DCG-UPUP-Away: Automatic Symbol Acquisition through Grounding to Unknowns

by

Mycal Tucker

S.B. Massachusetts Institute of Technology (2015)

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Master of Engineering in Computer Science and Engineering

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

September 2016

Author . . . .
Department of Electrical Engineering and Computer Science
August 16, 2016

Certified by . . . .
Nicholas Roy
Professor
Thesis Supervisor

Accepted by . . . .
Dr. Christopher Terman
Chairman, Masters of Engineering Thesis Committee

DCG-UPUP-Away: Automatic Symbol Acquisition through Grounding to Unknowns

by

Mycal Tucker

Submitted to the Department of Electrical Engineering and Computer Science
on August 16, 2016, in partial fulfillment of the
requirements for the degree of
Master of Engineering in Computer Science and Engineering

Abstract

Research in automatic natural language grounding, in which robots understand how phrases relate to real-world objects or actions, offers a compelling reality in which untrained humans can operate highly sophisticated robots. Current techniques for training robots to understand natural language, however, assume that there is a fixed set of phrases or objects that the robot will encounter during deployment. Instead, the real world is full of confusing jargon and unique objects that are nearly impossible to anticipate and therefore train for. This thesis presents a model called the Distributed Correspondence Graph - Unknown Phrase, Unknown Percept - Away (DCG-UPUP-Away) that augments the state of the art Distributed Correspondence Graph by recognizing unknown phrases and objects as unknown, as well as reasoning about objects that are not currently perceived. Furthermore, experimental results in simulation, as well as a trial run on a turtlebot platform, validate the effectiveness of DCG-UPUP-Away in grounding phrases and learning new phrases.

Thesis Supervisor: Nicholas Roy
Title: Professor

Acknowledgments

Although there are few names listed in this thesis, countless parties have contributed significantly to this work and my brief experience as a graduate student. Of course, I must thank Professor Roy who opened up his lab and schedule to accommodate me. Along those lines, I am profoundly grateful to everyone in the Robust Robotics Group for aiding me in everything from developing a proper problem formulation to finding the best boba tea in Boston. Rohan, in particular, dedicated significant time in trying to transform my haphazard thoughts into something presentable.

I owe my gratitude to my family and friends as well for supporting me whenever I emerged from lab. Without sporadic Skype calls, weekly dinners, or late nights in which I tried to explain my ideas through a mouthful of toothpaste, my sanity would departed long ago.

This work was supported in part by the Robotics Consortium of the U.S Army Research Laboratory under the Collaborative Technology Alliance Program.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Perhaps the defining vision in robotics since its inception as a field has been autonomy — how to make a robotic agent act intelligently and robustly without any guidance from humans. Although this view of autonomy seems to preclude interaction with humans, in fact it requires it. A truly autonomous robot operating in an environment co-habitated by humans must be able to interact with humans in a natural way; one would not argue that people who talk to other people are not autonomous, but someone who must be told how to respond to each individual phrase loses that title. Conversely, ignoring natural language from humans, although technically autonomous behavior, meets autonomy requirements in the same way a rock is autonomous. Therefore, natural language interaction is a necessary requirement for a robot to fulfill the grand promise of autonomous robotics.

Driven by such a vision, considerable work has been done on developing approaches to the "grounding problem." The grounding problem is the association of natural language to real world actions or objects - i.e. the realization that "chair" refers to the structure of wood and a cushion in a room full of clutter. Probabilistic graphical models in particular have yielded impressive results such as training forklifts to drive to pallets when told "pick up the pallet" [54, 29]. Unfortunately, the models used in such techniques often make two assumptions:

1) the solution to the grounding problem will be an association between objects

and phrases the robot has been trained to recognize;

2) the solution to the grounding problem will refer to objects in the robot's known environment (either in a map or in the robot's field of view).

