ISSN 0891-2017.

W. Burgard, A. B. Cremers, D. Fox, D. Hähnel, G. Lakemeyer, D. Schulz, W. Steiner, and S. Thrun. Experiences with an interactive museum tour-guide robot. *Artificial Intelligence*, 114(1-2):3–55, 1999.

Q. Cai and A. Yates. Semantic parsing freebase: Towards open-domain semantic parsing. *Atlanta, Georgia, USA*, 30:328, 2013.

M. Cakmak, C. Chao, and A. L. Thomaz. Designing interactions for robot active learners. *Autonomous Mental Development, IEEE Transactions on*, 2(2):108–118, 2010.

K. Capek, N. Playfair, P. Selver, and W. Landes. Rossum's universal robots. *Prague, CZ*, 1920.

J. Y. Chai, L. She, R. Fang, S. Ottarson, C. Littley, C. Liu, and K. Hanson. Collaborative effort towards common ground in situated human-robot dialogue. In *Proceedings of the 2014 ACM/IEEE international conference on Human-robot interaction*, pages 33–40. ACM, 2014.

C. Chao, J. Lee, M. Begum, and A. L. Thomaz. Simon plays simon says: The timing of turn-taking in an imitation game. In *RO-MAN, 2011 IEEE*, pages 235–240. IEEE, 2011.

D. Chen and R. Mooney. Learning to sportscast: a test of grounded language acquisition. In *ICML 2008: Proc. of the 25th international conference on Machine learning*, pages 128–135, Helsinki, Finland, 2008. ACM.

D. Chen. Fast online lexicon learning for grounded language acquisition. In *Proc. of the Annual Meetings of the Association for Computational Linguistics (ACL)*, 2012.

D. Chen and R. Mooney. Learning to interpret natural language navigation instructions from observations. In *Proceedings of the 25th AAAI Conference on Artificial Intelligence (AAAI-2011)*, pages 859–865, 2011.

D. Chen, J. Kim, and R. Mooney. Training a multilingual sportscaster: using perceptual context to learn language. *Journal of Artificial Intelligence Research*, 37(1): 397–436, 2010.

D. Chiang. An Introduction to Synchronous Grammars. From the *ACL 2006 Tutorial on Synchronous Grammars and Tree Automata*, Sydney, Australia. `http://www.isi.edu/~chiang/papers/synchtut.pdf`, 2006.

H. H. Clark. Pointing and placing. *Pointing: Where language, culture, and cognition meet*, pages 243–268, 2003.

H. H. Clark. Coordinating with each other in a material world. *Discourse studies*, 7 (4-5):507–525, 2005.

J. Clarke, D. Goldwasser, M. Chang, and D. Roth. Driving semantic parsing from the world's response. In *Proc. of the Conf. on Computational Natural Language Learning*, 2010.

N. Dalal and B. Triggs. Histograms of Oriented Gradients for Human Detection. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2005.

K. Dautenhahn and A. Billard. Games Children with Autism can Play with Robota, A Humanoid Robotic Doll. In *Proceedings of the 1st Cambridge Workshop on Universal Access and Assistive Technology*, pages 179–190. Springer-Verlag, 2002.

R. Deits, S. Tellex, P. Thaker, D. Simeonov, T. Kollar, and N. Roy. Clarifying commands with information-theoretic human-robot dialog. *Journal of Human-Robot Interaction*, 2(2):58–79, 2013.

F. Duvallet, T. Kollar, and A. Stentz. Imitation learning for natural language direction following through unknown environments. In *Robotics and Automation (ICRA), 2013 IEEE International Conference on*, pages 1047–1053. IEEE, 2013.

J. Dzifcak, M. Scheutz, C. Baral, and P. Schermerhorn. What to do and how to do it: Translating natural language directives into temporal and dynamic logic representation for goal management and action execution. In *Proceedings of the 2009 IEEE International Conference on Robotics and Automation (ICRA '09)*, 2009.

R. Fan, K. Chang, C. Hsieh, X. Wang, and C. Lin. LIBLINEAR: A Library for Large Linear Classification. *Journal of Machine Learning Research*, 9:1871–1874, 2008.

R. Fang, M. Doering, and J. Y. Chai. Collaborative models for referring expression generation in situated dialogue. In *Proc. of the Twenty-eighth National Conference on Artificial Intelligence (AAAI)*, Québec City, Quebec, Canada, 2014.

