Generalized Grounding Graphs: A Probabilistic Framework for Understanding Grounded Commands

Thomas Kollar[1], Stefanie Tellex[1], Matthew R. Walter,
Albert Huang, Abraham Bachrach, Sachi Hemachandra, Emma Brunskill,
Ashis Banerjee, Deb Roy, Seth Teller, Nicholas Roy

December 26, 2021

Abstract

Many task domains require robots to interpret and act upon natural language commands which are given by people and which refer to the robot's physical surroundings. Such interpretation is known variously as the symbol grounding problem (Harnad, 1990), grounded semantics (Feldman et al., 1996) and grounded language acquisition (Nenov and Dyer, 1993, 1994). This problem is challenging because people employ diverse vocabulary and grammar, and because robots have substantial uncertainty about the nature and contents of their surroundings, making it difficult to associate the constitutive language elements (principally noun phrases and spatial relations) of the command text to elements of those surroundings. Symbolic models capture linguistic structure but have not scaled successfully to handle the diverse language produced by untrained users. Existing statistical approaches can better handle diversity, but have not to date modeled complex linguistic structure, limiting achievable accuracy. Recent hybrid approaches have addressed limitations in scaling and complexity, but have not effectively associated linguistic and perceptual features. Our framework, called Generalized Grounding Graphs (G^3), addresses these issues by defining a probabilistic graphical model dynamically according to the linguistic parse structure of a natural language command. This approach scales effectively, handles linguistic diversity, and enables the system to associate parts of a command with the specific objects, places, and events in the external world to which they refer. We show that robots can learn word meanings and use those learned meanings to robustly follow natural language commands produced by untrained users. We demonstrate our approach for both mobility commands (e.g. route directions like "Go down the hallway through the door") and mobile manipulation commands (e.g. physical directives like "Pick up the pallet on the truck") involving a variety of semi-autonomous robotic platforms, including a wheelchair, a micro-air vehicle, a forklift, and the Willow Garage PR2.

[1]The first two authors contributed equally to this paper.

1 Introduction

To be useful teammates to human partners, robots must be able to robustly follow spoken instructions. For example, a human supervisor might tell an autonomous forklift, "Put the tire pallet on the truck," or the occupant of a wheelchair equipped with a robotic arm might say, "Get me the book from the coffee table." Understanding such commands is challenging for a robot because they involve verbs ("Put"), noun phrases ("the tire pallet"), and prepositional phrases ("on the truck"), each of which must be grounded to aspects of the world and which may be composed in many different ways. Figure 1 shows some of the wide variety of human-generated commands that might be given to different robots in different situations.

Traditional approaches, starting with Winograd (1971), have manually created symbol systems that map between language and the external world, connecting each linguistic term onto a space of pre-specified actions and environmental features (Hsiao et al., 2003; Roy et al., 2003; Bugmann et al., 2004; Roy, 2005; MacMahon et al., 2006; Kress-Gazit and Fainekos, 2008; Dzifcak et al., 2009). This class of systems takes advantage of the structure of language, but usually does not involve learning and has a fixed action space, limiting the ability of the resulting system to robustly understand language produced by untrained users.

Statistical approaches address these limitations by using data-driven training to learn robust models of word meanings (Shimizu and Haas, 2009; Regier, 1992; Branavan et al., 2009, 2012; Vogel and Jurafsky, 2010). However, these approaches use a fixed and flat sequential structure that does not capture the argument structure of language (e.g., it does not allow for variable arguments or nested clauses). At training time, a system that assumes a flat structure sees the entire phrase "the pallet beside the truck" and has no way to separate the meanings of relations like "beside" from objects such as "the truck." Furthermore, a flat structure ignores the argument structure of verbs. For example, the command "put the box pallet on the ground beside the truck," has two arguments ("the box pallet" and "on the ground beside the truck"), both of which must be isolated in order to infer the appropriate meaning of the verb "put" in this instance. To infer the meaning of unconstrained natural language commands, it is critical for the model to reason over these compositional and hierarchical linguistic structures at both learning and inference time.

Existing approaches that combine statistical and symbolic approaches assume access to a predefined lexicon of symbols that they then learn to map to natural language (Artzi and Zettlemoyer, 2013; Chen and Mooney, 2011; Matuszek et al., 2012b; Ge and Mooney, 2005; Liang et al., 2011; Arumugam et al., ress; Misra et al., 2016). These symbols must be provided by the designer to the system in advance. Newer approaches (Misra et al., 2017a) use deep learning in an MDP setting to map between actions and visual features, but do not decompose the language command according to the parse structure, which makes it harder to reason about what parts of the command were and were not understood by the system.

To address these issues, we present a framework called Generalized Grounding Graphs (G^3). A *grounding graph* is a probabilistic graphical model defined dynamically according to the compositional and hierarchical structure of a natural language command. The model predicts physical interpretations or *groundings* for linguistic constituents. Groundings are specific physical concepts that are referred to by the language and can be objects (e.g., a truck or a door), places (e.g., a particular location in the world), paths (e.g., a trajectory through the environment), or actions / events (e.g., a sequence of actions taken by the robot). The system is trained in a supervised way, using a corpus of language paired with groundings, enabling it to learn probabilistic predicates that map between language and groundings from lower-level features, without prespecifying the predicates in advance. At inference time, the system is given a natural language command and infers the most probable set of groundings in the external world. For example, for a command such as "Put the tire pallet on the truck," the system infers that the noun phrase "the tire pallet" maps to a specific pallet in the robot's representation of nearby objects, and the prepositional phrase "on the truck" maps to a particular location in the environment. For the entire sentence, the robot infers a specific sequence of actions that it should execute.

We evaluate G^3 on four robotic domains: a robotic forklift, the PR2 mobile manipulator, a robotic wheelchair, and a robotic micro-air vehicle (MAV). The forklift domain, shown in Figure 1(a), considers mobile manipulation commands, such as "Pick up the tire pallet." This domain shows that our approach is able to understand hierarchical and compositional commands, learning verbs, such as "put" and "take," as well as spatial relations, such as "on" and "to." The wheelchair and MAV domains demonstrate the ability to interpret longer route direction commands for ground and aerial robots. Figures 1(b) and 1(d) shows samples of these longer commands from the route direction domain. Performing inference in the full hierarchical model is computationally expensive and becomes intractable when understanding these longer route directions. We present an approximation of the full hierarchical structure that is able to successfully follow many route directions to within 10 meters of the destination, along with real-world

Place the pallet of tires on the left side of the trailer.

Arrange tire pallet to the truck.

Please lift the set of six tires up and set them on the trailer, to the right of the set of tires already on it.

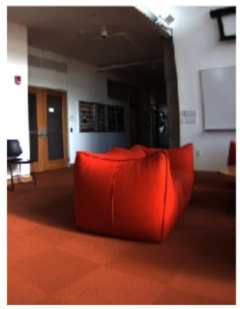
Load the skid right next to the other skid of tires on the trailer.

Place a second pallet of tires on the trailer.

Put the tire pallet on the trailer.

Place the pallet of tires on the right side of the truck.

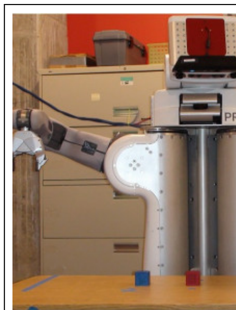
(a) Robot forklift.



With your back to the windows, walk straight through the door near the elevators. Continue to walk straight, going through one door until you come to an intersection just past a whiteboard. Turn left, turn right, and enter the second door on your right (sign says “Administrative Assistant”).

Go through the set of double doors by the red couches. Continue straight down the corridor, pass through two sets of wooden double doors, and enter through the gray door facing you with a phone next to it. Take a left through the blue double doors, and follow the corridor. The room is the second door on the left after you have rounded the corner.

(b) Robot wheelchair.



Pick up the red block.

Pick up the blue block.

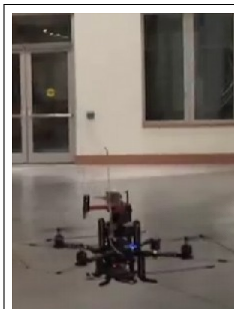
Put the red block near the blue block.

Put the blue block behind the red block.

Pick up the blue box and keep it on the right side.

Pick up the blue block and place it to the far right of the orange block.

(c) PR2.



Go forward until you are able to make a left. Then move ahead until you reach the opposite wall, then make a right. Go straight, past one staircase and to the next staircase. At the top of this staircase you will find a fire alarm on your left at approximately 7ft up.

Turn right and go ahead through the gap between two walls to a large white column. Turn right again and move until you are the base of the staircase to your right. Ahead, suspended from the ceiling is a large white butterfly sculpture. (*sic*)

Go forward ten feet and turn right. Go down the hall and over the stairs. Look straight ahead at the wall at the end of the walkway.

(d) Micro-air vehicle.

Figure 1: Four domains studied in this paper, along with sample commands from the evaluation corpora used for each domain.

demonstrations of our approach on a micro-air vehicle.

This article provides the definitive statement of the G^3 framework, unifying our previous work in this area (Kollar et al., 2010b; Huang et al., 2010; Kollar et al., 2010a; Tellex et al., 2011a). It precisely specifies the technical details of our approach, and provides corpus-based evaluations and robotic demonstrations in several domains. Tellex et al. (2011b) gives a more reflective overview of the framework, focusing on learned word meanings, but that work does not include our results on indoor mobility domains, and does not present all domains in a unified technical framework. In our more recent work, we have extended the G^3 framework in several ways. Tellex et al. (2013) and Deits et al. (2013) showed how to use G^3 to ask questions based on entropy when the robot is "confused," i.e. when its uncertainty is high. The G^3 framework can also be used by the robot to generate a request for help, by inverting the semantics model (Knepper et al., 2013). Bollini et al. (2012) applied the framework to the cooking domain, enabling a robot to follow natural language recipes. Tellex et al. (2013) described how to train the G^3 framework with less supervision. Krishnamurthy and Kollar (2013) and Kollar et al. (2013a) introduced a compositional parser based on a combinatory categorial grammar (CCG); the parser can be trained jointly using limited supervision. Expanding beyond understanding, Kollar et al. (2013b) used dialog to learn new symbols and referring expressions for physical locations, enabling robots to execute commands to find and fetch objects. Similar to the local version of the G^3 model, Duvallet et al. (2013) described an approach for learning a policy that follows directions explicitly in unknown environments. Most recently, Howard et al. (2014) and Paul et al. (2016) described hierarchical extensions that enable model inference over learned abstractions.

2 Related Work

The problem of robust natural language understanding has been studied since the earliest days of artificial intelligence. Beginning with SHRDLU (Winograd, 1971), many systems have exploited the compositional structure of language to generate symbolic representations of the natural language input, for example with approaches based on formal logic (Kress-Gazit and Fainekos, 2008; Dzifcak et al., 2009) or symbol grounding (Skubic et al., 2004). Notably, MacMahon et al. (2006) developed a symbol-based system for following route directions through simulated environments, and Hsiao et al. (2008) created a system for enabling a humanoid robot to follow natural language commands involving manipulation of table-top objects. These systems exploit the structure of language, but usually do not involve learning, and have a fixed action space. Our work, in contrast, defines a probabilistic graphical model according to the structure of the natural language command, inducing a probability distribution over the mapping between words and groundings in the external world. This factored structure enables the system to understand novel commands never seen during training, by compositionally combining learned word meanings according to the factorization induced by the command.

Another approach is to associate language with different aspects of the environment, typically by employing statistical methods to learn the mapping. Harnad (1990) called the problem of mapping between words in the language and aspects of the external world the "symbol grounding problem." Some systems learn the meaning of words directly in the sensorimotor space (e.g., joint angles and images) of the robot (Roy and Pentland, 2002; Sugita and Tani, 2005; Modayil and Kuipers, 2007; Marocco et al., 2010). By treating linguistic terms as a sensory input, these systems must learn directly from complex features extracted by perceptual modules, limiting the set of commands that they can robustly understand, and their ability to handle complex syntactic structures. Other computational approaches use more complex syntactic structures but can handle just a few words, such as "above," (Regier, 1992) or "near" (Carlson and Covey, 2005); Tellex et al. (2010) describe models for just six spatial prepositions applied to the problem of video retrieval. Our work, in contrast, learns grounded word meanings for the much wider variety of words that appear in a training set.