If a robot is to operate in the real world, these assumptions will likely be violated. Simply put, it is nearly impossible to teach a robot all correct groundings before deploying it in the real world. On average, American high school graduates know 12,000 word families while college-educated Americans typically know upwards of 17,000 word families, often including domain-specific terms [11]. Even if a robot were to learn every word in the dictionary, it would also have to recognize every possible object in the world in order to generate the right groundings. Furthermore, a context-dependent lexicon often attaches new meaning to words while specific environments often contain otherwise rarely-found objects. Compounding theses issues, robots rarely have complete information about their environment ahead of time and must therefore be able to reason about objects they cannot currently perceive.

In this work, a probabilistic graphical model is presented in order to reason in environments in which both assumptions are relaxed. This model is called the Distributed Correspondence Graph - Unknown Phrase, Unknown Percept - Away (DCG-UPUP-Away). A previously successful model, the Distributed Correspondence Graph (DCG), is adapted to operate over broader domains to include nouns that have never been encountered in training, objects that have never been perceived before, and hypothesized objects that are not currently perceived. After expanding these domains, the same techniques are used to solve for the most likely grounding as in DCG. The proposed model is implemented, trained, and tested on a turtlebot navigating a simulated environment containing a variety of objects.

1.1 Technical Gap

The fundamental problem that DCG-UPUP-Away addresses is the grounding problem. The correct solution to a grounding problem is the interpretation of natural language into real-world objects and actions. For example, in a room with a chair and a sofa the correct grounding of "drive to the chair" is the execution of a path to the chair. Generally, the natural language phrase is not required to be a command (e.g. "this is a chair"), but in this work only commands will be considered.

First, one must define the domains over which the grounding problem will be solved. In the general formulation of the grounding problem, three variable domains must be considered: Γ, which represents what symbolic actions or objects may be grounded to, Λ, which represents what natural language phrases may be grounded, and Υ, which represents a model of the world in which phrases may be grounded by combining symbolic objects with the properties of each actual object. Different grounding models approximate these three domains in order to leverage simplifying assumptions, so it is useful to define partitions that may be combined to generate the full Γ, Λ, and Υ. These partitions are defined in Table 1.1.

Table 1.1: The 4 partitions for Γ allow for grounding to known or unknown actions or objects. The Λ partitions are divisible into sets of sentences using known or unknown nouns. Finally, the Υ partitions represent perceived and unperceived objects crossed with each object's properties.

symbol	domain definition	example element
Γ_{KA}	{all known symbolic actions}	`drive_to`
Γ_{KO}	{all known symbolic objects}	cube
Γ_{UA}	{all unknown symbolic actions}	`hop`
Γ_{UO}	{all unknown symbolic objects}	cone
Λ_K	{all parseable sentences with known nouns}	"drive to the cube"
Λ_U	{all parseable sentences with unknown nouns}	"drive to the cone"
P	{object properties}	object pose in SE(3)
Υ_P	$\{\{\text{all perceived objects}\} \times P\}$	cube 1 ft. in front of robot facing north horizontal to ground
Υ_U	$\{\{\text{all unperceived objects}\} \times P\}$	cube 1 ft. behind robot facing north horizontal to ground

Given these partitions, one may simultaneously formulate the grounding problem as a probability maximization problem in Equation 1.1 and set the domains over which to solve the maximization problem in Table 1.2.

$$\boldsymbol{\gamma}^* = \underset{\boldsymbol{\gamma} \epsilon \Gamma^{|\boldsymbol{\lambda}|}}{\arg\max}\, P(\boldsymbol{\gamma}|\boldsymbol{\lambda}, \Upsilon) \tag{1.1}$$

Table 1.2: The full domains for Γ, Λ, and Υ, without making any simplifying assumptions, are the unions of all their respective partitions.

$\Gamma = \Gamma_{KA} \cup \Gamma_{KO} \cup \Gamma_{UA} \cup \Gamma_{UO}$
$\Lambda = \Lambda_K \cup \Lambda_U$
$\Upsilon = \Upsilon_P \cup \Upsilon_U$

In such a formulation, $\boldsymbol{\gamma}^*$ is a vector of individual groundings $\gamma \epsilon \Gamma$ for the vector of phrases $\boldsymbol{\lambda}$ composed of individual phrases $\lambda \epsilon \Lambda$. In other words, the grounding problem must find the groundings $\boldsymbol{\gamma}^*$ that maximizes the probability of being the right groundings given a particular phrase $\boldsymbol{\lambda}$ and a model of the world Υ.