A. Farhadi, I. Endres, D. Hoiem, and D. Forsyth. Describing objects by their attributes. In *IEEE Conf. on Computer Vision and Pattern Recognition*, 2009.

A. Farhadi, I. Endres, and D. Hoiem. Attribute-centric recognition for cross-category generalization. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conf. on*, pages 2352–2359. IEEE, 2010.

J. T. Feddema and O. Mitchell. Vision-guided Servoing with Feature-based Trajectory Generation for Robots. *IEEE Transactions on Robotics and Automation*, 5(5):691–700, 1989. ISSN 1042-296X. doi: 10.1109/70.88086.

P. Felzenszwalb, R. Girshick, D. McAllester, and D. Ramanan. Object detection with discriminatively trained part based models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(9):1627–1645, 2009.

A. Ferrein and G. Lakemeyer. Logic-based robot control in highly dynamic domains. *Robotics and Autonomous Systems*, 56(11), 2008.

M. A. Fischler and R. C. Bolles. Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography. *Communications of the ACM*, 24:381–395, 1981.

S. Friedman, H. Pasula, and D. Fox. Voronoi random fields: Extracting topological structure of indoor environments via place labeling. In M. M. Veloso, editor, *IJCAI*, pages 2109–2114, 2007.

D. Goldwasser and D. Roth. Learning from natural instructions. *Machine Learning*, 94(2):205–232, 2014.

M. A. Goodrich and A. C. Schultz. Human-robot interaction: a survey. *Foundations and Trends in HCI*, 1(3):203–275, 2007.

P. Gorniak and D. Roy. Understanding complex visually referring utterances. In *Proc. of the HLT-NAACL 2003 Workshop on Learning Word Meaning from Non-Linguistic Data*, 2003.

S. Guadarrama, N. Krishnamoorthy, G. Malkarnenkar, S. Venugopalan, R. Mooney, T. Darrell, and K. Saenko. Youtube2text: Recognizing and describing arbitrary activities using semantic hierarchies and zero-shot recognition. In *Proceedings of the 14th International Conference on Computer Vision (ICCV-2013)*, pages 2712–2719, Sydney, Australia, December 2013.

J.-S. Gutmann, M. Fukuchi, and M. Fujita. 3d perception and environment map generation for humanoid robot navigation. *Int. J. Rob. Res.*, 27(10):1117–1134, 2008.

D. Hähnel, W. Burgard, and G. Lakemeyer. GOLEX - bridging the gap between logic (GOLOG) and a real robot. In *Proc. of the German Conference on Artificial Intelligence (KI), Germany*, 1998.

S. Harnad. The symbol grounding problem. *Physica D: Nonlinear Phenomena*, 42 (1):335–346, 1990.

J. Harris and D. Barber. Speech and gesture interfaces for squad-level human-robot teaming. In *Proc. of Unmanned Systems Technology*, 2014.

Y. He and S. Young. Spoken language understanding using the hidden vector state model. *Speech Communication*, 48(3-4), 2006.

P. Henry, M. Krainin, E. Herbst, X. Ren, and D. Fox. RGB-D mapping: Using Kinect-style depth cameras for dense 3D modeling of indoor environments. *Int'l Journal of Robotics Research (IJRR)*, 31(4), 2012.

P. Henry, D. Fox, A. Bhowmik, and R. Mongia. Patch Volumes: Segmentation-based Consistent Mapping with RGB-D Cameras. In *International Conference on 3D Vision (3DV)*, 2013.

F. Ingrand, R. Chatila, R. Alami, and F. Robert. PRS: A high level supervision and control language for autonomous mobile robots. In *Proc. of the IEEE International Conference on Robotics & Automation (ICRA)*, 1996.

J. Jain, A. Lund, and D. Wixon. The future of natural user interfaces. In *Proc. of the 2011 annual conference extended abstracts on Human factors in computing systems*, pages 211–214. ACM, 2011.

A. Johnson and M. Hebert. Using spin images for efficient object recognition in cluttered 3D scenes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 21(5), 1999.

S. Kiesler. Fostering common ground in human-robot interaction. In *Robot and Human Interactive Communication, 2005. ROMAN 2005. IEEE International Workshop on*, pages 729–734. IEEE, 2005.

W. B. Knox, P. Stone, and C. Breazeal. Training a robot via human feedback: A case study. In *Social Robotics*, pages 460–470. Springer, 2013.

