Understanding Natural Language Commands for Robotic Navigation and Mobile Manipulation

Stefanie Tellex[1] and **Thomas Kollar**[1] and **Steven Dickerson**[1] and
Matthew R. Walter and **Ashis Gopal Banerjee** and **Seth Teller** and **Nicholas Roy**
Computer Science and Artificial Intelligence Laboratory
Massachusetts Institute of Technology
Cambridge, MA 02139

Abstract

This paper describes a new model for understanding natural language commands given to autonomous systems that perform navigation and mobile manipulation in semi-structured environments. Previous approaches have used models with fixed structure to infer the likelihood of a sequence of actions given the environment and the command. In contrast, our framework, called Generalized Grounding Graphs (G^3), dynamically instantiates a probabilistic graphical model for a particular natural language command according to the command's hierarchical and compositional semantic structure. Our system performs inference in the model to successfully find and execute plans corresponding to natural language commands such as "Put the tire pallet on the truck." The model is trained using a corpus of commands collected using crowdsourcing. We pair each command with robot actions and use the corpus to learn the parameters of the model. We evaluate the robot's performance by inferring plans from natural language commands, executing each plan in a realistic robot simulator, and asking users to evaluate the system's performance. We demonstrate that our system can successfully follow many natural language commands from the corpus.

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." Such commands are challenging because they involve events ("Put"), objects ("the tire pallet"), and places ("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 our system is able to follow for the robotic forklift domain.

We frame the problem of following instructions as inferring the most likely robot state sequence from a natural language command. Previous approaches (Kollar et al., 2010; Shimizu and Haas, 2009) assume that natural language commands have a fixed and flat structure that can be exploited when inferring actions for the robot. However, this kind of fixed and flat sequential structure does not allow for variable arguments or nested clauses. At training time, when using a flat structure, the system 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 on the pallet beside the truck," has two arguments ("the box" and "on the pallet beside the truck"), both of which are necessary to learn an accurate meaning for the verb "put." In order to infer the meaning of unconstrained natural language commands, it is critical for the model to exploit these compositional and hierarchical linguistic structures at both learning and inference time.

Commands from the corpus

- 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

(a) Robotic forklift (b) Sample commands

Figure 1: A target robotic platform for mobile manipulation and navigation (Teller et al. 2010), and sample commands from the domain, created by untrained human annotators. Our system can successfully follow these commands.

To address these issues, we introduce a new model called Generalized Grounding Graphs (G^3). A *grounding graph* is a probabilistic graphical model that is instantiated dynamically according to the compositional and hierarchical structure of a natural language command. Given a natural language command, the structure of the grounding graph model is induced using Spatial Description Clauses (SDCs), a semantic structure introduced by (Kollar et al. 2010). Each SDC represents a linguistic constituent from the command that can be mapped to an aspect of the world or *grounding*, such as an object, place, path or event. In the G^3 frame-

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

work, the structure of each individual SDC and the random variables, nodes, and edges in the overall grounding graph depend on the specific words in the text.

The model is trained on a corpus of natural language commands paired with groundings for each part of the command, enabling the system to automatically learn meanings for words in the corpus, including complex verbs such as "put" and "take." We evaluate the system in the specific domain of natural language commands given to a robotic forklift, although our approach generalizes to any domain where linguistic constituents can be associated with specific actions and environmental features. Videos of example commands paired with inferred action sequences can be seen at http://spatial.csail.mit.edu/grounding.

2 Related Work

Beginning with SHRDLU (Winograd 1970), many systems have exploited the compositional structure of language to statically generate a plan corresponding to a natural language command (Dzifcak et al., 2009; Hsiao et al., 2008; MacMahon, Stankiewicz, and Kuipers, 2006; Skubic et al., 2004). Our work moves beyond this framework by defining a probabilistic graphical model according to the structure of the natural language command, inducing a distribution over plans and groundings. This approach enables the system to learn models for the meanings of words in the command and efficiently perform inference over many plans to find the best sequence of actions and groundings corresponding to each part of the command.