While Table 1.2 defines the domains for the general grounding problem, current solutions to the grounding problem restrict Λ and Γ to only contain phrases or groundings that are used in training the grounding model. Furthermore, many (but admittedly not all) grounding systems restrict Υ to only cover the set of perceived objects [54, 29, 19]. These standard language grounding models do not have a principled way of reasoning about unknown groundings, unknown phrases, or objects not currently perceived and therefore only use the domains $\Gamma_{KA} \cup \Gamma_{KO}$, Λ_K, and Υ_P (for known groundings, known phrases, and perceived world, respectively), as shown in Table 1.3.

Table 1.3: Restricted domain definitions often used in current solutions to the grounding problem.

$\Gamma = \Gamma_{KA} \cup \Gamma_{KO}$
$\Lambda = \Lambda_K$
$\Upsilon = \Upsilon_P$

Unfortunately, such restricted domains preclude a large number of real-world scenarios in which phrases or objects are unknown, or phrases refer to objects that are not currently perceived. When models only use the domains from Table 1.3, unknown objects are often completely ignored or categorized as a known type of object (e.g. perceived sofas may be labeled as chairs). Similarly, unknown phrases are either discarded or grounded to the *a priori* most likely grounding regardless of language [29]. Thus, there is a clear gap between existing grounding models that only succeed under special conditions and the promise of a general model that correctly determines what any phrase refers to in any environment. A new language grounding model is presented in this thesis in order to close this gap.

1.2 Problem Statement

This work presents a new model called DCG-UPUP-Away, based on an existing probabilistic graphical model, DCG, to reason over larger subsets of the full variable domains from Table 1.2. First, phrases may include unknown nouns, as long as the full sentence remains parseable. Second, groundings may include unknown objects. Third, the world model includes perceived objects and hypothesized objects that may exist in the world outside the perceived environment. These expanded domains are shown in Table 1.4.

Table 1.4: Domain definitions used for DCG-UPUP-Away.

$\Gamma = \Gamma_{KA} \cup \Gamma_{KO} \cup \Gamma_{UO}$
$\Lambda = \Lambda_K \cup \Lambda_U$
$\Upsilon = \Upsilon_P \cup \Upsilon_U$

Formal definitions of what it means for phrases or objects to be known or unknown will be presented in Chapter 3, but, intuitively, unknowns are things that have not been seen in training. Similarly, hypothesized objects are more formally presented in Chapter 4 but may be thought of, for now, as imaginary instances of objects that may be anywhere in the world.

The problem statement for this work is fully defined by the grounding probability maximization equation, Equation 1.1, and the domains of the variables in Table 1.4. While formulating the problem may be simple, generating solutions to the problem is anything but. For example, whether or not DCG-UPUP-Away has been trained to ground the noun "ewer" (a rarely used synonym for pitcher), "ewer" now falls within the domain of Λ, so DCG-UPUP-Away is expected to correctly ground "ewer" to pitchers. Furthermore, phrases may refer to objects that DCG-UPUP-Away does not even know exist yet: in the same example as before, one may command DCG-UPUP-Away to "pick up the ewer" even if DCG-UPUP-Away does not know what "ewer" means or what pitchers look like and if there is no pitcher in the room.

Such scenarios may be difficult to solve, but they are vitally important to future deployment of autonomous robots. Humans often do not have the time or even ability to train robots to recognize all phrases and objects they may encounter.

1.3 Contributions

This thesis makes 3 main contributions. First, the notions of unknown phrases and unknown percepts are introduced into a language grounding model called DCG-UPUP-Away. DCG-UPUP-Away may therefore reason in more complex environments than the state of the art. Second, DCG-UPUP-Away automatically acquires new concepts through unsupervised learning, enabled in part by recognition of unknowns. Third, hypothesized objects allow DCG-UPUP-Away to reason about objects it cannot currently perceive.