T. Kollar, S. Tellex, D. Roy, and N. Roy. Toward understanding natural language directions. In *HRI 2010: Proceedings of the 5th International Conference on Human-Robot Interaction*. ACM Press, 2010.

H. Kose-Bagci, K. Dautenhahn, and C. L. Nehaniv. Emergent Dynamics of Turn-taking Interaction in Drumming Games with a Humanoid Robot. In *Proceedings of the 17th IEEE International Symposium on Robot and Human Interactive Communication*. IEEE, 2008.

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

H. Kress-Gazit, T.Wongpiromsarn, and U. Topcu. Correct, reactive robot control from abstraction and temporal logic specifications. *IEEE Robotics and Automation Magazine, special issue on Formal Methods for Robotics and Automation*, 18(3): 65–74, 2011.

J. Krishnamurthy and T. Kollar. Jointly learning to parse and perceive: Connecting natural language to the physical world. *Transactions of the Association for Computational Linguistics*, 1:193–206, 2013.

G.-J. M. Kruijff. Symbol grounding as social, situated construction of meaning in human-robot interaction. *KI-Künstliche Intelligenz*, 27(2):153–160, 2013.

G.-J. M. Kruijff, M. Janíček, and H. Zender. Situated communication for joint activity in human-robot teams. *IEEE Intelligent Systems*, 27(2):0027–35, 2012.

T. Kwiatkowksi, L. Zettlemoyer, S. Goldwater, and M. Steedman. Inducing probabilistic ccg grammars from logical form with higher-order unification. In *EMNLP*, 2010.

T. Kwiatkowski, L. Zettlemoyer, S. Goldwater, and M. Steedman. Lexical generalization in CCG grammar induction for semantic parsing. In *Proc. of the Conf. on Empirical Methods in Natural Language Processing*, 2011.

T. Kwiatkowski, E. Choi, Y. Artzi, and L. Zettlemoyer. Scaling semantic parsers with on-the-fly ontology matching. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1545–1556, Seattle, Washington, USA, October 2013. Association for Computational Linguistics. URL `http://www.aclweb.org/anthology/D13-1161`.

K. Lai, L. Bo, X. Ren, and D. Fox. A large-scale hierarchical multi-view RGB-D object dataset. In *Proc. of the IEEE Int'l Conf. on Robotics & Automation (ICRA)*, 2011.

K. Lai, L. Bo, X. Ren, and D. Fox. RGB-D object recognition: Features, algorithms, and a large scale benchmark. In A. Fossati, J. Gall, H. Grabner, X. Ren, and K. Konolige, editors, *Consumer Depth Cameras for Computer Vision: Research Topics and Applications*, pages 167–192. Springer, 2013.

Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, November 1998.

H. Lee, R. Grosse, R. Ranganath, and A. Ng. Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations. In *Proc. of the Int'l Conf. on Machine Learning (ICML)*, 2009.

P. Liang, M. Jordan, and D. Klein. Learning dependency-based compositional semantics. In *Proc. of the Association for Computational Linguistics*, 2011.

P. Liu, D. F. Glas, T. Kanda, H. Ishiguro, and N. Hagita. It's not polite to point: generating socially-appropriate deictic behaviors towards people. In *Proceedings of the 8th ACM/IEEE international conference on Human-robot interaction*, pages 267–274. IEEE Press, 2013.

A. Lopez. Statistical machine translation. *ACM Comput. Surv.*, 40(3):1–49, 2008. ISSN 0360-0300.

D. Lowe. Distinctive Image Features from Scale-Invariant Keypoints. *International Journal of Computer Vision*, 60:91–110, 2004. doi: 10.1023/B:VISI.0000029664.99615.94.

M. Macmahon, B. Stankiewicz, and B. Kuipers. Walk the talk: Connecting language, knowledge, action in route instructions. In *In Proc. of the Nat. Conf. on Artificial Intelligence (AAAI)*, pages 1475–1482, 2006.

M. T. Macmahon and B. J. Kuipers. *Following natural language route instructions.* PhD thesis, University of Texas Libraries, 2007.

A. Malizia and A. Bellucci. The artificiality of natural user interfaces. *Communications of the ACM*, 55(3):36–38, 2012.

S. G. Mallat and Z. Zhang. Matching pursuits with time-frequency dictionaries. *IEEE Transactions on Signal Processing*, 41(12):3397–3415, 1993.