Others have used generative and discriminative models for understanding route instructions, but did not use the hierarchical nature of the language to understand mobile manipulation commands (Kollar et al., 2010; Matuszek, Fox, and Koscher, 2010; Vogel and Jurafsky, 2010). (Shimizu and Haas 2009) use a flat, fixed action space to train a CRF that followed route instructions. Our approach, in contrast, interprets a grounding graph as a structured CRF, enabling the system to learn over a rich compositional action space.

The structure of SDCs builds on the work of (Jackendoff 1983), (Landau and Jackendoff 1993) and (Talmy 2005), providing a computational instantiation of their formalisms. (Katz 1988) devised ternary expressions to capture relations between words in a sentence. The SDC representation adds types for each clause, each of which induces a candidate space of groundings, as well as the ability to represent multiple landmark objects, making it straightforward to directly associate groundings with SDCs.

3 Approach

Our system takes as input a natural language command and outputs a plan for the robot. In order to infer a correct plan, it must find a mapping between parts of the natural language command and corresponding groundings (objects, paths, and places) in the world. We formalize this mapping with a *grounding graph*, a probabilistic graphical model with random variables corresponding to groundings in the world. Each grounding is taken from a semantic map of the environment, which consists of a metric map with the location, shape and name of each object and place, along with a topology that defines the environment's connectivity. At the top level, the system infers a grounding corresponding to the entire command, which is then interpreted as a plan for the robot to execute.

More formally, we define Γ to be the set of all groundings γ_i for a given command. In order to allow for uncertainty in candidate groundings, we introduce binary correspondence variables Φ; each $\phi_i \in \Phi$ is true if $\gamma_i \in \Gamma$ is correctly mapped to part of the natural language command, and false otherwise. Then we want to maximize the conditional distribution:

$$\underset{\Gamma}{\operatorname{argmax}}\, p(\Phi = True|\text{command}, \Gamma) \quad (1)$$

This optimization is different from conventional CRF inference, where the goal is to infer the most likely hidden labels Φ. Although our setting is discriminative, we fix the correspondence variables Φ and search over features induced by Γ to find the most likely grounding. By formulating the problem in this way, we are able to perform domain-independent learning and inference.

3.1 Spatial Description Clauses

The factorization of the distribution in Equation 1 is defined according to the grounding graph constructed for a natural language command. To construct a probabilistic model according to the linguistic structure of the command, we decompose a natural language command into a hierarchy of Spatial Description Clauses or SDCs (Kollar et al. 2010). Each SDC corresponds to a constituent of the linguistic input and consists of a *figure* f, a *relation* r, and a variable number of *landmarks* l_i. A general natural language command is represented as a tree of SDCs. SDCs for the command "Put the tire pallet on the truck" appear in Figure 2a, and "Go to the pallet on the truck" in Figure 3a. Leaf SDCs in the tree contain only text in the figure field, such as "the tire pallet." Internal SDCs have other fields populated, such as "the tire pallet on the truck." The figure and landmark fields of internal SDCs are always themselves SDCs. The text in fields of an SDC does not have to be contiguous. For phrasal verbs such as "Put the tire pallet down," the relation field contains "Put down," and the landmark field is "the tire pallet."

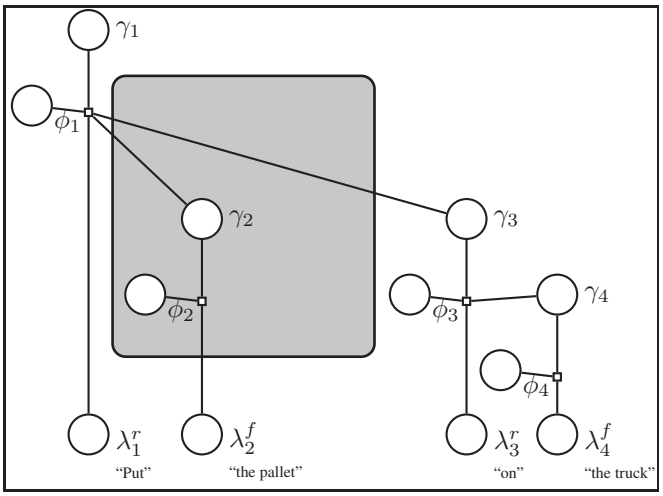
(Kollar et al. 2010) introduced SDCs and used them to define a probabilistic model that factors according to the sequential structure of language. Here we change the formalism slightly to collapse the verb and spatial relation fields into a single relation and exploit the hierarchical structure of SDCs in the factorization of the model.