A different approach is to apply semantic parsing, which maps natural language sentences to symbolic semantic representations. Early work in semantic parsing uses labeled data that consists of sentences paired with their logical form, beginning with Thompson and Mooney (2003) on the GeoQuery dataset, which consists of natural language queries about geography. More recent work that considers this dataset and others provided improved training methods (Zettlemoyer and Collins, 2005b; Wong and Mooney, 2007; Piantadosi et al., 2008; Kwiatkowski et al., 2010). Some of these approaches have been applied to command understanding for robotics (Chen and Mooney, 2011; Matuszek et al., 2012b; Artzi and Zettlemoyer, 2013). However these approaches require the designers to provide relational predicates to the system that map between natural language and world knowledge, such as TO or NEAR. Our approach, in contrast, learns predicates using lower-level features. Shimizu (2006) and Shimizu and Haas (2009) train a CRF to map from language to a small fixed command set for following route directions;

grounding occurs by applying these commands to a backend executor. Newer approaches learn symbolic word meanings with less supervision; Poon and Domingos (2009) presented an approach to unsupervised semantic parsing, while other approaches use an external reward signal (Liang et al., 2011; Clarke et al., 2010). Our work, in contrast, requires more supervision, but learns grounded meaning representations in terms of perceptual features rather than symbols. Liang et al. (2006) created an end-to-end discriminative approach to machine translation that introduced a correspondence structure between the input and output translations. As in their approach, we optimize over all possible groundings within a correspondence structure that allows us to introduce features that measure the mapping between words and groundings, measuring the faithfulness of particular groundings in the environment as they correspond to words in the language.

Cohen and Oviatt (1995) report that speech interfaces are useful when the hands and eyes are otherwise engaged and when there is limited availability of keyboards or screens. Robots that operate in unstructured, real-world environments fit these scenarios perfectly. Despite these characteristics of human-robot interaction problem, there is no consensus that human-robot interfaces should be built around natural language, due to the challenges in building dialog interfaces in changing social contexts (Fong et al., 2003; Severinson-Eklundh et al., 2003). The aim of our work is to develop robust natural language systems so that robots can interact with people flexibly using language.

Other recent work built upon semantic parsing by explicitly considering perception at the same time as parsing (Matuszek et al., 2012a; Krishnamurthy and Kollar, 2013; Kollar et al., 2013a; Andreas et al., 2016). These approaches used limited supervision to learn models that connect language to sets of objects, attributes and relations. Other work used attention-based neural networks to answer questions about images Xu and Saenko (2015); Yang et al. (2015); Zitnick et al. (2016), which do not generally use an intermediate parse. Although G^3 uses full supervision to train its semantic parser and grounding classifiers, it is able to additionally learn to understand verb phrases such as "pick up" and "put," which are challenging because such learning involves parameter estimation and inference over state sequences.

Newer approaches use learning in an MDP setting to map between actions and visual features. Some existing approaches learn policies to map language to the correct action (Branavan et al., 2009; Vogel and Jurafsky, 2010). These approaches can learn from weaker supervision but do not explicitly learn word meanings or probabilistic predicates as in our approach. Misra et al. (2017b) learns to map between actions in a predefined space and visual observations in an integrated system, but does not decompose word meanings into separately trained factors. Separately trained models allow us to decompose word meanings into separate factors and recombine them later to understand novel commands, as well as learn spatially relevant word meanings that can be applied later in different settings. Janner et al. (2017) created a system for spatial reasoning for a mobile robot that uses deep reinforcement learning. This approach automatically learns end poses for a mobile robot, including relative expressions like "above the westernmost rock." However it only handles movement commands and does not do manipulation. Additionally, the approach does not decompose into separate factors, which makes it harder to ask targeted questions. Andreas and Klein (2015) created a system that explicitly models low-level compositional structure for instruction following. They use the same technical term, *grounding graph* to describe the perceptual representation of the world model, rather than the grounding structure to map between language and the external world. By using graph semantics, the work created a map between nonbinary features and learned concepts such as "you are on top of the hill."

3 Generalized Grounding Graphs

To understand natural language commands, a robot must be able to map between the linguistic elements of a command, such as "Pick up the tire pallet," and the corresponding aspects of the external world. Each constituent phrase in the command refers to a particular object, place, or action that the robot should execute in the environment. We refer to the object, place, path, or event as the *grounding* for a linguistic constituent. The aim of language understanding is to find the set of most probable groundings Γ given a parsed natural language command Λ and the robot's model of the environment, M.

$$\operatorname*{argmax}_{\Gamma} p(\Gamma|\Lambda, M). \tag{1}$$

The environment model M consists of the robot's location along with the locations, structure and appearance of objects and places in the external world. It defines a space of possible values for the set of groundings $\gamma_i \in \Gamma$. A robot computes the environment model using sensor input. An object such as a pallet is represented as a three-dimensional geometric shape, along with a set of symbolic tags that might be produced by an object classifier, such as "pallet" or "truck." A place grounding represents a particular location in the map, corresponding to a phrase such as "on the truck." Path and

Variable	Description
$\Gamma : \gamma_1 \dots \gamma_N$	Grounding; can be an object, place, path or event in the world. Each γ_n consists of a tuple (g, t, p).
$\Lambda : \lambda_1 \dots \lambda_M$	Language command. Each linguistic constituent λ_m refers to a subset of grounding variables based on the parse structure of the command.
$\Phi : \phi_1 \dots \phi_M$	Correspondence variable; Each there is a $\phi_m \in \{0, 1\}$ for each λ_m and a subset of the grounding variables γ_n.
$\Psi : \psi_1 \dots \psi_M$	Factors defined according to the parse structure of the command.
M	Environment model; the robot's model of the external world, acquired from its sensors.
Z	Normalization constant.
g	A three-dimensional shape of an object grounding γ_m, such as a room, or the vicinity of an object. It is expressed as a set of points that define a polygon $(x_1, y_1), \dots, (x_N, y_N)$ together with a height z (e.g., as a vertical prism).
$p \in \mathbb{R}^{T\times 7}$	a sequence of T points. Each point is a pose for the region, g. It consists of a tuple $(\tau, x, y, z, roll, pitch, yaw)$ representing the location and orientation of g at time τ (with location interpolated linearly across time). The combination of the shape, g, and the trajectory, T, define a three-dimensional region in the environment corresponding to the object at each time step.
t	A set of pre-defined textual tags $\{tag_1, \dots, tag_M\}$ that are the output of perceptual classifiers, such as object recognizers or scene classifiers.
s_j	Feature functions.
θ_j	Feature weights.

Table 1: Table of variables.

action groundings are constrained by the environment, but are not fully defined by it, since the continuous state space of the environment defines an infinite space of paths for the robot and objects. As such, the environment model defines the starting location for the robot and the objects in the environment, from which paths of the robot and objects can be derived. A path grounding is a trajectory of points over time, corresponding to a phrase such as "to the elevators." An action or event grounding consists of the robot's trajectory over time as it performs some action (e.g., as it picks up a pallet).[2] The groundings Γ correspond to n items $\gamma_1 \dots \gamma_n$, where each γ_i is a tuple, (g, t, p); g is the three dimensional shape of the object, t is a trajectory through space, and p is a set of symbolic perceptual tags. These variables are defined more formally in Table 1.

Since learning the joint distribution over commands and groundings is intractable, we must factor Equation 1 into simpler components. Natural language has a well-known compositional, hierarchical argument structure (Jackendoff, 1983), dividing a sentence into a parse tree where each node is one of the *linguistic constituents* λ_i (Akmajian, 2010) produced by the parser. Each linguistic constituent has corresponding grounding variables $\{\gamma_i, \dots, \gamma_{i+k}\} \in \Gamma$ that are automatically instantiated based on the structure of the parse and is scored using a factor ψ_m; each factor ψ_m defines the score of a *probabilistic predicate* for the linguistic constituent λ_i, and factors as follows:

$$p(\Gamma|\Lambda, M) = \frac{1}{Z} \prod_m \psi_m(\lambda_m, \gamma_{m_1} \dots \gamma_{m_k}). \tag{2}$$

In G^3, each factor ψ_m is trained as a binary classifier that predicts which assignments to the grounding variables make sense for each linguistic constituent. These binary classes are introduced via a correspondence variable $\phi_i \in \{0, 1\}$. When $\phi_i = 1$ for a linguistic constituent, then the probabilistic predicate is true; when $\phi_i = 0$, then it is false. For example, the probabilistic predicate for "the pallet" is true for object 3, but not for object 1 in Figure 2(b). The resulting random variables and factors for each linguistic constituent in the parse tree therefore factor as follows:

[2] In general, an event could consist of any state change in the environment, for example the generation of an appropriate sound for the command, "Speak your name." In this paper, we focus on events that can be represented geometrically.

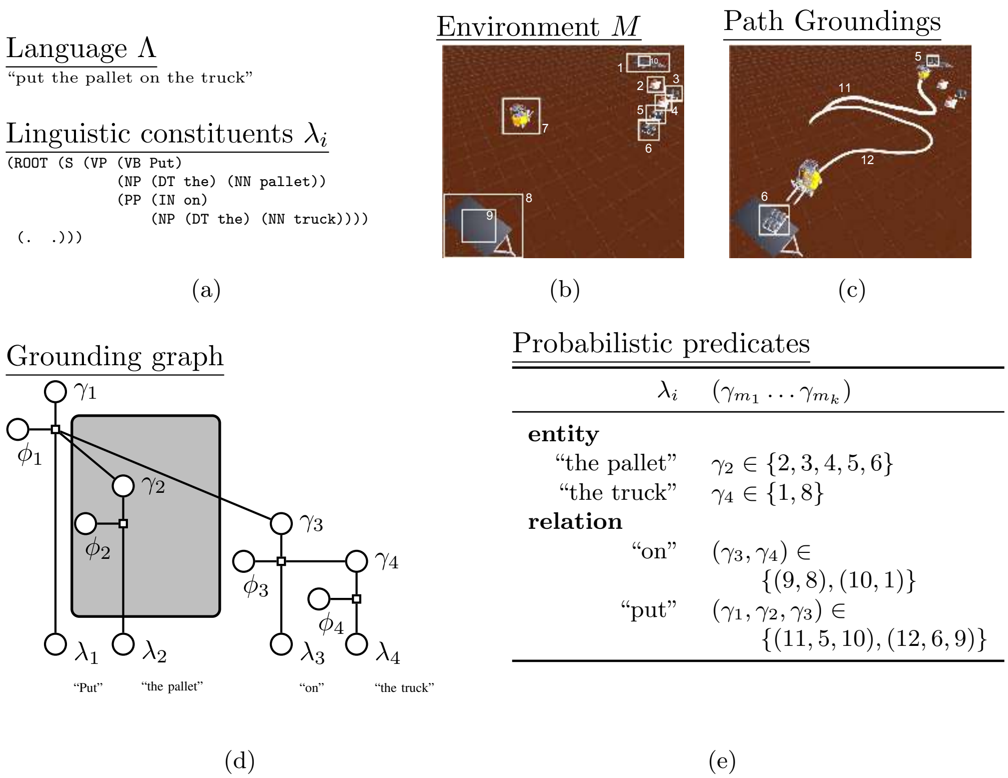


Figure 2: Grounding graph for the phrase "put the pallet on the truck.". Both the entity and relation candidates have been simplified, as in general many more candidates are considered. The best decoding result would result in $(\gamma_1, \gamma_2, \gamma_3, \gamma_4) \in \{(11, 5, 10, 1), (12, 6, 9, 8))\}$ having high probability.

$$p(\Gamma|\Lambda, \Phi, M) = \frac{1}{Z} \prod_m \psi_m(\phi_m, \lambda_m, \gamma_{m_1} \dots \gamma_{m_k}). \tag{3}$$

The factorization in Equation 3 is the core of the grounding graph; we can infer the grounding of linguistic constituents independently, and the likelihood of the entire text is the the joint likelihood of the independent factors. We ensure that we can preserve needed correlations between groundings by introducing two types of factors ψ. Entity factors are those that predict the assignment of a constituent to a single specific grounding $(\psi(\phi_i, \lambda_i, \gamma_{m_1}))$, typically a noun grounded to some object or other discrete concept in the world. In contrast, relational factors predict the correspondence of a constituent across two groundings (e.g., $\psi(\phi_i, \lambda_i, \gamma_{m_1}, \gamma_{m_2})$). Relational factors can have arity greater than two ($\psi(\phi_i, \lambda_i, \gamma_{m_1}, \gamma_{m_2}, \gamma_{m_3})$) and are mainly used for multi-argument verbs, such as "put."[3]. In both cases, the individual factors ψ_m quantify the correlation between words in the linguistic constituent λ_m and the groundings $\gamma_{m_1} \ldots \gamma_{m_k}$; the factors should have large values where words correspond well to groundings and small values otherwise.

The specific entity and relational factors ψ_m are created automatically for each sentence depending on the syntactic type of linguistic constituents obtained from parsing that sentence, and the parametric form of the ψ_m will vary from application to application. For example, for the application of mobile manipulation (section 4), we learn the ψ_m as logistic regression classifiers, but for the application of route following (Section 5), we learn the ψ_m as category distributions. We discuss in each section how we chose the different correspondence models.