E. Martins, M. Pascoal, and J. Santos. A new improvement for a k shortest paths algorithm. *Investigação Operational*, 2001.

C. Matuszek, D. Fox, and K. Koscher. Following directions using statistical machine translation. In *ACM/IEEE International Conference on Human-Robot Interaction (HRI)*, 2010.

C. Matuszek, B. Mayton, R. Aimi, L. Bo, M. Deisenroth, R. Chu, M. Kung, L. LeGrand, J. Smith, and D. Fox. Gambit: A robust chess-playing robotic system. In *Proc. of the IEEE Int'l Conf. on Robotics & Automation (ICRA)*, 2011.

C. Matuszek, N. FitzGerald, L. Zettlemoyer, L. Bo, and D. Fox. A Joint Model of Language and Perception for Grounded Attribute Learning. In *Proceedings of the 2012 International Conference on Machine Learning*, Edinburgh, Scotland, June 2012a.

C. Matuszek, E. Herbst, L. Zettlemoyer, and D. Fox. Learning to parse natural language commands to a robot control system. In *Proc. of the 13th Int'l Symposium on Experimental Robotics (ISER)*, June 2012b.

C. Matuszek, L. Bo, L. Zettlemoyer, and D. Fox. Learning from unscripted deictic gesture and language for human-robot interactions. In *Proc. of the Twenty-eighth National Conference on Artificial Intelligence (AAAI)*, Québec City, Quebec, Canada, March 2014.

N. Mavridis and D. Roy. Grounded situation models for robots: Where words and percepts meet. In *IEEE/RSJ Int'l Conf. on Intelligent Robots and Systems*, 2006.

B. Mayton, L. LeGrand, and J. R. Smith. An Electric Field Pretouch System for Grasping and Co-Manipulation. In *Proceedings of the 2010 IEEE International Conference on Robotics and Automation (ICRA 2010)*. IEEE, 2010.

K. K. McGregor, K. J. Rohlfing, A. Bean, and E. Marschner. Gesture as a support for word learning: The case of under. *Journal of Child Language*, 36, 9 2009.

S. Mitra and T. Acharya. Gesture recognition: A survey. *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*, 37(3):311–324, May 2007.

S. Mohan and J. E. Laird. Learning goal-oriented hierarchical tasks from situated interactive instruction. In *Proc. of the Twenty-eighth National Conference on Artificial Intelligence (AAAI)*, Québec City, Quebec, Canada, 2014.

S. Mohan, A. Mininger, and J. E. Laird. Towards an indexical model of situated language comprehension for real-world cognitive agents. In *Proceedings of the Second Annual Conference on Advances in Cognitive Systems*, 2013.

R. J. Mooney. Learning to connect language and perception. In D. Fox and C. P. Gomes, editors, *Proc. of the Twenty-Third AAAI Conf. on Artificial Intelligence, AAAI 2008*, pages 1598–1601, Chicago, Illinois, July 2008. AAAI Press.

A. Newell and H. A. Simon. Computer science as empirical inquiry: Symbols and search. *Communincations of the ACM*, 19(3):113–126, March 1976.

K. Nickel and R. Stiefelhagen. Visual recognition of pointing gestures for human–robot interaction. *Image and Vision Computing*, 25(12):1875–1884, 2007.

F. J. Och and H. Ney. A systematic comparison of various statistical alignment models. *Computational Linguistics*, 29(1):19–51, 2003.

D. Parikh and K. Grauman. Relative attributes. In *Int'l Conf. on Computer Vision*, 2011.

J. Peltason, N. Riether, B. Wrede, and I. Lütkebohle. Talking with robots about objects: a system-level evaluation in hri. In *Proceedings of the 7th annual ACM/IEEE international conference on Human-Robot Interaction*. ACM, 2012.

D. Perzanowski, A. C. Schultz, W. Adams, E. Marsh, and M. Bugajska. Building a multimodal human-robot interface. *Intelligent Systems, IEEE*, 16(1):16–21, 2001.

A. Quattoni, S. Wang, L. p Morency, M. Collins, T. Darrell, and M. Csail. Hidden-state conditional random fields. In *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2007.

M. Quigley, B. Gerkey, K. Conley, J. Faust, T. Foote, J. Leibsz, E. Berger, R. Wheeler, and A. Ng. ROS: An Open-source Robot Operating System. In *Open-source Software Workshop of the International Conference on Robotics and Automation*. IEEE, 2009.