The system infers groundings in the world corresponding to each SDC. To structure the search for groundings and limit the size of the search space, we follow (Jackendoff 1983) and assign a type to each SDC:

- **EVENT** An action sequence that takes place (or should take place) in the world (e.g. "Move the tire pallet").
- **OBJECT** A thing in the world. This category includes people and the robot as well as physical objects (e.g. "Forklift," "the tire pallet," "the truck," "the person").

$EVENT_1(r = \texttt{Put},$
$l = OBJ_2(f = \texttt{the pallet}),$
$l2 = PLACE_3(r = \texttt{on},$
$l = OBJ_4(f = \texttt{the truck})))$

(a) SDC tree

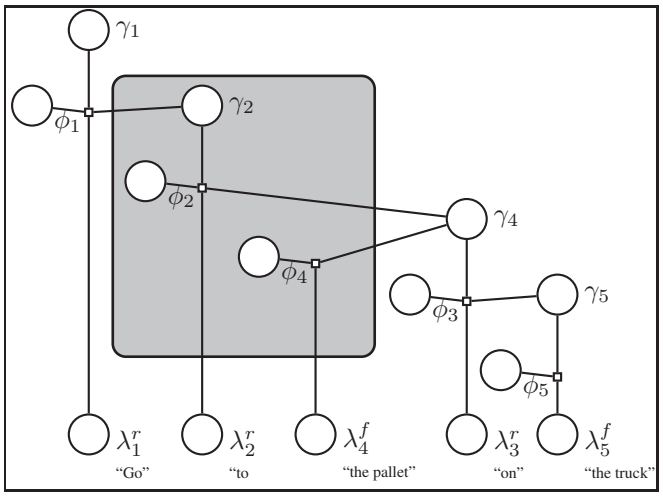


(b) Induced model

Figure 2: (a) SDC tree for "Put the pallet on the truck." (b) Induced graphical model and factorization.

$EVENT_1(r = \texttt{Go}$
$l = PATH_2(r = \texttt{to},$
$l = OBJ_3(f = OBJ_4(f = \texttt{the pallet}),$
$r = \texttt{on},$
$l = OBJ_5(f = \texttt{the truck}))))$

(a) SDC tree



(b) Induced model

Figure 3: (a) SDC tree for "Go to the pallet on the truck." (b) A different induced factor graph from Figure 2. Structural differences between the two models are highlighted in gray.

- **PLACE** A place in the world (e.g. "on the truck," or "next to the tire pallet").
- **PATH** A path or path fragment through the world (e.g. "past the truck," or "toward receiving").

Each EVENT and PATH SDC contains a relation with one or more core arguments. Since almost all relations (e.g. verbs) take two or fewer core arguments, we use at most two landmark fields l_1 and l_2 for the rest of the paper. We have built an automatic SDC extractor that uses the Stanford dependencies, which are extracted using the Stanford Parser (de Marneffe, MacCartney, and Manning 2006).

3.2 Generalized Grounding Graphs

We present an algorithm for constructing a grounding graph according to the linguistic structure defined by a tree of SDCs. The induced grounding graph for a given command is a bipartite factor graph corresponding to a factorization of the distribution from Equation 1 with factors Ψ_i and normalization constant Z:

$$p(\Phi|\text{commands}, \Gamma) = p(\Phi|\text{SDCs}, \Gamma) \quad (2)$$

$$= \frac{1}{Z} \prod_i \Psi_i(\phi_i, \text{SDC}_i, \Gamma) \quad (3)$$

The graph has two types of nodes: random variables and factors. First we define the following random variables:

- ϕ_i True if the grounding γ_i corresponds to i^{th} SDC, and false otherwise.
- λ_i^f The words of the figure field of the i^{th} SDC.
- λ_i^r The words of the relation field of the i^{th} SDC.
- $\lambda_i^{l1}, \lambda_i^{l2}$ The words of the first and second landmark fields of the i^{th} SDC; if non-empty, always a child SDC.
- $\gamma_i^f, \gamma_i^{l1}, \gamma_i^{l2} \in \Gamma$ The groundings associated with the corresponding field(s) of the i^{th} SDC: the state sequence of the robot (or an object), or a location in the semantic map.