For example, Figure 2 shows the graphical model for the phrase "Put the pallet on the truck." There are two entity grounding variables γ_2 and γ_4 for the noun phrases "the pallet" and "the truck" and values for these variables range over objects in the environment. Note that a γ_i could be referenced by more than one phrasal constituent; this translates to being connected to more than one factor, as we see that the value of γ_4 affects both the factor ψ_4 for "the truck" and the factor ψ_3 for "on". However, the value for γ_4 does not affect the grounding for ψ_2, "the pallet" — the model relies on the "on" factor to ensure that the object grounded to "the pallet" is the same object that is on "the truck", and that it is indeed on the truck. Such independence assumptions enable efficient inference and learning: our approach can learn a model for the word "on" that can be used whenever this word is encountered, without needing to manually introduce a symbolic predicate for this concept. We describe how to formally derive the structural independencies from the parse tree in Algorithm 1.

In order to enable efficient learning of generalizable predicates, we use locally normalized factors for ψ_m. Specifically:

$$\psi_m(\phi_m, \lambda_m, \gamma_{m_1} \ldots \gamma_{m_k}) \equiv p(\phi_m | \lambda_m, \gamma_{m_1} \ldots \gamma_{m_k}). \tag{4}$$

We can then write the factored form:

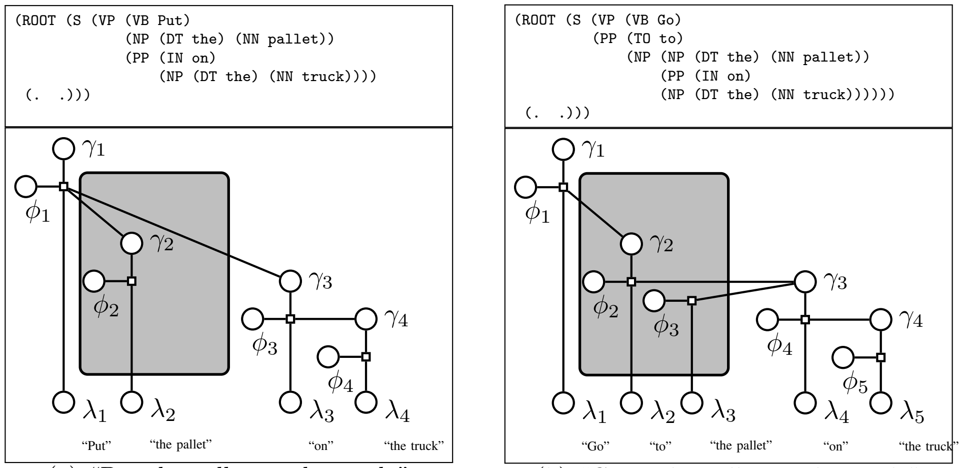
$$p(\Gamma | \Lambda, \Phi, M) = \prod_m p(\phi_m | \lambda_m, \gamma_{m_1} \ldots \gamma_{m_k}). \tag{5}$$

[3]All of the factors in this work were either entity factors or relational factors with arity 3 or less.

This factorization defines a probabilistic graphical model that constitutes the grounding graph. It is equivalent to Equation 3, except for a constant, but assumes that the factors take a specific, locally normalized form. Each factor $p(\phi_m|\lambda_m, \gamma_{m_1} \dots \gamma_{m_k})$ can be normalized appropriately — only over the domain of the ϕ_m variable for each phrasal constituent, substantially improving our ability to learn the model. Moreover, although each factor roughly corresponds to a probabilistic predicate, these predicates are defined by the linguistic structure. The arity of the predicate is defined by the number of children in the parse tree for that node. This representation is consistent with logical representations for word meanings used in the literature on semantics (Zettlemoyer and Collins, 2005a). The application of factors ψ results in set of scored groundings for each probabilistic predicate, as in Figure 2(e). Based on these scored candidates and the inferred linguistic structure, a grounding graph is created (Figure 2(d)) to represent the constraints that exist between the different groundings. During inference, a configuration of Γ is scored under the assumption that $\phi_i = 1$, which is equivalent to the notion of virtual evidence nodes (Bilmes, 2004) (Li, 2009) in a conditional random field. Our approach learns these predicates during training; because they are locally normalized they can be recombined through Equation 5 at test time in order to interpret commands that have not been previously encountered.

Formally, a grounding graph consists of random variables for the command $\Lambda = [\lambda_1 \dots \lambda_M]$, the grounding variables $\Gamma = [\gamma_1 \dots \gamma_M]$, and the correspondence variables $\Phi = [\phi_1 \dots \phi_M]$, along with a set of factors Ψ. Each factor $\psi \in \Psi$ consists of $\phi_m, \lambda_m, \gamma_{m_1} \dots \gamma_{m_k}$. Algorithm 1 specifies how random variables and factors are automatically created from the parse structure of a natural language command.

Because the factors themselves are learned, this graphical structure enables our approach to learn predicates for words such as "on", "to," "near" and "next to" from lower-level features, unlike in previous statistical approaches where these predicates must be provided by the system designer. These learned predicates can be reused in commands not seen during training. Furthermore, this model allows us to perform inference over similar sentences that have very different meaning and parse structure. For example, Figure 3 shows the grounding graphs for two different natural language commands. Figure 3(a) shows the parse tree and graphical model generated for the command "Put the pallet on the truck." The random variable ϕ_2 is associated with the constituent "the pallet" and the grounding variable γ_2. The random variable ϕ_1 is associated with the entire phrase, "Put the pallet on the truck" and depends on both the grounding variables γ_1 (the



(a) "Put the pallet on the truck." (b) "Go to the pallet on the truck."

Figure 3: Parse tree and induced model for two different commands. The shaded region shows where the factorization differs.

action that the robot takes) their arguments γ_2 (the object being manipulated) and γ_3 (the target location). The λ_i variables correspond to the text associated with each constituent in the parse tree. Figure 3(b) shows the parse tree and induced model for a different command, "Go to the pallet on the truck." Although the words are almost the same between the two examples, the parse structure is different, yielding a different graphical structure, highlighted in gray, and a different grounding.

4 Mobile Manipulation

Robots capable of manipulating objects can prove tremendously useful to human partners. Example platforms include a robotic forklift (Figure 4(a)), designed for outdoor, large-scale manipulation (Teller et al., 2010), and the PR2 (Figure 4(b)), designed for indoor, household tasks). For example, the autonomous forklift might be tasked with moving pallets in a supply depot in preparation for offloading or distribution. The PR2 might carry out tasks in the home, such as unpacking and putting away a shipment of household products. In both cases, a user might wish to specify where the objects should go, as in "Put the pallet in receiving," or "Hand me the stapler." This section describes how we use the G^3 framework to understand natural language commands for robots like these. We present a quantitative

Algorithm 1 Generating a grounding graph from natural language command. *args* is a function that determines the arguments to a relational linguistic constituent (usually this function just gets the children of λ_i). For arguments to the relation that are implicit (but necessary for understanding), we add implicit groundings γ_j.

Input:

Parsed natural language command, $\Lambda = \lambda_1 \ldots \lambda_M$, with root $\lambda_{root} \in \Lambda$. $|N|$ is the number entity factors. $M = |\Lambda|$.

$\Phi \leftarrow \phi_1 \ldots \phi_M$

$\Gamma \leftarrow \gamma_1 \ldots \gamma_N$

$\Psi \leftarrow []$

for $\lambda_i \in \Lambda$ **do**

    If λ_i **is entity:**

    Add $(\phi_i, \lambda_i, \gamma_i)$ to Ψ

    If λ_i **is relation:**

    $\Gamma_{factor} \leftarrow []$

    for each $\lambda_j \in args(\lambda_i)$ **do**

        Add γ_j to Γ_{factor}

    end for

    Add γ_j to Γ_{factor} for each implicit argument for λ_i

    Add $(\phi_i, \lambda_i, \Gamma_{factor})$ to Ψ

end for

Output: $\Lambda, \Phi, \Gamma, \Psi$

(a) Robotic forklift, operating in an outdoor supply depot.

(b) PR2, a mobile humanoid operating in indoor, household environments.

Figure 4: We demonstrated our command-understanding framework on two mobile-manipulation robots.

evaluation in simulation for the forklift domain, and demonstrate the end-to-end system on both the forklift and the PR2.

4.1 Modeling Word Meanings

The specific form for each factor $p(\phi_m|\lambda_m, \gamma_{m_1} \dots \gamma_{m_k})$ varies in our different application domains. For mobile manipulation, we assume that each factor in Equation 5 takes a log-linear form with feature functions s_j and feature weights θ_j:

$$p(\phi_m|\lambda_m, \gamma_{m_1} \dots \gamma_{m_k}) = \frac{1}{Z} \exp\left(\sum_j \theta_j s_j(\phi_m, \lambda_m, \gamma_{m_1} \dots \gamma_{m_k})\right). \tag{6}$$

Word meanings are represented as weights associated with these feature functions. The specific features capture the mapping between words in language and aspects of the external world and are described in more detail in Section 4.1. We optimize feature weights θ_j to maximize the likelihood of the training data. This function is convex and can be optimized with gradient-based methods, such as L-BFGS (Andrew and Gao, 2007; McCallum, 2002).

We train the system using data that consists of natural language commands together with positive and negative examples of groundings for each constituent in the command. The robot was given information about the environment, including all objects in the environment with their labels. For example, the forklift model was given information about all pallets in the environment, their tag as "pallet" as well as their location and geometry (as well as other objects such as the truck). We manually annotated the alignment between nouns in the corpus that corresponded to an object, place, path, or event sequence in the external world. By contrast, verb phrase grounding was automatically aligned with an agent path or event from the log associated with the original video. This automatic annotation substantially reduced the annotation effort for verb phrases, but was an approximation that admittedly led to inaccurate alignments for compound commands such as "Pick up the right skid of tires and place it parallel and a bit closer to the trailer," where each verb phrase refers to a different part of the state sequence. This lack of perfect alignment introduced noise into the training process. In the future we plan to explore automatically segmenting the trajectories and associating each part of the trajectory with the appropriate part of the verb phrase. However this problem is challenging because trajectories are continuous in time.

The annotations above provided positive examples of grounded language. In order to train the model, we also need negative examples. We generated negative examples by associating a random grounding with each linguistic constituent. Although this heuristic works well for verb phrases, ambiguous noun phrases such as "the pallet" or "the one on the right" are often associated with a different, but still correct, object (in the context of that phrase alone). For this reason, we manually corrected the negative noun phrase examples, reannotating some of them as positive.

We define binary features s_k for each factor. These features enable the system to determine which values for Γ correctly ground the corresponding linguistic constituent. Geometric "base" features enable the system to represent relations involving objects, places, paths, and events. For a relation such as "on," a natural geometric feature is whether the first argument is supported by the second argument, taking into account their geometric relationship. However, the base feature $supports(\gamma_i^f, \gamma_i^l)$ alone is not enough to enable the model to learn that "on" corresponds to $supports(\gamma_i^f, \gamma_i^l)$, because this feature is independent of the language. Instead we need a feature like $supports(\gamma_i^f, \gamma_i^l) \wedge \text{"on"} \in \lambda_i^r$. Thus, the system creates the Cartesian product of base features and the words in the corresponding linguistic constituents to compute features s_k. Base features that are continuous-valued functions such as $DistanceTo(\gamma_i)$ are discretized to create a set of binary base features.

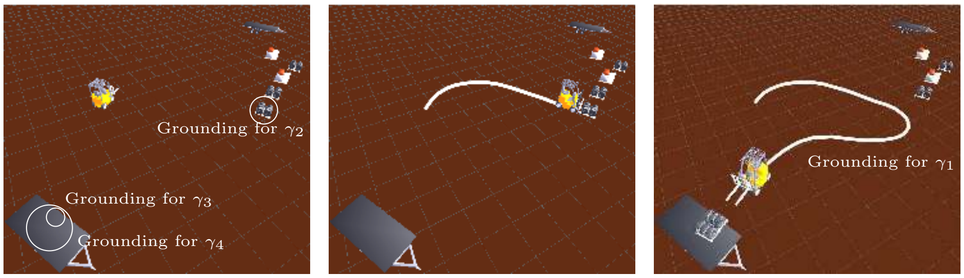
We manually implemented a set of base features that involve geometric relations between the groundings γ_i. Since groundings for prepositional and verb phrases correspond to the location and trajectory of the robot and any objects it manipulates over time, the feature functions for these linguistic constituents require as input the geometry of the robot, the object it is manipulating, as well as their trajectories through space. Examples include:

- The displacement of a path toward or away from a landmark object.
- The average distance of a path from a landmark object.

We also used the complete set of features described in Tellex et al. (2010), which capture concepts such as contact, and the relative geometric positions between two objects. We use 49 base features for leaf noun phrases (e.g., "the truck") and prepositional phrases (e.g., "on the truck"), 56 base features for compound noun phrases (e.g., "the pallet on the truck"), and 112 base features for verb phrases (e.g., "Pick up the pallet."). Automatically discretizing features and taking the Cartesian product with words from the training set produces 147,274 binary features. These features are defined in the Appendix A.