The effectiveness of DCG-UPUP is evaluated both in a simulation and on a real robotic platform. A large-scale study using natural language commands in a simulated environment shows that a robot can correctly ground phrases around 80% of the time while in an environment in which it initially knows fewer than half of the objects present. In addition, a sample trial run on a turtlebot (a simple mobile robot platform) in a laboratory office demonstrates that, with some modifications to object recognition software, DCG-UPUP-Away can be used in the real world.

1.4 Outline

The remainder of this thesis is outlined as follows. Chapter 2 discusses relevant research especially in the field of natural language grounding, but also in human-robot dialog, object hypotheses in unknown environments, and learning techniques in the context of natural language navigation. Chapter 3 presents an intermediate model that builds upon DCG to associate unknown phrases with unknown percepts. Chapter 4 augments that intermediate model by creating hypothetical objects that may be grounded to. Chapter 5 offers a grounding heuristic that allows for more subtle behaviors based on adjectives and object properties. Chapter 6 evaluates the performance of DCG-UPUP-Away in simulation trials and a sample demonstration on real hardware. Chapter 7 concludes with a discussion of the contributions made in this thesis as well as steps for future work.

Chapter 2

Background

The work presented in this thesis builds upon research on the natural language grounding problem. As discussed in the introduction, the grounding problem is formulated as determing the most likely physical meaning for a natural language phrase. For example, when given in the context of a room with a cube and a cone, the phrase "the cube" has a high likelihood of referring to the physical cube itself. (Stating that "the cube" grounds to the cube with 100% probability would be a dangerous assumption — perhaps "the cube" refers to a cone in a different room, or perhaps the person who said "the cube" incorrectly believes that "cube" is the noun for describing cones. It is therefore safer to reason about the probability distribution over the space of possible groundings.) More generally, phrases other than nouns may also be grounded to their physical meanings (e.g. "skipping" to a hopping sort of run), but such examples are harder to describe.

A useful perspective in viewing the grounding problem is through the lens of symbols. Symbols represent semantic entities or concepts in the real world. On the natural language side, phrases (e.g. "cube") are associated with symbols (e.g. a canonical cube type). On the real-world side, percepts (e.g. an RGB image of a cube) are associated with symbols (e.g. a canonical cube type). Such relations are shown in Figure 2-1. Using such a representation appears to simplify the grounding problem significantly: finding the correct grounding for a given phrase merely involves trans-

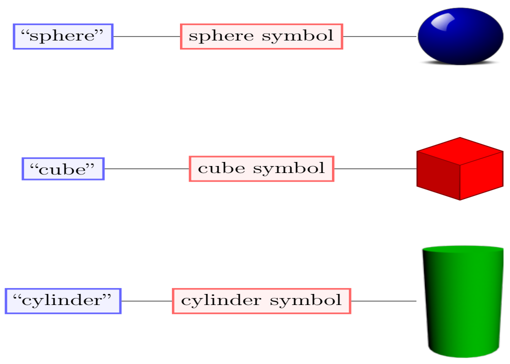


Figure 2-1: Mappings among phrases on the left, symbols in the middle, and percepts on the right.

lating from language to symbols and from those symbols to percepts.

Unfortunately, although such a method may work in theory in a world with one-to-one mappings with symbols, natural language and the real world are too complex for such an approach to work in practice. For example, consider the fact that several words may refer to the same symbol (e.g. “cat” and “feline” both refer to a canonical cat), a single word may refer to multiple symbols (e.g. “dust” refers to a powder of detritus, the act of removing a powder of detritus, or the act of adding a powder like sugar), several percepts may contain the same symbol (e.g. two different photos of a cone), and a single percept may refer to multiple symbols (e.g. a photo containing a cone and a helmet). Thus, Figure 2-1 is better represented with Figure 2-2. Even within a simplified block world, language and perception ambiguity is unavoidable.

Given the complexity of phrase-symbol-percept mappings, a simple and deterministic solution to the grounding problem appears impossible. A probabilistic view, however, elegantly reflects the uncertainty inherent in language. For the remainder of this section, two probabilistic models used to solve the grounding problem will be introduced before discussing related - but less directly relevant - work in natural language robotics.