For a phrase such as "the pallet on the truck," λ_i^r is the word "on," and γ_i^f and γ_i^{l1} correspond to objects in the world, represented as a location, a bounding box, and a list of labels. ϕ_i would be true if the induced features between γ_i^f and γ_i^{l1} correspond to "on," and false otherwise.

Each random variable connects to one or more factor nodes, Ψ_i. Graphically, there is an edge between a variable and a factor if the factor takes that variable as an argument. The specific factors created depend on the structure of the SDC tree. The factors Ψ fall into two types:

- $\Psi(\phi_i, \lambda_i^f, \gamma_i)$ for leaf SDCs.
- $\Psi(\phi_i, \lambda_i^r, \gamma_i^f, \gamma_i^{l1})$ or $\Psi(\phi_i, \lambda_i^r, \gamma_i^f, \gamma_i^{l1}, \gamma_i^{l2})$ for internal SDCs.

Leaf SDCs contain only λ_i^f and a grounding γ_i^f. For example, the phrase "the truck" is a leaf SDC that generates the subgraph in Figure 3 containing variables γ_5, ϕ_5 and λ_5^f. The value of γ_5 is an object in the world, and ϕ_5 is true if the

object corresponds to the words "the truck" and false otherwise (for example, if γ_5 was a pallet).

An internal SDC has text in the relation field and SDCs in the figure and landmark fields. For these SDCs, ϕ_i depends on the text of the relation field, and the groundings (rather than the text) of the figure and landmark fields. For example, "the pallet on the truck" is an internal SDC, with a corresponding grounding that is a place in the world. This SDC generates the subgraph in Figure 3 containing the variables γ_4, γ_5, ϕ_3, and λ_3^r. ϕ_3 is true if γ_4 is "on" γ_5, and false otherwise.

Figures 2 and 3 show the SDC trees and induced grounding graphs for two similar commands: "Put the pallet on the truck" and "Go to the pallet on the truck." In the first case, "Put" is a two-argument verb that takes an OBJECT and a PLACE. The model in Figure 2b connects the grounding γ_3 for "on the truck" directly to the factor for "Put." In the second case, "on the truck" modifies "the pallet." For this reason, the grounding γ_4 for "on the truck" is connected to "the pallet." The differences between the two models are highlighted in gray.

In this paper we use generalized grounding graphs to define a discriminative model in order to train the model from a large corpus of data. However, the same graphical formalism can also be used to define factors for a generative graphical model, or even a constraint network that does not take a probabilistic approach at all. For example, the generative model described in (Kollar et al. 2010) for following route instructions is a special case of this more general framework.

We model the distribution in Equation 2 as a conditional random field in which each potential function Ψ takes the following form (Lafferty, McCallum, and Pereira 2001):

$$\Psi_i(\phi_i, \text{SDC}_i, \Gamma) = \exp\left(\sum_k \mu_k s_k(\phi_i, \text{SDC}_i, \Gamma)\right) \quad (4)$$

Here, s_k are feature functions that take as input the binary correspondence variable, an SDC and a set of groundings and output a binary decision. The μ_k are the weights corresponding to the output of a particular feature function.

At training time, we observe SDCs, their corresponding groundings Γ, and the output variable Φ. In order to learn the parameters μ_k that maximize the likelihood of the training dataset, we compute the gradient, and use the Mallet toolkit (McCallum 2002) to optimize the parameters of the model via gradient descent with L-BFGS (Andrew and Gao 2007). When inferring a plan, we optimize over Γ by fixing Φ and the SDCs as in Equation 1.