4.2 Inference

Given a command, we want to find the set of most probable groundings for that command. During inference, we search for groundings Γ that maximize the likelihood of the groundings for each linguistic constituent as in Equation 3. To limit the search space, the system uses a topological map of the environment that defines a limited set of salient objects and locations. In our experiments, the map size ranged between approximately three and twenty objects and locations. Using the topological map, we define a discrete state/action space for the robot, and search for sequences of actions corresponding to the grounding for verb phrases. Even though we discretize the search space, our optimization considers all permutations of object assignments as well as every feasible sequence of actions the agent might perform. As a result, the search space becomes large as the number of objects, paths, and events in the world increases. In order to make the inference tractable, we use beam search with a fixed beam width. For noun phrases and place-prepositional phrase groundings, the beam width was 10; for verb phrase and path-prepositional phrases, the beam width over candidate state sequences was 5. We determined the beam width empirically from data, finding the narrowest beam width that led to good performance. Figure 5 shows the results of inference for an example from the forklift domain; it consists of



(a) Object groundings. (b) Pick up the pallet. (c) Put it on the truck.

Figure 5: A sequence of the actions that the forklift takes in response to the command "Put the tire pallet on the truck." In (a) the search grounds objects and places in the world based on their initial positions. In (b) the forklift executes the first action and picks up the pallet. In (c) the forklift puts the pallet on the truck.

groundings for each linguistic constituent in the command along with an action trajectory for the robot.

4.3 Evaluation

We collected a corpus of mobile manipulation commands paired with robot actions and environment state sequences. We used this corpus to train the G^3 framework and also to evaluate end-to-end performance of the system at following realistic commands from untrained users. To collect commands, we posted videos of action sequences to Amazon's Mechanical Turk (AMT) and collected language associated with each video. The videos showed a simulated robotic forklift engaging in an action, such as picking up a pallet or moving through the environment. Paired with each video, we collected a complete log of the state of the environment and the robot's actions. Subjects were asked to type a natural language command that would cause an expert human forklift operator to carry out the action shown in the video. We collected commands from 45 subjects for twenty-two different videos. Each subject interpreted each video only once, but we collected multiple commands (an average of 13) for each video, for a total of 285 commands. The corpus contained 6149 words, with 508 unique words. Figure 6 shows example commands from our dataset, which the system can successfully follow.

Subjects were not primed with any example words or phrases from the domain describing the actions or objects in the video, leading to a wide

Go to the first crate on the left and pick it up.

Pick up the pallet of boxes in the middle and place them on the trailer to the left.

Go forward and drop the pallets to the right of the first set of tires.

Pick up the tire pallet off the truck and set it down.

Figure 6: Example commands from our corpus, which the system can successfully follow.

variety of natural language commands including nonsensical ones such as "Load the forklift onto the trailer," and misspelled ones such as "tailor" (trailer). Figure 1(a) shows commands collected using this methodology for one video in our dataset.

4.3.1 Model Evaluation

We annotated each constituent in the corpus with the corresponding grounding. Using the annotated data, we trained the model and evaluated its performance on a held-out test set. We did a single random training/test split with 70% in training and 30% in test. We used one set of videos (with associated commands) in the training set, and a different set in the test set. There was no development set used to validate hyperparameters. We assessed the model's performance at predicting the correspondence variable given access to words in the language and ground truth values for the grounding variables. The test set pairs a disjoint set of scenarios from the training set with language given by subjects from AMT. This process evaluates Equation 6 directly; Section 4.3.2 conducts an end-to-end evaluation.

Table 2 reports overall performance on this test set and performance broken down by constituent type. The performance of the model on this corpus indicates that it robustly learns to predict when constituents match groundings from the corpus. We evaluated how much training was required to achieve good performance on the test dataset and found that the test error asymptotes at around 1,000 (of 3,000) annotated constituents.

For noun phrases, correctly-classified high-scoring examples in the dataset include "the tire pallet," "tires," "pallet," "pallette [*sic*]," "the truck," and "the trailer." Low-scoring examples included noun phrases with incorrectly

Constituent type	Precision	Recall	F-score	Accuracy
Noun Phrase	0.93	0.94	0.94	0.91
Prepositional Phrase (Place)	0.70	0.70	0.70	0.70
Prepositional Phrase (Path)	0.86	0.75	0.80	0.81
Verb Phrase	0.84	0.73	0.78	0.80
Overall	0.90	0.88	0.89	0.86

Table 2: Performance of the learned model at predicting the correspondence variable ϕ.

annotated groundings that the system actually got right. A second class of low-scoring examples arose due to words that appeared rarely in the corpus.

For place prepositional phrases, the system often correctly classified examples involving the relation "on," such as "on the trailer." However, the model often misclassified place prepositional phrases that involve frame-of-reference. For example, "just to the right of the furthest skid of tires" requires the model to have features for "furthest," which requires a comparison to other possible objects that match the phrase "the skid of tires." Understanding "to the right" requires reasoning about the location and orientation of the agent with respect to the landmark object. Similarly, the phrase "between the pallets on the ground and the other trailer" requires reasoning about multiple objects and a place prepositional phrase that has two arguments. The model is not capable of interpreting phrases like "the first" or "second" crate on the left, because it cannot handle subsets or groups of objects. (Although our more recent work has addressed this limitation (Paul et al., 2016).)

For verb phrases, the model generally performed well on "pick up," "move," and "take" commands. The model correctly predicted plans for commands such as "Lift pallet box," "Pick up the pallets of tires," and "Take the pallet of tires on the left side of the trailer." It predicted incorrect plans for commands like, "move back to your original spot," or "pull parallel to the skid next to it." The word "parallel" appeared in the corpus only twice, which was apparently insufficient to learn a good model. "Move" had few good negative examples, since we did not have in the training set contrasting examples of paths in which the forklift did not move.

4.3.2 End-to-end Evaluation

To evaluate end-to-end performance, the system inferred plans given only commands from the test set, a starting location for the robot and a map of the environment with the location and type of objects located in the map. We segmented commands containing multiple top-level verb phrases into separate clauses. Next, the system used the generated grounding graph to infer a plan and a set of groundings for each clause. We simulated plan execution on a realistic, high-fidelity robot simulator from which we created a video of the robot's actions. We uploaded these videos to Amazon Mechanical Turk (AMT)[4] where subjects viewed each video paired with a command and reported their agreement with the statement, "The forklift in the video is executing the above spoken command" on a five-point Likert scale. We report command-video pairs as correct if the subjects agreed or strongly agreed with the statement, and incorrect if they were neutral, disagreed or strongly disagreed. We collected five annotator judgments for each command-video pair.

To validate our evaluation strategy, ensure that the collected data is interpretable and that a Likert evaluation metric can correctly characterize good and bad grounding performance, we gave known correct and incorrect command-video pairs to subjects on AMT. In the case of known correct grounding, subjects saw videos with known-correct commands that were generated by other subjects. In the case of known incorrect grounding, the subject saw the command paired with a random video that was not used to generate the original command. Table 3 depicts the percentage of command-video pairs deemed consistent for these two conditions. As expected, there is a large difference in Likert score between commands paired with the original and randomly selected videos, validating our approach to evaluation. Additionally, these results show that commands in the corpus are generally understandable by a different annotator but that some people did give commands that other people found difficult to follow.

We then evaluated our system by considering three different configurations. Serving as a baseline, the first experimental evaluation consisted of ground truth parse trees and a cost function which selected actions at random. The second configuration involved ground truth parse trees, and a learned cost function that selects the best action to follow the commands. The third consisted of automatically extracted parse trees and a learned cost function for selecting the best action.

Table 3 reports the performance of each configuration along with their

[4]www.mturk.com

Scenario	% Correct
Commands paired with original video	$91\% \pm 1\%$
Commands paired with random video	$11\% \pm 2\%$
Annotated parses (high conf.), learned cost	$63\% \pm 8\%$
Automatic parses (high conf.), learned cost	$54\% \pm 8\%$
Annotated parses (all), learned cost	$47\% \pm 4\%$
Annotated parses (all), random cost	$28\% \pm 5\%$

Table 3: The fraction of commands considered correct by our annotators for different configurations. High conf. evaluated in the top 30 most probable commands according to the model. We report 95% confidence intervals.

95% confidence intervals. We evaluated on all commands in the test set. Additionally, as a second trial, we report results on a subset of commands that were ranked according to the model's own confidence score (*high conf*). The relatively high performance of the random cost function configuration relative to commands paired with random videos demonstrates the inherent knowledge captured by the discretized state/action space. However, in all conditions, the system performed statistically significantly better than a random cost function. The system qualitatively produced compelling end-to-end performance. Even when it made a mistake, it often correctly followed parts of the command. For example, it sometimes picked up the left-hand tire pallet rather than the right-hand pallet. Other types of errors arose due to ambiguous or unusual language, such as "remove the goods" or "the lonely pallet."

The system performed noticeably better on the high confidence commands than on the entire test set. This result indicates the validity of our probability measure, suggesting that the system had some knowledge of when it is correct and incorrect. We have demonstrated that the system can use this information to decide when to ask for confirmation before acting (Tellex et al., 2013; Deits et al., 2013).

4.3.3 Real-world Demonstration

Finally, we demonstrated the end-to-end system on two platforms: a robotic forklift and the PR2 mobile manipulator. The robotic forklift, described in detail by Teller et al. (2010), is an autonomous robotic vehicle capable of driving through real-world warehouse environments. It can localize itself, avoid obstacles, track people, recognize objects, and move pallets. Using the models described in the previous section, we demonstrated that it can follow commands such as "Put the tire pallet on the truck" and "Pick up the tire pallet." Figure 7(a) shows scenes from a video of the forklift executing the command "Put the tire pallet on the truck." Audio was captured using a headset microphone and converted to text using the SLS recognizer (Glass, 2003). Next, the system parsed the text with the Stanford Parser (Marneffe et al., 2006) and extracted the graphical model structure using Algorithm 1. The robot had previously been given a tour of the environment that provided a model of the visual appearance and label of salient objects, such as the tire pallet and the truck, as described by Walter et al. (2012). Finally, it carried out the inference in Equation 1 to infer an action sequence and executed the action. See `http://youtu.be/JyDRXOhr3b0` and `http://youtu.be/OzWTyH4nGIc` for videos.

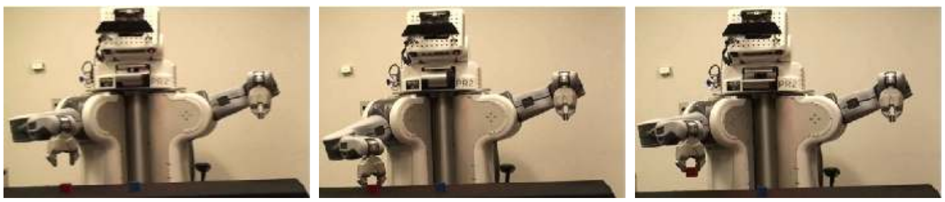
Using the PR2, we demonstrated command-following in a simple blocks-world domain. We used the Kinect sensor to detect the plane of the table and segment the locations of blocks. We collected a small training corpus of natural language commands to train the model. The robot recognized speech using Google's speech recognizer, extracted the graphical model from the text, and inferred groundings, including an action for the robot. We used `rosbridge` (Crick et al., 2011) to connect between the different systems. Scenes from the robot executing a command appear in Figure 7(b); see `http://youtu.be/Nf2NHlTqvak` for video.

Our real-world demonstrations tended to use simpler commands than the corpus because of the difficulty and expense creating real-world test environments.

5 Route Directions

In the previous section, we tested our system at following mobile-manipulation commands on single verb phrases. However, in many task domains, it would be useful to understand longer movement commands. For example, Figure 8(a) shows a robotic wheelchair; understanding natural language movement commands would enable a person to use language to control the chair, even if they were unable to control it with a conventional interface. Figure 8(b) shows a robotic micro-air vehicle (MAV), which can engage in inspection tasks as well as search-and-rescue operations. Specifying a three-dimensional trajectory for such a vehicle using conventional interfaces is challenging for untrained users. If instead, a user could speak a natural language command describing where s/he wanted the vehicle to go, it would

(a) Photographs of the forklift executing the command, “Put the tire pallet on the truck.”



(b) Photographs of the PR2 executing the command, “Pick up the red block.”

Figure 7: Imagery from the PR2 and forklift following natural language commands using the G^3 framework.

enable higher-level, intuitive interaction. Figures 1(b) and 1(d) show examples of these types of longer instructions.