A. Ramey, V. González-Pacheco, and M. A. Salichs. Integration of a low-cost rgb-d sensor in a social robot for gesture recognition. In *Human-Robot Interaction (HRI), 2011 6th ACM/IEEE International Conference on*, pages 229–230. IEEE, 2011.

N. D. Ratliff, J. A. Bagnell, and M. A. Zinkevich. Maximum margin planning. In *In Proceedings of the 23rd International Conf. on Machine Learning (ICML)*, 2006.

H. Reckman, J. Orkin, and D. Roy. Learning meanings of words and constructions, grounded in a virtual game. In *Proc. of the 10th Conf. on Natural Language Processing (KONVENS)*, 2010.

P. Rouanet, F. Danieau, and P.-Y. Oudeyer. A robotic game to evaluate interfaces used to show and teach visual objects to a robot in real world condition. In *Human-Robot Interaction (HRI), 2011 6th ACM/IEEE International Conference on*, pages 313–320. IEEE, 2011.

P. Rouanet, P.-Y. Oudeyer, F. Danieau, and D. Filliat. The Impact of Human-Robot Interfaces on the Learning of Visual Objects. *IEEE Transactions on Robotics*, 29 (2):525–541, 2013.

A. Sandygulova, A. G. Campbell, M. Dragone, and G. O'Hare. Immersive human-robot interaction. In *Proc. of the seventh annual ACM/IEEE international conference on Human-Robot Interaction*, HRI '12, pages 227–228, New York, NY, USA, 2012. ACM.

S. Schaffer. Enlightened Automata. In *The Sciences in Enlightened Europe*. University of Chicago Press, 1999.

B. Schölkopf and A. J. Smola. *Learning with Kernels—Support Vector Machines, Regularization, Optimization, and Beyond*. The MIT Press, 2002.

J. R. Searle. Minds, brains, and programs. *Behavioral and brain sciences*, 3(03): 417–424, 1980.

N. Shimizu and A. Haas. Learning to Follow Navigational Route Instructions. In *Int'l Joint Conf. on Artificial Intelligence (IJCAI)*, 2009.

M. Skubic, D. Perzanowski, S. Blisard, A. Schultz, W. Adams, M. Bugajska, and D. Brock. Spatial language for human-robot dialogs. *IEEE Transactions on Sys-*

*tems, Man, and Cybernetics, Part C, Special Issue on Human-Robot Interaction*, 34(2):154–167, May 2001.

T. Standage. *The Turk: The Life and Times of the Famous Eighteenth-Century Chess-playing Machine*. Walker, 2002.

M. Steedman. *Surface Structure and Interpretation*. The MIT Press, 1996.

M. Steedman. *The Syntactic Process*. MIT Press, 2000.

L. Steels. Language games for autonomous robots. *IEEE Intelligent systems*, pages 16–22, October 2001. URL `http://groups.lis.illinois.edu/amag/langev/paper/steels01languageGames.html`.

L. Steels and M. Loetzsch. The grounded naming game. In *Experiments in Cultural Language Evolution*. 2012.

Y. Sun, L. Bo, and D. Fox. Attribute based object identification. In *Robotics and Automation (ICRA), 2013 IEEE International Conference on*, pages 2096–2103, May 2013.

S. Tellex, T. Kollar, S. Dickerson, M. Walter, A. Banerjee, S. Teller, and N. Roy. Understanding natural language commands for robotic navigation and mobile manipulation. In *Proceedings of the National Conference on Artificial Intelligence (AAAI)*, August 2011.

S. Tellex, T. Kollar, G. Shaw, N. Roy, and D. Roy. Grounding spatial language for video search. In W. Gao, C.-H. Lee, J. Y. 0001, X. Chen, M. Eskenazi, and Z. Zhang, editors, *ICMI-MLMI*. ACM, 2010.

S. Tellex, R. A. Knepper, A. Li, T. M. Howard, D. Rus, and N. Roy. Asking for help using inverse semantics. In *Robotics: Science and Systems (RSS)*, Berkeley, California, 2014a.

S. Tellex, P. Thaker, J. Joseph, and N. Roy. Learning perceptually grounded word meanings from unaligned parallel data. *Machine Learning*, 94(2):151–167, 2014b.

J. Thomason, S. Venugopalan, S. Guadarrama, K. Saenko, and R. Mooney. Integrating language and vision to generate natural language descriptions of videos in the wild. In *Proceedings of the 25th International Conference on Computational Linguistics (COLING 2014)*, Dublin, Ireland, August 2014.