3.3 Features

To train the model, the system extracts binary features s_k for each factor Ψ_i. These features correspond to the degree to which each Γ correctly grounds SDC_i. For a relation such as "on," a natural feature is whether the the landmark grounding supports the figure grounding. However, the feature $supports(\gamma_i^f, \gamma_i^l)$ alone is not enough to enable the model to learn that "on" corresponds to $supports(\gamma_i^f, \gamma_i^l)$. Instead we need a feature that also takes into account the word "on:"

$$supports(\gamma_i^f, \gamma_i^l) \wedge (\text{"on"} \in \lambda_i^r) \quad (5)$$

More generally, we implemented a set of base features involving geometric relations between the γ_i. Then to compute features s_k we generate the Cartesian product of the base features with the presence of words in the corresponding fields of the SDC. A second problem is that many natural features between geometric objects are continuous rather than binary valued. For example, for the relation "next to," one feature is the normalized distance between γ_i^f and γ_i^l. To solve this problem, we discretize continuous features into uniform bins. We use 49 base features for leaf OBJECT and PATH SDCs, 56 base features for internal OBJECT and PATH SDCs, 112 base features for EVENT SDCs and 47 base features for PATH SDCs. This translates to 147,274 binary features after the Cartesian product with words and discretization.

For OBJECTs and PLACEs, geometric features correspond to relations between two three-dimensional boxes in the world. All continuous features are first normalized so they are scale-invariant, then discretized to be a set of binary features. Examples include

- $supports(\gamma_i^f, \gamma_i^l)$. For "on" and "pick up."
- $distance(\gamma_i^f, \gamma_i^l)$. For "near" and "by."
- $avs(\gamma_i^f, \gamma_i^l)$. For "in front of" and "to the left of." Attention Vector Sum or AVS (Regier and Carlson 2001) measures the degree to which relations like "in front of" or "to the left of" are true for particular groundings.

In order to compute features for relations like "to the left" or "to the right," the system needs to compute a *frame of reference*, or the orientation of a coordinate system. We compute these features for frames of reference in all four cardinal directions at the agent's starting orientation, the agent's ending orientation, and the agent's average orientation during the action sequence.

For PATH and EVENT SDCs, groundings correspond to the location and trajectory of the robot and any objects it manipulates over time. Base features are computed with respect to the entire motion trajectory of a three-dimensional object through space. Examples include:

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

We also use the complete set of features described in (Tellex 2010). Finally, we compute the same set of features as for OBJECTs and PLACEs using the state at the beginning of the trajectory, the end of the trajectory, and the average during the trajectory.

The system must map noun phrases such as "the wheel skid" to a grounding γ_i^f for a physical object in the world with location, geometry, and a set of labels such as {"tires", "pallet"}. To address this issue we introduce a second class of base features that correspond to the likelihood that an unknown word actually denotes a known concept. The system computes word-label similarity in two ways: using WordNet; and from co-occurrence statistics obtained by downloading millions of images and corresponding tags from Flickr (Kollar et al. 2010).

3.4 Inference

Given a command, we want to find the set of most probable groundings. During inference, we fix the values of Φ and the SDCs and search for groundings Γ that maximize the probability of a match, as in Equation 1. Because the space of potential groundings includes all permutations of object assignments, as well as every feasible sequence of actions the agent might perform, the search space becomes large as the number of objects and potential manipulations in the world increases. In order to make the inference tractable, we use a beam search with a fixed beam-width of twenty in order to bound the number of candidate groundings considered for any particular SDC.

A second optimization is that we search in two passes: the algorithm first finds and scores candidate groundings for OBJECT and PLACE SDCs, then uses those candidates to search the much larger space of robot action sequences, corresponding to EVENTs and PATHs. This optimization exploits the types and independence relations among SDCs to structure the search so that these candidates need to be computed only once, rather than for every possible EVENT.

Once a full set of candidate OBJECT and PLACE groundings is obtained up to the beam width, the system searches over possible action sequences for the agent, scoring each sequence against the language in the EVENT and PATH SDCs of the command. After searching over potential action sequences, the system returns a set of object groundings and a sequence of actions for the agent to perform. Figure 4 shows the actions and groundings identified in response to the command "Put the tire pallet on the truck."

4 Evaluation

To train and evaluate the system, we collected a corpus of natural language commands paired with robot actions and environment state sequences. We use this corpus both to train the model and to evaluate end-to-end performance of the system when following real-world commands from untrained users.

4.1 Corpus

To quickly generate a large corpus of examples of language paired with robot plans, 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 had 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 showing the forklift executing an action in a simulated warehouse. Each subject interpreted each video only once, but we collected multiple commands (an average of 13) for each video.