As can be seen in these examples, typical route directions can result in a long sequence of clauses that the robot must follow, making the inference problem more challenging. In order to infer the optimal set groundings (e.g., paths, landmarks) for the sequential structure of route directions, inference must consider vastly more paths through the environment than in Section 4 because commands are substantially longer. To enable efficient and robust inference, we make an approximation to the full hierarchical model by creating a fixed, flat linguistic structure. This structure not only enables the search space over paths to become more tractable, we also found that in practice the flat model is more effective at capturing the meaning of route directions because they better tolerate discrepancies in object groundings. For example, if a robot is told “Go to the lounge with couches” and it maps the noun phrase “the lounge with couches” to the couches instead of the lounge, it will probably perform roughly the right action when it tries to go there. Note that this kind of approximation generally does not hold for mobile manipulation: if the person says, “Pick up the tire pallet near the box pallet” and the robot picks up the box pallet instead of the tire pallet, then it has not accurately followed the command. This section describes how we perform efficient inference in a flattened model to enable a robot to

(a) Robotic wheelchair. (b) Robotic quad-rotor helicopter.

Figure 8: Two robots used in our direction following work.

understand longer sequences of natural language movement commands.

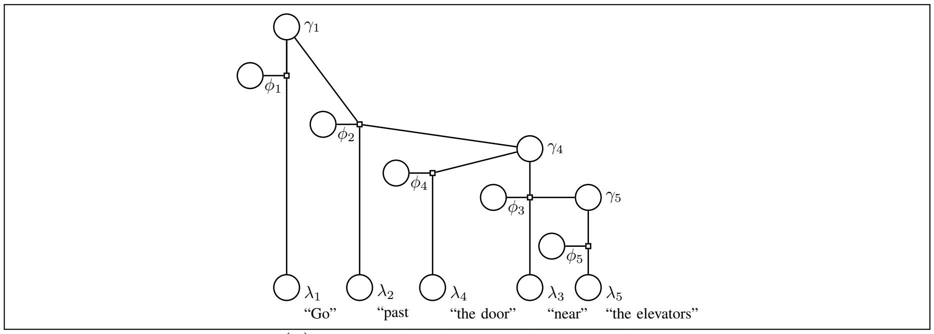
5.1 Modeling Word Meanings

To generate a grounding graph with a simplified structure, we use a flattened parse tree extracted using a CRF chunker (Kudo, 2009). Figure 9 shows a comparison of the flattened and full hierarchical models for the same sentence. The flattened structure enables much more efficient inference using a variant of the Viterbi algorithm for multiple sentences, so that the system can quickly infer the actions implied by a paragraph-length set of route instructions.

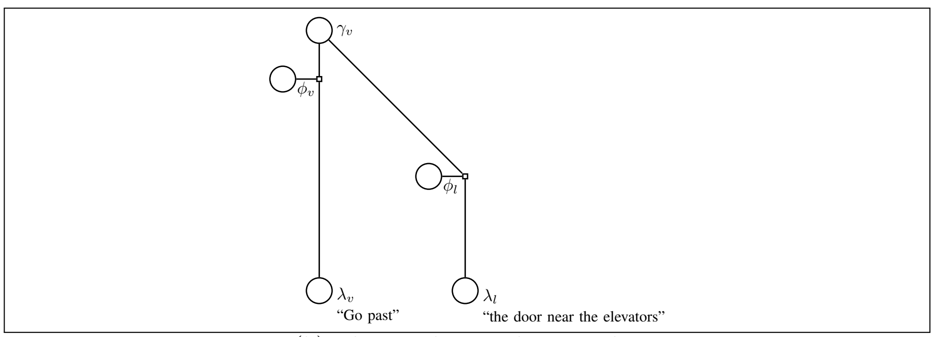
We assume that the command consists of a sequence of S phrases, where each phrase breaks down into linguistic constituents (λ_v, λ_l) and corresponding groundings (γ_v, γ_l). This structure leads to efficient inference algorithms for finding the optimal trajectory, described in the following section. Adapting the framework described in Section 3 to this more simplified structure, there is a correspondence variable for the landmark ϕ_v and one for the landmark ϕ_l, such that:

$$p(\Gamma|\Lambda, \Phi, M) = \prod_s p(\phi_v|\lambda_v, \gamma_t) \times p(\phi_l|\lambda_l, \gamma_t). \tag{7}$$

This factored form is equivalent to Equation 5 but is explicit about how factors are constructed from the command. By using a fixed factorization, we sacrifice linguistic flexibility in order to simplify model learning and obtain efficient inference algorithms that exploit the repeating sequential structure



(a) Hierarchical grounding graph.



(b) Flattened grounding graph.

Figure 9: Hierarchical and flattened grounding graphs for the sentence, "Go past the door near the elevators."

of the factorization. The grounding variable γ_t corresponds to the trajectory associated with this path segment, represented as T points $s_1 \dots s_T$. Note that the landmark factor ϕ_l connects directly to γ_t without an intermediary variable γ_l.

Because route directions generally use a restricted set of movement actions but a more open-ended set of landmark objects, we use different models for each factor in the grounding graph. We connect the landmark factor directly to the trajectory and estimate the probability that an object corresponding to the landmark phrase can be seen from the endpoint of the trajectory using object-object co-occurrence statistics mined from a large dataset (described in Section 5.1.2). In addition, we model how well the trajectory corresponds to a verb phrase in the command (described in Section 5.1.1). Modeling more of the linguistic structure, such as spatial relations, requires that the landmark be accurately grounded to a specific object in the robot's model of the environment; our previous work showed that explicitly modeling spatial relations did not contribute significantly to overall performance (Kollar et al., 2010b).

5.1.1 Verbs

To compute $p(\phi_v|\lambda_v, \gamma_t)$, that is, to ground verb phrases, we use a three-state model that classifies a verb phrase λ_v as either "left," "right," (if λ_v contains the word "left" or "right") or "straight" (otherwise). We define the verb features precisely in the Appendix A. This approach is able to interpret verbs such as "Go left," "go right," "move forward" and the like.

5.1.2 Noun Phrases

Modeling noun phrases is challenging because people refer to a wide variety of objects in natural language directions and use diverse expressions to describe them. In our corpus, people used more than 150 types of objects as landmarks, ranging from "the door near the elevators" to "a beautiful view of the domes." To compute $p(\phi_l|\lambda_l, \gamma_l)$, that is, to ground landmark phrases, the system takes a semantic map seeded with the locations of known objects and uses object-object context to predict the locations of the unknown landmark terms, following Kollar and Roy (2009). Object-object context allows the system to predict that a computer is nearby if it can directly detect a monitor and a keyboard, even if it cannot predict the computer's exact location and geometry. To predict where a novel landmark may occur, we downloaded over a million images, along with their associated labels from the photo-sharing website Flickr.

We computed the probability that a particular word $w_j \in \lambda_l$ applies to the trajectory γ_t, given the detected object labels associated with the trajectory $O(\gamma_t)$:

$$p(\phi_l|\lambda_l, \gamma_t) = p(\phi_l|w_1 \ldots w_J, O(\gamma_t)). \tag{8}$$

Next, we rewrite the equation using Bayes' rule, assuming the words $w_j \in \lambda_l$ are independent:

$$p(\phi_l|w_1 \ldots w_J, O(\gamma_t)) = \frac{\prod_j p(w_j|\phi_l, O(\gamma_t))p(\phi_l|O(\gamma_t))}{p(w_1 \ldots w_J|O(\gamma_t))} \tag{9}$$

This "bag of words" assumption is not strictly true but simplifies training and inference. The denominator can be rewritten without $O(\gamma_t)$, since the two terms are independent when ϕ is not known:

$$p(\phi_l|w_1 \dots w_J, O(\gamma_t)) = \frac{\prod_j p(w_j|\phi_l, O(\gamma_t))p(\phi_l|O(\gamma_t))}{p(w_1 \dots w_J)} \tag{10}$$

We assume the priors are constant and, therefore, do not consider them in the inference process. For brevity, we drop ϕ_l; it is implicitly assumed $True$.

We estimate the distribution $p(w_j|\phi_l, O(\gamma_t))$ as a multinomial distribution using Naive Bayes. First we rewrite with Bayes' rule:

$$p(w_j|O(\gamma_t)) = \frac{p(O(\gamma_t)|w_j) \times p(w_j)}{p(O(\gamma_t)|w_j)p(w_j) + p(O(\gamma_t)|\neg w_j)p(\neg w_j)} \tag{11}$$

Next, we make the assumption that $O(\gamma_t)$ consists of labels, $o_1 \dots o_K$, which are independent:

$$p(w_j|o_1 \dots o_K) = \frac{\prod_k p(o_k|w_j) \times p(w_j)}{\prod_k p(o_k|w_j)p(w_j) + \prod_k p(o_k|\neg w_j)p(\neg w)} \tag{12}$$

We compute the set of labels using the objects visible at the end of the trajectory γ_t. We found that most of our directions referred to landmarks that were visible at the end of the trajectory. Even for phrases such as "Go through the set of double doors," where the doors are located in the middle of the trajectory, they are visible from the end, so using this set of objects works well in practice. However, this assumption may be violated for much longer directions.

We estimate these probabilities using co-occurrence statistics from tags from over a million images downloaded from the Flickr website (Kollar and Roy, 2009). For example, using this corpus, the system can infer which bedroom is "the baby's bedroom" without an explicit label, since only that room contains a crib and a changing table. This data allows the robot to interpret language about landmarks that may not be in its semantic map by connecting them through co-occurrence to landmarks that are present in the semantic map. Specifically we compute the following estimates, where *count* is defined as the number of captions that contain word w in the Flickr corpus:

$$p(o_k|w_j) = \frac{count(o_k, w_j)}{count(w_j)} \tag{13}$$

And without w_j:

$$p(o_k|\neg w_j) = \frac{count(o_k, \neg w_j)}{count(\neg w_j)} \tag{14}$$

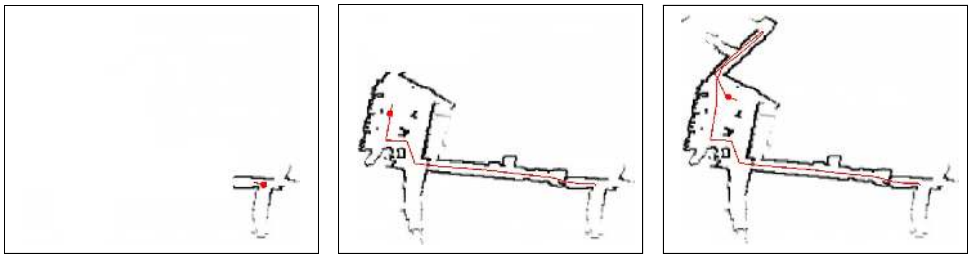
We refer to the above method as the Naive Bayes model for landmark factors. Using this approximation leads to problems because many words in the context $o_1 \ldots o_K$ are not relevant to a particular landmark phrase. For example, if the robot is going to "the kitchen," observations of the refrigerator and microwave will help it identify a promising candidate region, but observations of a door will not give much additional information. Incorporating these objects into the approximation in Equation 12 causes it to underestimate the true probability of encountering a landmark phrase. To compensate, we use the subset of $o_1 \ldots o_k$ that has the highest probability when estimating $p(\phi_l|w_1 \ldots w_J, o_1 \ldots o_K)$:

$$p(\phi_l|\lambda_l, \gamma_t) \approx \max_{O \in powerset(\{o_1 \ldots o_K\})} p(\phi_l|w_1 \ldots w_J, O). \tag{15}$$

We estimate this probability over the powerset of all objects observed at a particular location. This computation is tractable because of the limited numbers of object types available in the semantic map; typically no more than ten different types of objects will be visible at a particular location. We refer to this version as the salient object model because it finds the subset O that provides the strongest evidence for ϕ_l. We are able to compute this term exactly because at any particular location, relatively few unique objects are visible, so the cardinality of the powerset of visible tags is small.

5.2 Inference

Given the graphical model for the natural language command, the inference algorithm for following directions through completely known maps (*global* inference) searches through all possible paths in a topological map of the environment to find the maximum of the distribution in Equation 1. Global inference is performed using a dynamic programming algorithm (Viterbi, 1967) that finds the most probable trajectory corresponding to a set of natural language directions. The algorithm takes as input a set of starting locations, a map of the environment with some labeled objects as well as a topology, and the graphical model created from the parsed directions. It outputs a sequence of nodes in the topological map, corresponding to trajectory through the environment.



(a) t=0 (b) t=50 (c) t=100

Go through the double doors and past the lobby. Go into the lounge with some couches. Enjoy the view over there. Go past the spiral staircase. Continue towards the hall with some cubby holes but don't go down the hallway. Instead take a right into the kitchen.

(d) Command

Figure 10: Explored map at three different phases the robot explores the environment. Black shows obstacles/walls visible explored by the robot. The red circle shows the robot's current location, and the red line shows its trajectory. The system misunderstands the negation in the phrase "don't go down the hallway," but backtracks when it does not find a kitchen. After backtracking, it reaches the correct destination. The robot's current location and past trajectory are shown in red.