S. Thrun, W. Burgard, and D. Fox. *Probabilistic Robotics*. MIT Press, Cambridge, MA, September 2005. ISBN 0-262-20162-3.

E. A. Topp, H. Huettenrauch, H. I. Christensen, and K. S. Eklundh. Bringing together human and robotic environment representations-a pilot study. In *Intelligent Robots and Systems, 2006 IEEE/RSJ International Conference on*, pages 4946–4952. IEEE, 2006.

Y. Tsuruoka and J. Tsujii. Iterative cky parsing for probabilistic context-free grammars. *Proceedings of the International Joint Conference on Natural Language Processing*, 2005.

A. M. Turing. Computing machinery and intelligence. *Mind*, 59(236):433–460, 1950.

A. Vogel and D. Jurafsky. Learning to follow navigational directions. In *Proc. of the Association for Computational Linguistics*, 2010.

A.-L. Vollmer, K. S. Lohan, K. Fischer, Y. Nagai, K. Pitsch, J. Fritsch, K. Rohlfing, and B. Wrede. People modify their tutoring behavior in robot-directed interaction for action learning. Development and Learning, 2009. ICDL 2009. IEEE 8th International Conference on Development and Learning, pages 1–6. IEEE, 2009.

J. P. Wachs, M. Kölsch, H. Stern, and Y. Edan. Vision-based hand-gesture applications. *Commun. ACM*, 54(2):60–71, February 2011. ISSN 0001-0782.

M. R. Walter, S. Hemachandra, B. Homberg, S. Tellex, and S. Teller. Learning semantic maps from natural language descriptions. Robotics: Science and Systems, 2013.

M. R. Walter, S. Hemachandra, B. Homberg, S. Tellex, and S. Teller. A framework for learning semantic maps from grounded natural language descriptions. *The International Journal of Robotics Research*, 33(9):1167–1190, 2014.

Y. Wei, E. Brunskill, T. Kollar, and N. Roy. Where to go: Interpreting natural directions using global inference. In *Int'l Conf. on Robotics and Automation (ICRA)*, 2009.

Y. W. Wong. *Learning for Semantic Parsing and Natural Language Generation Using Statistical Machine Translation Techniques*. PhD thesis, Univ. of Texas at Austin, August 2007.

Y. W. Wong and R. J. Mooney. Learning for semantic parsing with statistical machine translation. In *Proc. of the main conference on Human Language Technology Conf. of the North American Chapter of the Association of Computational Linguistics*, pages 439–446. Association for Computational Linguistics, 2006.

Y. Wong and R. Mooney. Learning synchronous grammars for semantic parsing with lambda calculus. In *Proceedings of the Association for Computational Linguistics*, 2007.

H.-D. Yang, A.-Y. Park, and S.-W. Lee. Gesture spotting and recognition for human–robot interaction. *Robotics, IEEE Transactions on*, 23(2):256–270, 2007.

J. Yang, K. Yu, Y. Gong, and T. Huang. Linear spatial pyramid matching using sparse coding for image classification. In *Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2009.

J. Y. Yen. Finding the k shortest loopless paths in a network. *Management Science*, 17(11):712–716, 1971.

K. Yu, T. Zhang, and Y. Gong. Nonlinear Learning using Local Coordinate Coding . In *NIPS*, December 2009.

J. Zelle and R. Mooney. Learning to parse database queries using inductive logic programming. In *Proc. of the National Conf. on Artificial Intelligence*, 1996.

H. Zender, O. Martínez Mozos, P. Jensfelt, G.-J. Kruijff, and W. Burgard. Conceptual spatial representations for indoor mobile robots. *Robotics and Autonomous Systems*, 56(6):493–502, 2008.

L. Zettlemoyer and M. Collins. Learning to map sentences to logical form: Structured classification with probabilistic categorial grammars. In *Proc. of the Conf. on Uncertainty in Artificial Intelligence*, 2005.

L. Zettlemoyer and M. Collins. Online learning of relaxed CCG grammars for parsing to logical form. In *Proc. of the Joint Conf. on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, 2007.

B. Ziebart, A. Maas, A. Dey, and J. D. Bagnell. Navigate like a cabbie: Probabilistic reasoning from observed context-aware behavior. In *UBICOMP: Ubiquitious Computation*, 2008.