Actions included moving objects from one location to another, picking up objects, and driving to specific locations. Subjects did not see any text describing the actions or objects in the video, leading to a wide variety of natural language commands including nonsensical ones such as "Load the forklift onto the trailer," and misspelled ones such as "tyre" (tire) or "tailor" (trailer). Example commands from the corpus are shown in Figure 1.

To train the system, each SDC must be associated with a grounded object in the world. We manually annotated SDCs in the corpus, and then annotated each OBJECT and PLACE SDC with an appropriate grounding. Each PATH and EVENT grounding was automatically associated with the action or agent path from the log associated with the original video. This approximation is faster to annotate but leads to problems 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 EVENT SDC refers to a different part of the state sequence.

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 SDC. Although this heuristic works well for EVENTs and PATHs, ambiguous object SDCs 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 these examples we re-annotated them as positive.

4.2 Cost Function Evaluation

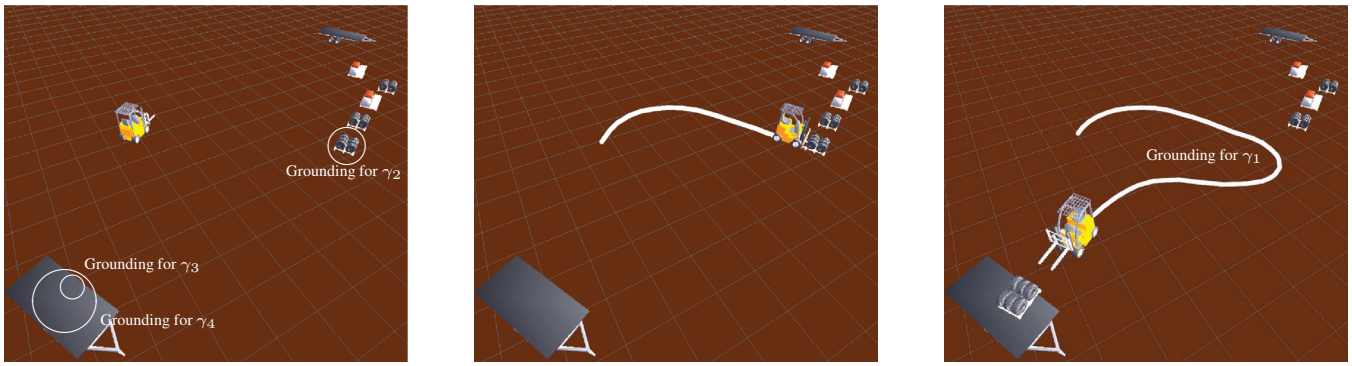
Using the annotated data, we trained the model and evaluated its performance on a held-out test set in a similar environment. We assessed the model's performance at predicting the correspondence variable given access to SDCs and groundings. The test set pairs a disjoint set of scenarios from the training set with language given by subjects from AMT.

SDC type	Precision	Recall	F-score	Accuracy
OBJECT	0.93	0.94	0.94	0.91
PLACE	0.70	0.70	0.70	0.70
PATH	0.86	0.75	0.80	0.81
EVENT	0.84	0.73	0.78	0.80
Overall	0.90	0.88	0.89	0.86

Table 1: Performance of the learned model at predicting the correspondence variable ϕ.

Table 1 reports overall performance on this test set and performance broken down by SDC type. The performance of the model on this corpus indicates that it robustly learns to predict when SDCs 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 SDCs.

For OBJECT SDCs, 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 SDCs with incorrectly annotated groundings that the system actually got right. A second class



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

Figure 4: A sequence of the actions that the forklift takes in response to the command, "Put the tire pallet on the truck." (a) The search grounds objects and places in the world based on their initial positions. (b) The forklift executes the first action, picking up the pallet. (c) The forklift puts the pallet on the trailer.

of low-scoring examples were due to words that did not appear many times in the corpus.

For PLACE SDCs, the system often correctly classifies examples involving the relation "on," such as "on the trailer." However, the model often misclassifies PLACE SDCs 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" and the principal orientation of the "skid of tires" to reason about which location should be grounded to the language "to the right," or "between the pallets on the ground and the other trailer" requires reasoning about multiple objects and a PLACE SDC that has two arguments.