Following route directions with a complete map is useful for scenarios where a robot might be able to perform both exploration and mapping beforehand, such as in a home or building. For less structured environments or new environments where exploration and mapping cannot be performed beforehand, a robot must follow commands with incomplete information. In this more general scenario, the robot must follow directions without a pre-built map, discover the environment as it navigates and update its plans when it discovers new information. We present two different approaches to address the scenario where the map is not known beforehand. The first follows the directions step by step, exploring the map as it goes. At each step, it chooses the next action based on the best available action at its current location. If it makes a wrong turn, it does not backtrack or explore. When the robot reaches a region not in the existing map, it explores this region, incrementally growing its map of the environment. We refer to this algorithm as *greedy local inference*. The second algorithm is similar, but if there is no transition at the current location with probability above a threshold, it backtracks and explores a different region. We refer to this algorithm as *exploring local inference*. Figure 10 shows the explored map at two different phases of the exploring local inference algorithm. We expect global inference to perform better because it searches through all possible paths to find the one that best matches the descriptions. However, the local inference algorithms are more practical because they do not need a complete map of the environment to follow directions.

The system creates a topological roadmap from the gridmap of the environment, then searches for a path within this graph. The roadmap is created by segmenting spaces based on visibility and detected objects and then extracting a topology of the environment from this segmentation, building on techniques described by Brunskill et al. (2007). As a robot path extends through each of the nodes, it may take on any of the four cardinal directions, which leads to connections in the topological map that include the Cartesian product of the original topological map connections and the four cardinal directions. This enables the system to use features of the orientation of the robot along a path to differentiate the next correct action. For example, "turn right" might be differentiated from "go straight" only by the fact that the orientation at the end of the path differs by approximately 90° across the two choices. For the wheelchair domain, we assume that the robot moves in two dimensions. For the MAV, we create a two-level topological map by duplicating all nodes at each level, enabling it to fly through a three-dimensional space.

5.3 Evaluation

To evaluate our system, we collected corpora for two application domains: the robotic wheelchair, and the robotic MAV. We also demonstrated the system end-to-end on these two robotic platforms.

5.3.1 Corpus-Based Evaluation

For the wheelchair, we collected a corpus of route directions between two locations in an indoor office environment. Subjects wrote directions as if they were directing another person through the space. We collected directions through two environments. Environment 1 is a work area with a computer lab and offices ($133\,\mathrm{m} \times 137\,\mathrm{m}$), while Environment 2 is an atrium with a cafe and classrooms ($99\,\mathrm{m} \times 62\,\mathrm{m}$). We asked fifteen subjects to write directions between 10 different starting and ending locations in environment 1, and another fifteen to write directions between 10 different location pairs in environment 2, for a total of 300 directions. The corpus from Environment 1 consisted of 8799 words and 939 unique words; the corpus from Environment 2 consisted of 6467 words and 921 unique words.

Experimenters did not refer to any of the areas by name, but instead used codes labeled on a map. Subjects were from the general population of MIT, between the ages of 18 and 30 years old, were proficient in English, were unfamiliar with the test environment, and were approximately of equal gender (47% female and 53% male subjects). Sample commands from the corpus appear in Figure 1(b).

For the MAV domain, users were familiarized with the test environment (which was the same as Environment 2) and were asked to instruct the pilot to take video of seven objects in the environment, each starting from a different location. Objects to be inspected were difficult to see closely from the ground and included a wireless router mounted high on the wall, a hanging sculpture, and an elevated window. Subjects were told the vehicle's initial pose and were asked to write down instructions for a human pilot to fly a MAV to the desired object and take video of that object. The corpus consists of forty-nine natural language commands, for a total of 2576 words and 467 unique words. Subjects were engineering undergraduates unfamiliar with the system. Figure 1(d) shows an example set of directions from this corpus.

We evaluate the global and local inference methods, as well as several baseline methods. Our first baseline is human performance at following directions. To measure human performance, we had one person follow each of the directions in this corpus, asking them to notify the experimenter if they became lost or confused. In this situation, we found that human performance at following directions in this dataset was 85% and 86% for the two datasets (e.g., 15% of the directions were not followed correctly by people). Qualitatively, we observed that people would sometimes give incorrect directions (e.g., they would say "right" when they meant "left"), or that their spatial ability in following directions appeared to be poor. We did not assess human performance for the MAV domain because of the difficulty of having untrained users follow an aerial trajectory. We implemented two computational baselines. The first, *Random*, is the distance between the true destination and a randomly selected viewpoint in the map. This estimates the complexity of the environment without taking into account the language input and provides a lower bound on performance. The second, *Last Phrase*, uses the location that best matches the last phrase in the directions according to our model, ignoring the rest of the directions. This provides a lower bound that shows whether the model is able to leverage information from the entire natural language command. We present all baselines using the Naive Bayes model for the landmark factor (Equation 12), as well as the salient object model (Equation 15).

Table 4 shows results for the wheelchair. We present the fraction of commands successfully followed to within 10 meters of the true destination. Our global inference model performed the best because it has complete access to the map, demonstrating the capability of the model with complete information. The fact that G^3 (Global Inference) outperformed the *Last Phrase* baseline demonstrates that our method is successfully applying information from the whole set of natural language directions, rather than just the last phrase. We found that performance was better in Environment 1 than in Environment 2 because the former has a simpler topology and more distinct landmarks. We also see that the salient object model for the landmark factor significantly outperformed the Naive Bayes model. The salient object model provides a filter that removes irrelevant terms from the Naive Bayes approximation, so that the algorithm only uses terms known to be relevant to the landmark phrase given by the language. To measure the effect of parsing on our results, we tested the salient object model using ground truth parses. Because of the fixed, simple parse structure, the parsing accuracy did not matter as much, and we observed a small effect on overall performance when using ground truth parses.

The greedy local inference method performed very poorly; in contrast, the exploring local inference method achieved performance competitive with the global inference algorithm. Table 5 shows the fraction of the environment explored by the local inference algorithms. We report the fraction of topological map nodes visited by the algorithm, excluding nodes on the shortest path between the start of the trajectory and the end. This metric represents extra exploration by the algorithm as it followed the directions. The greedy local inference algorithm explored a small part of the environment, but also achieved low performance compared to the global inference algorithm, which had access to the entire map. In contrast, the exploring local inference algorithm achieved performance competitive with global inference without visiting the entire map first.

Table 6 shows results for the MAV domain. We see the same high-level patterns as in the wheelchair domain: the salient object model outperformed the Naive Bayes model, and global inference outperformed greedy local inference, while exploring local inference was competitive with global inference. Table 7 shows the fraction of explored environment by the local inference algorithms. This consistent pattern suggests our results will likely generalize to different domains. In addition, we compared the performance of the

	Environment 1 % correct	Environment 2 % correct
Human Performance	85%	86%
Random	0%	0%
Naive Bayes (Equation 12)		
Last Phrase only	40%	25%
G^3 (greedy local inference)	30%	20%
G^3 (exploring local inference)	39%	27%
G^3 (global inference)	39%	27%
Salient Objects (Equation 15)		
Last Phrase only	50%	33%
G^3 (greedy local inference)	30%	20%
G^3 (exploring local inference)	71%	54%
G^3 (global inference)	71%	57%
G^3 (global inference, annotated parses)	67%	59%

Table 4: Performance at following directions to within 10 meters of the true destination in our two test environments, for directions given to a robotic wheelchair.

two dimensional model (identical to the wheelchair domain) to a three dimensional model, where the robot must infer a three-dimensional trajectory through the environment, including heights. We found that the 3D model outperformed the 2D model, suggesting that the 3D model better matches language produced in this domain, quantitatively demonstrating that the 3D structure improves performance.

5.3.2 Real-world Demonstration

We evaluated the G^3 framework end-to-end at following directions on the wheelchair (moving in two dimensions) and a robotic MAV (moving in three dimensions). Our autonomous wheelchair, shown in Figure 8(a), is equipped with laser range scanners for obstacle sensing, navigation, and localization. We initialized it with a semantic map of the environment, in which we labeled the locations of known objects. Given a typed or spoken command, it inferred a trajectory through the environment using the G^3 framework. Once the trajectory was inferred, the vehicle executed the trajectory autonomously. Figure 12 shows photos of the wheelchair as it follows natural language commands; see `http://youtu.be/yLkjM7rYtW8` for video.

t

	Environment 1	Environment 2
Naive Bayes (Equation 12)		
G^3 (greedy local inference)	2%	2%
G^3 (exploring local inference)	62%	61%
Salient Objects (Equation 15)		
G^3 (greedy local inference)	1%	2%
G^3 (exploring local inference)	77%	70%

Table 5: Fraction of the environment explored for each algorithm.

Figure 11: (top) Photographs of the MAV executing an interactive series of instructions. (bottom) Imagery from the on-board camera, transmitted to the operator as the MAV flies.

Our MAV, shown in Figure 8(b), is the AscTec Pelican quad-rotor helicopter, manufactured by Ascending Technologies GmBH. We outfitted the vehicle with both LIDAR and camera sensors, which allows us to obtain accurate information about the environment around the vehicle. In previous work (Bachrach et al., 2009) we developed a suite of sensing and control algorithms that enable the vehicle to explore unstructured and unknown GPS-denied environments. Here, we leverage that system to localize and control the vehicle in a previously explored, known environment (He et al., 2008; Grzonka et al., 2009). We developed an interface that enabled a person to give directions to the MAV using a speech recognition system or by typing a textual string. Paths computed by the G^3 framework were then executed autonomously by the MAV. Figure 11 shows scenes from the MAV

	Environment 2 % correct
Human Performance	–
Random	0%
Naive Bayes (Equation 12)	
Last Phrase only	20%
G^3 (greedy local inference)	0%
G^3 (exploring local inference)	20%
G^3 (global inference)	22%
Salient Objects (Equation 15)	
Last Phrase only	33%
G^3 (greedy local inference)	0%
G^3 (exploring local inference)	55%
G^3 (global inference, 2D)	55%
G^3 (global inference, 3D)	65%

Table 6: Performance at following directions to within 10 meters of the true destination in our environments for directions given to a robotic MAV.

following a set of directions; see `http://youtu.be/7nUq28utuGM` for video of the end-to-end system. Directions that our method handled successfully on the robotic platform include:

(a) "Go past the library and tables till you see a cafe to the left. Fly past the cafe and there will be other eateries. Head into this area."

(b) "Stand with your back to the exit doors. Pass the cafe on your right. Make a right directly after the cafe, and into a seating area. Go towards the big question mark."

(c) "Go straight away from the door that says CSAIL, passing a room on your right with doors saying MIT Libraries. Turn left, going around the cafe and walk towards the cow."

(d) "Turn right and fly past the libraries. Keep going straight and on the left near the end of the hallway there is a set of doors that say Children's technology center. You are at the destination."

(e) "Fly to the windows and then go up."

Our robotic experiments demonstrate that our approach is practical for

	Environment 2 % explored
Naive Bayes (Equation 12)	
G^3 (greedy local inference)	1%
G^3 (exploring local inference)	49%
Salient Objects (Equation 15)	
G^3 (greedy local inference)	1%
G^3 (exploring local inference)	49%

Table 7: Performance at following directions to within 10 meters of the true destination in our environments for directions given to a robotic MAV.

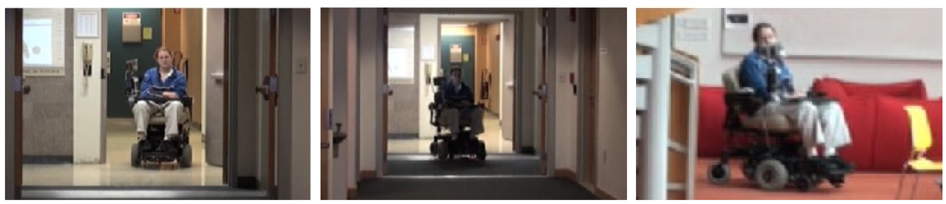
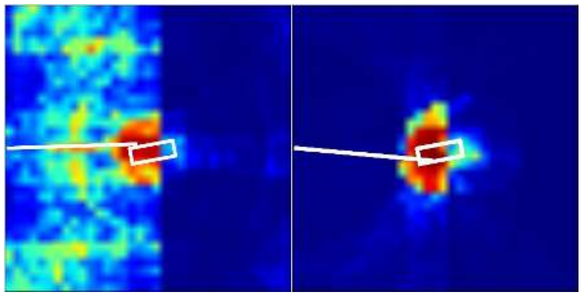


Figure 12: Photographs of the wheelchair following the command, “Go through the double doors, past the lobby. Go into the lounge with some couches. Enjoy the view. Go past the spiral staircase. Continue towards the cubby holes. Don’t go down the hallway. Instead take a right into the kitchen.”

realistic scenarios. The G^3 framework can quickly and robustly infer actions to take in response to natural language commands.

6 Discussion

Our approach is a significant step forward in terms of advancing robotic language understanding, however significant steps forward remain. As previously noted, our approach enables the robot to learn probabilistic predicates for words such as “to” and “near.” It is instructive to explore qualitatively what is learned. Figure 13 shows a heatmap generated by querying the mobile manipulation and route direction model for the phrase “to the truck.” Each cell in the heat map is generated by assessing the value of the $p(\phi|\text{“to”}, \gamma_{truck}, \gamma_{path}, M)$ for a straight line path that starts at the left side of the heat map and goes to that pixel. Red corresponds to high probability



(a) Mobile manipulation. (b) Route directions.

Figure 13: Heat map generated for the phrase “to the truck” starting at the left side of the scene. The truck is indicated with the white rectangle.

and blue corresponds to low probability. The highest probability path is drawn in white, and the truck’s geometry (viewed from the top) is the white rectangle. Both models learn reasonable models for “to”; the hot spots are close to the landmark of the truck. Note that these probabilistic predicates were learned automatically by the system from lower-level features without providing a geometric definition for the word “to.”

A significant limitation to our approach is that it must be adapted to the domain it is being used on. At minimum this adaptation requires training data including language and groundings from the specific domain. Sometimes additional changes might be necessary, such as a parser that is capable of handling language in the specific domain, as well as specific features that capture the semantics of words in the domain. Features are currently constructed manually; with new advances in deep learning it is intriguing to consider automatic feature construction. However a key question is to define the input representation - if the robot has already detected and localized objects, then the deep learning representation must learn probabilistic predicates over a different space from the low-level image space.

Our approach could be extended to other languages given data, but it requires adding additional features specific to those languages. For example, Korean distinguishes between two forms of the English preposition “in,” one for when the objects are tight-fitting, and a second when they are loose fitting (Norbury et al., 2008). Our approach would need additional features to enable the framework to learn this concept.

An additional limitation is the parsers that we are building on. Our approach used existing technology to parse the sentences. Using parsers trained on external datasets has potential because it can lead to increased performance by leveraging the external datasets. However domain mismatches can also cause problems. For example, the Stanford parser often parsed phrases such as "down the hall" incorrectly because the word "down" is most often used as an adverb in the training set. It would be useful to retry these experiments with more recent parsers which might not suffer from these deficiencies and to adapt parsers to our domains. Another approach, which others have taken, is to jointly train a parser along with semantic understanding modules (Krishnamurthy and Kollar, 2013; Matuszek et al., 2012b; Artzi and Zettlemoyer, 2013). These approaches require predicates to be provided by the system designer, but allow the parser to adapt to domain specific utterances and language, including imperative commands which are not often found in newswire text. Krishnamurthy and Kollar (2013) extends Matuszek et al. (2012a) to include relational predicates.

A key challenge in our real-world demonstrations was obtaining an accurate and complete world model for the robot. In the case of the robotic forklift, there were many objects in the environment that were not in the robot's world model (such as a pile of dirt and parked cars) and thus could not be referred to using language. This problem is pervasive in robotics: people want to talk to the robot about everything they can see, but the robot does not have a model of everything the person can see, due to limitations in its sensors and perceptual abilities.

7 Conclusions

In this paper, we have taken steps toward robust spatial language understanding using the G^3 framework. Our approach learns grounded word meanings that map between aspects of the external world. The G^3 framework defines a probabilistic graphical model dynamically according to the compositional, hierarchical structure of language, enabling word meanings to be combined to understand novel commands not present in the training set. We have demonstrated that the G^3 framework can model word meanings for a variety of mobile-manipulator robots, such as a forklift, the PR2, a wheelchair, and a robotic MAV. Our more recent work has shown how the probabilistic framework described here can be trained with less supervision (Tellex et al., 2013). We have also adapted it to ask targeted questions when the robot is confused by measuring entropy of the marginal distributions over specific grounding variables (Deits et al., 2013). Knepper et al. (2013) demonstrate how to invert the model, searching for language that corresponds to groundings, rather than groundings that match the language, in order to generate natural language requests for help.

The G^3 framework is a step toward robust language understanding systems, but many challenges remain. A key limitation is the requirement to define the search space for values of the grounding variables. This search space must be manually defined and tuned, because if it is too large, the search process is intractable, and if it is too small, then it is impossible to understand the command. For example, in the mobile manipulation domain, some commands referred to "the empty space" in a row of pallets, referring to an object with no physical instantiation at all. This problem is particularly important when understanding longer commands; each linguistic constituent which must be grounded increases the size of the search space during inference. Defining the search space itself dynamically based on the language, and searching it efficiently, remain open problems. Another remaining challenge is the acquisition of generalizable world meanings. In this paper we learned word meanings from large datasets, but we used word meanings tailored to each domain, and we have not demonstrated that those learned meanings generalize to different domains. In some cases, word meanings dynamically create a visual classifier, as in "Cook until the cornbread is brown," which requires recognizing when an object, which is changing over time, takes on a certain appearance. Developing a common framework for representing word meanings and learning a large vocabulary of words in this framework is a source of ongoing research. Related to this issue is the training data required to learn good models. A fourth problem is more complex linguistic structures. For example, conditional statements such as "If a truck comes into receiving, empty all the tire pallets into storage alpha," requires identifying the event of the truck's arrival before acting. Our long-term research program is to develop a probabilistic framework for enabling people to flexibly interact with robots using language.

8 Acknowledgments

This work was sponsored by the Robotics Consortium of the U.S Army Research Laboratory under the Collaborative Technology Alliance Program, Cooperative Agreement W911NF-10-2-0016, and by the Office of Naval Research under MURI N00014-07-1-0749. We thank Chad Jenkins and the Brown Robotics lab for their help with the PR2 demonstration.

References

Akmajian, A. (2010). *Linguistics an introduction to language and communication*. MIT Press, Cambridge, Mass. Chapter 5.

Andreas, J. and Klein, D. (2015). Alignment-based compositional semantics for instruction following. *CoRR*, abs/1508.06491.

Andreas, J., Rohrbach, M., Darrell, T., and Klein, D. (2016). Learning to compose neural networks for question answering. *arXiv preprint arXiv:1601.01705*.

Andrew, G. and Gao, J. (2007). Scalable training of L1-regularized log-linear models. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 33–40.

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

Arumugam, D., Karamcheti, S., Gopalan, N., Wong, L. L. S., and Tellex, S. (2017 in press). Accurately and Efficiently Interpreting Human-Robot Instructions of Varying Granularities. In *Robotics: Science and Systems*.

Bachrach, A., He, R., and Roy, N. (2009). Autonomous flight in unstructured and unknown indoor environments. In *Proceedings of the European Micro Air Vehicle Conference (EMAV)*.

Bilmes, J. (2004). On virtual evidence and soft evidence in bayesian networks.

Bollini, M., Tellex, S., Thompson, T., Roy, N., and Rus, D. (2012). Interpreting and executing recipes with a cooking robot. In *Proceedings of the International Symposium on Experimental Robotics (ISER)*.

Branavan, S., Silver, D., and Barzilay, R. (2012). Learning to win by reading manuals in a Monte-Carlo framework. *Journal of Artificial Intelligence Research*, 43(1):661–704.

Branavan, S. R. K., Chen, H., Zettlemoyer, L. S., and Barzilay, R. (2009). Reinforcement learning for mapping instructions to actions. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, page 82–90.

Brunskill, E., Kollar, T., and Roy, N. (2007). Topological mapping using spectral clustering and classification. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3491–3496.

Bugmann, G., Klein, E., Lauria, S., and Kyriacou, T. (2004). Corpus-based robotics: A route instruction example. *Proceedings of the International Conference on Intelligent Autonomous Systems (IAS)*, pages 96–103.

Carlson, L. A. and Covey, E. S. (2005). How far is near? Inferring distance from spatial descriptions. *Language and Cognitive Processes*, 20:617–631.

Chen, D. L. and Mooney, R. J. (2011). Learning to interpret natural language navigation instructions from observations. In *Proceedings of the National Conference on Artificial Intelligence (AAAI)*.

Clarke, J., Goldwasser, D., Chang, M., and Roth, D. (2010). Driving semantic parsing from the world's response. In *Proceedings of the Conference on Computational Natural Language Learning (CoNLL)*, pages 18–27.

Cohen, P. and Oviatt, S. (1995). The role of voice input for human-machine communication. *Proceedings of the National Academy of Sciences*, 92(22):9921–9927.

Crick, C., Jay, G., Osentoski, S., Pitzer, B., and Jenkins, O. C. (2011). Rosbridge: Ros for non-ros users. In *Proceedings of the International Symposium on Robotics Research (ISRR)*.

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

Duvallet, F., Kollar, T., and Stentz, T. (2013). Imitation learning for natural language direction following through unknown environments. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*.

Dzifcak, J., Scheutz, M., Baral, C., and Schermerhorn, P. (2009). 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 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3768–3773.

Feldman, J., Lakoff, G., Bailey, D., Narayanan, S., Regier, T., and Stolcke, A. (1996). L_0-the first five years of an automated language acquisition project. *Integration of natural language and vision processing*, 10:205.

Fong, T., Nourbakhsh, I., and Dautenhahn, K. (2003). A survey of socially interactive robots. *Robotics and Autonomous Systems*, 42(3):143–166.

Ge, R. and Mooney, R. J. (2005). A statistical semantic parser that integrates syntax and semantics. In *Proceedings of the Conference on Computational Natural Language Learning (CoNLL)*, pages 9–16. Association for Computational Linguistics.

Glass, J. (2003). A probabilistic framework for segment-based speech recognition. *Computer Speech & Language*, 17(2-3):137–152.

Grzonka, S., Grisetti, G., and Burgard, W. (2009). Towards a navigation system for autonomous indoor flying. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 2878–2883.

Harnad, S. (1990). The symbol grounding problem. *Physica D: Nonlinear Phenomena*, 42(1-3):335 – 346.

He, R., Prentice, S., and Roy, N. (2008). Planning in information space for a quadrotor helicopter in a GPS-denied environments. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 1814–1820.

Howard, T., Tellex, S., and Roy, N. (2014). A natural language planner interface for mobile manipulators. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*.

Hsiao, K., Tellex, S., Vosoughi, S., Kubat, R., and Roy, D. (2008). Object schemas for grounding language in a responsive robot. *Connection Science*, 20(4):253–276.

Hsiao, K.-y., Mavridis, N., and Roy, D. (2003). Coupling perception and simulation: Steps towards conversational robotics. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 928–933.

Huang, A. S., Tellex, S., Bachrach, A., Kollar, T., Roy, D., and Roy, N. (2010). Natural language command of an autonomous micro-air vehicle. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*.

Jackendoff, R. S. (1983). *Semantics and Cognition*, pages 161–187. MIT Press.

Janner, M., Narasimhan, K., and Barzilay, R. (2017). Representation learning for grounded spatial reasoning. *arXiv preprint arXiv:1707.03938*.

Knepper, R. A., Tellex, S., Li, A., Roy, N., and Rus, D. (2013). Single assembly robot in search of human partner: Versatile grounded language generation. In *Proceedings of the Human Robot Interaction (HRI) Workshop on Collaborative Manipulation*.

Kollar, T., Krishnamurthy, J., and Strimel, G. (2013a). Toward interactive grounded language acquisition. In *Proceedings of Robotics: Science and Systems (RSS)*.

Kollar, T., Perera, V., Nardi, D., and Veloso, M. (2013b). Learning environmental knowledge from task-based human-robot dialog. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*.

Kollar, T. and Roy, N. (2009). Utilizing object-object and object-scene context when planning to find things. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 4116–4121.

Kollar, T., Tellex, S., Roy, D., and Roy, N. (2010a). Grounding verbs of motion in natural language commands to robots. In *Proceedings of the International Symposium on Experimental Robotics (ISER)*.

Kollar, T., Tellex, S., Roy, D., and Roy, N. (2010b). Toward understanding natural language directions. In *Proceedings of the ACM/IEEE International Conference on Human-Robot Interaction (HRI)*.

Kress-Gazit, H. and Fainekos, G. E. (2008). Translating structured English to robot controllers. *Advanced Robotics*, 22:1343–1359.

Krishnamurthy, J. and Kollar, T. (2013). Jointly learning to parse and perceive: Connecting natural language to the physical world. In *Transactions of the Association for Computational Linguistics*, volume 1, pages 192–206.

Kudo, T. (2009). CRF++: Yet another CRF toolkit. http://crfpp.sourceforge.net.

Kwiatkowski, T., Zettlemoyer, L., Goldwater, S., and Steedman, M. (2010). Inducing probabilistic CCG grammars from logical form with higher-order unification. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 1223–1233. Association for Computational Linguistics.

Landau, B. and Jackendoff, R. (1993). Whence and whither in spatial language and spatial cognition? *Behavioral and Brain Sciences*, 16(02):255–265.

Li, X. (2009). On the use of virtual evidence in conditional random fields. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 3-Volume 3*, pages 1289–1297. Association for Computational Linguistics.

Liang, P., Bouchard-Côté, A., Klein, D., and Taskar, B. (2006). An end-to-end discriminative approach to machine translation. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics*, pages 761–768. Association for Computational Linguistics.