For EVENT SDCs, the model generally performs well on "pick up," "move," and "take" commands. The model correctly predicts 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." We incorrectly predict 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 probably insufficient to learn a good model. "Move" had few good negative examples, since we did not have in the training set, to use as contrast, paths in which the forklift did not move.

4.3 End-to-end Evaluation

The fact that the model performs well at predicting the correspondence variable from annotated SDCs and groundings is promising but does not necessarily translate to good end-to-end performance when inferring groundings associated with a natural language command (as in Equation 1).

To evaluate end-to-end performance, we inferred plans given only commands from the test set and a starting location for the robot. We segmented commands containing multiple top-level SDCs into separate clauses, and utilized the system to infer a plan and a set of groundings for each clause. Plans were then simulated on a realistic, high-fidelity robot simulator from which we created a video of the robot's actions. We uploaded these videos to AMT, where subjects viewed the 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, we conducted the evaluation using known correct and incorrect command-video pairs. In the first condition, subjects saw a command paired with the original video that a different subject watched when creating the command. In the second condition, the subject saw the command paired with random video that was not used to generate the original command. As expected, there was a large difference in performance in the two conditions, shown in Table 2. Despite the diverse and challenging language in our corpus, new annotators agree that commands in the corpus are consistent with the original video. These results show that language in the corpus is understandable by a different annotator.

	Precision
Command with original video	0.91 ($\pm$0.01)
Command with random video	0.11 ($\pm$0.02)

Table 2: The fraction of end-to-end commands considered correct by our annotators for known correct and incorrect videos. We show the 95% confidence intervals in parentheses.

We then evaluated our system by considering three different configurations. Serving as a baseline, the first consisted of ground truth SDCs and a random probability distribution, resulting in a constrained search over a random cost function. The second configuration involved ground truth SDCs and our learned distribution, and the third consisted of automatically extracted SDCs with our learned distribution.

Due to the overhead of the end-to-end evaluation, we con-

	Precision
Constrained search, random cost	0.28 ($\pm$0.05)
Ground truth SDCs (top 30), learned cost	0.63 ($\pm$0.08)
Automatic SDCs (top 30), learned cost	0.54 ($\pm$0.08)
Ground truth SDCs (all), learned cost	0.47 ($\pm$0.04)

Table 3: The fraction of commands considered correct by our annotators for different configurations of our system. We show the 95% confidence intervals in parentheses.

sider results for the top 30 commands with the highest posterior probability of the final plan correctly corresponding to the command text for each configuration. In order to evaluate the relevance of the probability assessment, we also evaluate the entire test set for ground truth SDCs and our learned distribution. Table 3 reports the performance of each configuration along with their 95% confidence intervals. The relatively high performance of the random cost function configuration relative to the random baseline for the corpus is due the fact that the robot is not acting completely randomly on account of the constrained search space. In all conditions, the system performs statistically significantly better than a random cost function.

The system performs noticeably better on the 30 most probable commands than on the entire test set. This result indicates the validity of our probability measure, suggesting that the system has some knowledge of when it is correct and incorrect. The system could use this information to decide when to ask for confirmation before acting.

The system qualitatively produces compelling end-to-end performance. Even when the system makes a mistake, it is often partially correct. For example, it might pick up the left tire pallet instead of the right one. Other problems stem from ambiguous or unusual language in the corpus commands, such as "remove the goods" or "then swing to the right," that make the inference particularly challenging. Despite these limitations, however, our system successfully follows commands such as "Put the tire pallet on the truck," "Pick up the tire pallet" and "put down the tire pallet" and "go to the truck," using only data from the corpus to learn the model.

Although we conducted our evaluation with single SDCs, the framework supports multiple SDCs by performing beam search to find groundings for all components in both SDCs. Using this algorithm, the system successfully followed the commands listed in Figure 1. These commands are more challenging than those with single SDCs because the search space is larger, because there are often dependencies between commands, and because these commands often contain unresolved pronouns like "it."