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

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

Marneffe, M.-C. D., Maccartney, B., and Manning, C. D. (2006). Generating typed dependency parses from phrase structure parses. In *Proceedings of the International Conference on Language Resources and Evaluation (LREC)*, pages 449–454, Genoa, Italy.

Marocco, D., Cangelosi, A., Fischer, K., and Belpaeme, T. (2010). Grounding action words in the sensorimotor interaction with the world: experiments with a simulated icub humanoid robot. *Frontiers in Neurorobotics*, 4(0).

Matuszek, C., FitzGerald, N., Zettlemoyer, L., Bo, L., and Fox, D. (2012a). A joint model of language and perception for grounded attribute learning. *arXiv preprint arXiv:1206.6423*.

Matuszek, C., Herbst, E., Zettlemoyer, L., and Fox, D. (2012b). Learning to parse natural language commands to a robot control system. In *Proceedings of the International Symposium on Experimental Robotics (ISER)*.

McCallum, A. K. (2002). MALLET: A machine learning for language toolkit. http://mallet.cs.umass.edu.

Misra, D. K., Langford, J., and Artzi, Y. (2017a). Mapping instructions and visual observations to actions with reinforcement learning. *CoRR*, abs/1704.08795.

Misra, D. K., Langford, J., and Artzi, Y. (2017b). Mapping instructions and visual observations to actions with reinforcement learning. *arXiv preprint arXiv:1704.08795*.

Misra, D. K., Sung, J., Lee, K., and Saxena, A. (2016). Tell me dave: Context-sensitive grounding of natural language to manipulation instructions. *The International Journal of Robotics Research*, 35(1-3):281–300.

Modayil, J. and Kuipers, B. (2007). Autonomous development of a grounded object ontology by a learning robot. In *Proceedings of the National Conference on Artificial Intelligence (AAAI)*, pages 1095–1101.

Nenov, V. I. and Dyer, M. G. (1993). Perceptually grounded language learning: Part 1a neural network architecture for robust sequence association. *Connection Science*, 5(2):115–138.

Nenov, V. I. and Dyer, M. G. (1994). Perceptually grounded language learning: Part 2dete: a neural/procedural model. *Connection Science*, 6(1):3–41.

Norbury, H. M., Waxman, S. R., and Song, H.-J. (2008). Tight and loose are not created equal: An asymmetry underlying the representation of fit in english-and korean-speakers. *Cognition*, 109(3):316–325.

Paul, R., Arkin, J., Roy, N., and Howard, T. M. (2016). Efficient grounding of abstract spatial concepts for natural language interaction with robot manipulators. In *Proceedings of Robotics Science and Systems (RSS)*.

Piantadosi, S., Goodman, N., Ellis, B., and Tenenbaum, J. (2008). A Bayesian model of the acquisition of compositional semantics. In *Proceedings of the Annual Conference of the Cognitive Science Society (CSS)*.

Poon, H. and Domingos, P. (2009). Unsupervised semantic parsing. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1–10. Association for Computational Linguistics.

Regier, T. P. (1992). *The Acquisition of Lexical Semantics for Spatial Terms: A Connectionist Model of Perceptual Categorization*. PhD thesis, University of California at Berkeley.

Roy, D. (2005). Semiotic schemas: A framework for grounding language in action and perception. *Artificial Intelligence*, 167(1-2):170–205.

Roy, D., Hsiao, K.-Y., and Mavridis, N. (2003). Conversational robots: building blocks for grounding word meaning. In *Proceedings of the HLT-NAACL 2003 Workshop on Learning Word Meaning from Non-Linguistic Data*, pages 70–77, Stroudsburg, PA, USA. Association for Computational Linguistics.

Roy, D. K. and Pentland, A. P. (2002). Learning words from sights and sounds: A computational model. *Cognitive science*, 26(1):113–146.

Severinson-Eklundh, K., Green, A., and Hüttenrauch, H. (2003). Social and collaborative aspects of interaction with a service robot. *Robotics and Autonomous Systems*, 42(3):223–234.

Shimizu, N. (2006). Semantic discourse segmentation and labeling for route instructions. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics: Student Research Workshop*, COLING ACL '06, pages 31–36, Stroudsburg, PA, USA. Association for Computational Linguistics.

Shimizu, N. and Haas, A. (2009). Learning to follow navigational route instructions. In *Proceedings of the International Joint Conference on Artifical Intelligence (IJCAI)*, pages 1488–1493, San Francisco, CA.

Skubic, M., Perzanowski, D., Blisard, S., Schultz, A., Adams, W., Bugajska, M., and Brock, D. (2004). Spatial language for human-robot dialogs. *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews*, 34(2):154–167.

Sugita, Y. and Tani, J. (2005). Learning semantic combinatoriality from the interaction between linguistic and behavioral processes. *Adaptive Be-*

havior - Animals, Animats, Software Agents, Robots, Adaptive Systems, 13:33–52.

Talmy, L. (2005). The fundamental system of spatial schemas in language. In Hamp, B., editor, *From Perception to Meaning: Image Schemas in Cognitive Linguistics*, pages 199–234. Mouton de Gruyter.

Teller, S., Walter, M., Antone, M., Correa, A., Davis, R., Fletcher, L., Frazzoli, E., Glass, J., How, J., Huang, A., Jeon, J., Karaman, S., Luders, B., Roy, N., and Sainath, T. (2010). A voice-commandable robotic forklift working alongside humans in minimally-prepared outdoor environments. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 526–533.

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

Tellex, S., Kollar, T., Dickerson, S., Walter, M. R., Banerjee, A. G., Teller, S., and Roy, N. (2011b). Approaching the symbol grounding problem with probabilistic graphical models. *AI Magazine*, 32(4):64–76.

Tellex, S., Kollar, T., Shaw, G., Roy, N., and Roy, D. (2010). Grounding spatial language for video search. In *International Conference on Multimodal Interfaces and the Workshop on Machine Learning for Multimodal Interaction*, pages 31:1–31:8.

Tellex, S., Thaker, P., Joseph, J., and Roy, N. (2013). Learning perceptually grounded word meanings from unaligned parallel data. *Machine Learning Journal*.

Thompson, C. A. and Mooney, R. J. (2003). Acquiring word-meaning mappings for natural language interfaces. *Journal of Artificial Intelligence Research*, 18:1–44.

Viterbi, A. J. (1967). Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. *IEEE Transactions on Information Theory*, 13(2):260–269.

Vogel, A. and Jurafsky, D. (2010). Learning to follow navigational directions. In *Proceedings of the Association for Computational Linguistics (ACL)*, pages 806–814. Association for Computational Linguistics.

Walter, M., Friedman, Y., Antone, M., and Teller, S. (2012). One-shot visual appearance learning for mobile manipulation. *International Journal of Robotics Research*, 31(4):554–567.

Winograd, T. (1971). *Procedures as a Representation for Data in a Computer Program for Understanding Natural Language*. PhD thesis, Massachusetts Institute of Technology. Ph.D. thesis.

Wong, Y. and Mooney, R. (2007). Learning synchronous grammars for semantic parsing with lambda calculus. In *Proceedings of the Association for Computational Linguistics (ACL)*, pages 960–967.

Xu, H. and Saenko, K. (2015). Ask, attend and answer: Exploring question-guided spatial attention for visual question answering. *CoRR*, abs/1511.05234.

Yang, Z., He, X., Gao, J., Deng, L., and Smola, A. J. (2015). Stacked attention networks for image question answering. *CoRR*, abs/1511.02274.

Zettlemoyer, L. and Collins, M. (2005a). Learning to map sentences to logical form: Structured classification with probabilistic categorial grammars. *Proc. of 21th Conf. on Uncertainty in Artificial Intelligence (UAI-2005)*.

Zettlemoyer, L. S. and Collins, M. (2005b). Learning to map sentences to logical form: Structured classification with probabilistic categorial grammars. In *Proceedings of the Conference on Uncertainty in Artificial Intelligence (UAI)*, pages 658–666.

Zitnick, C. L., Agrawal, A., Antol, S., Mitchell, M., Batra, D., and Parikh, D. (2016). Measuring machine intelligence through visual question answering. *AI Magazine*, 37(1):63–72.

A Appendix: Features

We present a representative set of features used in our work below. Features not included here include distance and contact-based features intended to capture the semantics of words such as "on" and "near." Many spatial prepositions can be defined in terms of the coordinate axes of the landmark object (Talmy, 2005; Landau and Jackendoff, 1993). For example, "across" requires the figure to be perpendicular to the major axis of the landmark (Talmy, 2005; Landau and Jackendoff, 1993): to cross a road, one most go from one side to the other, and not from one end to the other. In order to define features that capture the meaning of words such as "across," we must define an algorithm for finding the major axis of the landmark. In many contexts, there is no single set of axes: for example, there are many paths across a square room. We solve this problem by computing the unique axes that the figure imposes on the landmark and then quantifying how well those axes match the landmark. The system computes these axes by finding the line that connects the first and last point in the figure and extending this line until it intersects the landmark. The origin of the axes is the midpoint of this line segment, and the endpoints are the two points where the axes intersect the landmark.

Talmy (2005) defines "past" as figure's path must be going perpendicular to a point P "at a proximal remove" from the landmark. The path of the figure must be perpendicular to a line going from the landmark to this point. We define the "past axes" as this line and a line perpendicular to it.

- **angleBtwnLinearizedObjects**: The angular difference of the slope of lines fit to the points in the figure and points in the boundary of the landmark, using linear regression.

- **angleFigureToPastAxes**: The angle between a line fit to points in the figure using linear regression and the line between the figure and the landmark at the closest point.

- **averageDistStartEndLandmarkBoundary**: The average of the distance between the start of the figure and the boundary of the landmark, and the distance between the end of the figure and the boundary of the landmark.

- **displacementFromLandmark**: The difference in distance between the start point of the figure to the landmark, and the end point of the figure to the landmark, illustrated in Figure 14.

- **distAlongLandmarkBtwnAxes**: The distance along the perimeter of landmark between the start and end of the minor axis.

- **distStartLandmarkBoundary**: The distance of the start of the figure to the boundary of the landmark.

- **distFigureEndToLandmark**: The distance from the end point of the figure to the boundary of the landmark.

- **distFigureStartToLandmark**: The distance from the start point of the figure to the landmark.
- **eigenAxesRatio**: The ratio between the eigenvectors of the covariance matrix of the landmark when represented as an occupancy grid.
- **figureCenterOfMassToAxesOrigin**: The distance between the center of mass of the points in the figure and the axes origin.
- **figureCenterOfMassToLandmarkCentroid**: The distance between the center of mass of the figure and the centroid of the landmark.
- **pastAxesLength**: The distance between the figure and the landmark, at the point the figure most closely approaches the landmark.
- **peakDistToAxes**: The maximum distance between the figure and the axes it imposes on the landmark, for the part of the figure that is inside the landmark.
- **ratioFigureToAxes**: The ratio between the distance between the start and end points of the figure and the length of the major axis.
- **stdDevToAxes**: The standard deviation of the distances between the figure and the major axis of the landmark, stepping along the figure.

Many of these features are based on distances. However, spatial relations are scale invariant: one can use a word like "across" to describe a small-scale event such as crab crawling across a rock, and a large-scale event such as a car driving across the country. Distance features such as *distFigureEndToLandmark* and *displacementFromLandmark* are made scale invariant by normalizing by the bounding box of the scene. The normalization prevents learned models from overfitting to a particular geometric scale.

A.1 Verb Features for Route Directions

Each type of directive corresponds to an expected turn amount, θ_{λ_v}, which is 0° for "straight," −90° for "right," and 90° for "left." Next, we define θ_{γ_t}, the actual amount that a robot would turn when following the trajectory. Finally, we use a sigmoid distribution on the difference between the expected turn amount and actual turn amount:

$$p(\phi_v|\lambda_v, \gamma_t) \approx \frac{1}{1 - e^{|\theta_{\lambda_v} - \theta_{\gamma_t}|}} \tag{16}$$

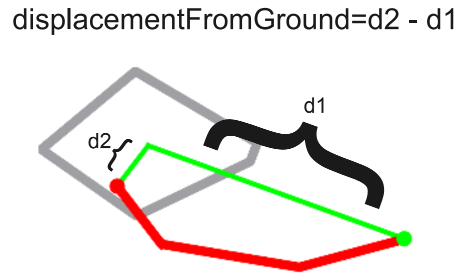


Figure 14: Illustration of the computation of the feature *displacementFromLandmark*.

In aerial task domains, commands also include three-dimensional components, such as "fly up" or "fly down." If λ_v contains one of these directives, we check whether the trajectory ends at a higher or lower elevation than the start, or stays the same. If the trajectory's elevation matches the directive, we set $p(\phi_v|\lambda_v, \gamma_t) = 1$, and otherwise, 0.000001.