5 Conclusion

In this paper, we present an approach for automatically generating a probabilistic graphical model according to the structure of natural language navigation or mobile manipulation commands. Our system automatically learns the meanings of complex manipulation verbs such as "put" or "take" from a corpus of natural language commands paired with correct robot actions. We demonstrate promising performance at following natural language commands from a challenging corpus collected from untrained users.

Our work constitutes a step toward robust language understanding systems, but many challenges remain. One limitation of our approach is the need for annotated training data. Unsupervised or semi-supervised modeling frameworks in which the object groundings are latent variables have the potential to exploit much larger corpora without the expense of annotation. Another limitation is the size of the search space; more complicated task domains require deeper search and more sophisticated algorithms. In particular, we plan to extend our approach to perform inference over possible parses as well as groundings in the world.

Our model provides a starting point for incorporating dialog, because it not only returns a plan corresponding to the command, but also groundings (with confidence scores) for each component in the command. This information can enable the system to identify confusing parts of the command in order to ask clarifying questions.

There are many complex linguistic phenomena that our framework does not yet support, such as abstract objects, negation, anaphora, conditionals, and quantifiers. Many of these could be addressed with a richer model, as in (Liang, Jordan, and Klein 2011). For example, our framework does not currently handle negation, such as "Don't pick up the pallet," but it might be possible to do so by fixing some correspondence variables to false (rather than true) during inference. The system could represent anaphora such as "it" in "Pick up the pallet and put it on the truck" by adding a factor linking "it" with its referent, "the pallet." The system could handle abstract objects such as "the row of pallets" if all possible objects were added to the space of candidate groundings. Since each of these modifications would substantially increase the size of the search space, solving these problems will require efficient approximate inference techniques combined with heuristic functions to make the search problem tractable.

6 Acknowledgments

We would like to thank Alejandro Perez, as well as the annotators on Amazon Mechanical Turk and the members of the Turker Nation forum. 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.

References

Andrew, G., and Gao, J. 2007. Scalable training of L1-regularized log-linear models. In *Proc. Int'l Conf. on Machine Learning (ICML)*.

de Marneffe, M.; MacCartney, B.; and Manning, C. 2006. Generating typed dependency parses from phrase structure parses. In *Proc. Int'l Conf. on Language Resources and Evaluation (LREC)*, 449–454.

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.

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

Katz, B. 1988. Using English for indexing and retrieving. In *Proc. Conf. on Adaptivity, Personilization and Fusion of Heterogeneous Information (RIAO)*. MIT Press.

Kollar, T.; Tellex, S.; Roy, D.; and Roy, N. 2010. Toward understanding natural language directions. In *Proc. ACM/IEEE Int'l Conf. on Human-Robot Interaction (HRI)*, 259–266.

Lafferty, J. D.; McCallum, A.; and Pereira, F. C. N. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proc. Int'l Conf. on Machine Learning (ICML)*, 282–289.

Landau, B., and Jackendoff, R. 1993. "What" and "where" in spatial language and spatial cognition. *Behavioral and Brain Sciences* 16:217–265.

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

Matuszek, C.; Fox, D.; and Koscher, K. 2010. Following directions using statistical machine translation. In *Proc. ACM/IEEE Int'l Conf. on Human-Robot Interaction (HRI)*, 251–258.

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

Regier, T., and Carlson, L. A. 2001. Grounding spatial language in perception: An empirical and computational investigation. *J. of Experimental Psychology: General* 130(2):273–98.

Shimizu, N., and Haas, A. 2009. Learning to follow navigational route instructions. In *Proc. Int'l Joint Conf. on Artificial Intelligence (IJCAI)*, 1488–1493.

Talmy, L. 2005. The fundamental system of spatial schemas in language. In Hamp, B., ed., *From Perception to Meaning: Image Schemas in Cognitive Linguistics*. Mouton de Gruyter.

Teller, S.; Walter, M. R.; 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 *Proc. IEEE Int'l Conf. on Robotics and Automation (ICRA)*, 526–533.

Tellex, S. 2010. *Natural Language and Spatial Reasoning*. Ph.D. Dissertation, Massachusetts Institute of Technology.

Winograd, T. 1970. *Procedures as a representation for data in a computer program for understanding natural language.* Ph.D. Dissertation, Massachusetts Institute of Technology.