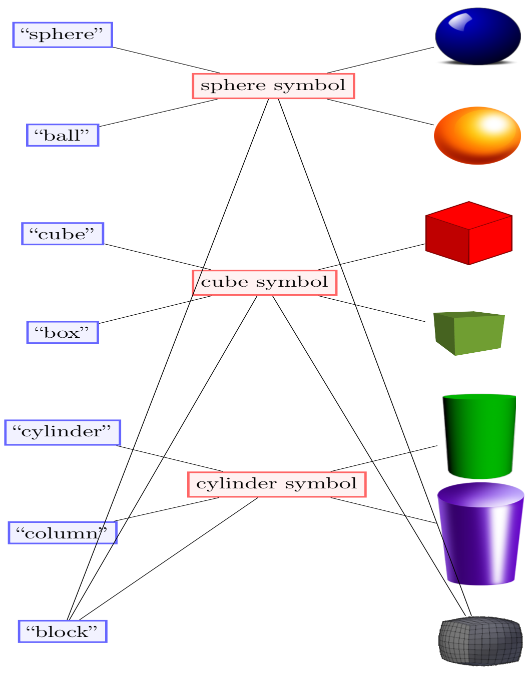


Figure 2-2: A more realistic but complex mapping among phrases on the left, symbols in the middle, and percepts on the right. Note how one-to-one mappings are not guaranteed.

2.1 Grounding with Probabilistic Graphical Models

In any probabilistic view of the grounding problem, the overall equation to be solved remains the probability maximization equation shown in Equation 1.1. The optimal grounding $\boldsymbol{\gamma}^*$ is the set of groundings that maximizes the likelihood of that set of groundings, given language $\boldsymbol{\lambda}$ and a model of the world Υ. By using probabilistic graphical models, the probability maximization problem may be reformulated as inference on a factor graph, structured according to Equation 1.1.

By introducing a binary correspondence variable ϕ that represents whether or not a phrase is paired with the correct grounding, Equation 1.1 may be rewritten as follows:

$$\boldsymbol{\gamma}^* = \underset{\boldsymbol{\gamma}\epsilon\Gamma^{|\boldsymbol{\lambda}|}}{\arg\max}\, P(\boldsymbol{\phi} = \mathbf{True}|\boldsymbol{\lambda}, \boldsymbol{\gamma}, \Upsilon), \tag{2.1}$$

where $\boldsymbol{\phi}$ is a vector composed of $\phi\epsilon\Phi$ for each grounded phrase. Thus, the correspondence variable is set to True for each phrase in the overall command $\boldsymbol{\lambda}$.

In previous work, the domains of the vector elements γ, λ, and Υ have been restricted to known phrases and groundings [54, 29]. Furthermore, although some works reason about unknown environments, many probabilistic grounding techniques assume fully observable worlds [19, 59]. Thus, the domains of the variables in Equation 2.1 are shown in Table 2.1.

Table 2.1: The domains of variables in common solutions to the grounding problem. Groundings fall within the union of known actions (Γ_{KA}) and known objects (Γ_{KO}); phrases must only use known words (Λ_K); the world model only includes perceived objects (Υ_P); correspondence variables are binary.

$\Gamma = \Gamma_{KA} \cup \Gamma_{KO}$
$\Lambda = \Lambda_K$
$\Upsilon = \Upsilon_P$
$\Phi = \{True, False\}$

One way of paraphrasing Equation 2.1 is that the optimal grounding $\boldsymbol{\gamma}^*$ is the vector of groundings such that the probability that the language and the groundings correspond, in the world model, is maximized. If the most likely setting of a given ϕ_i is true, then individual phrase λ_i and grounding γ_i likely correspond; otherwise they likely do not. Fixing ϕ_i to be true for all $i \; \epsilon [1, |\boldsymbol{\lambda}|]$ (for each word), and then searching over the most likely settings of γ_i therefore achieves the goal of finding the most likely correct groundings for the entire command.

If the only assumption one makes is that each word has one grounding, the graphical model representation of Equation 2.1 for the phrase "move to the cube" is the factor graph shown in Figure 2-3. Every grounding γ_i is related to every other grounding γ_j through factor f.

There are several problems with this graph, however. First, learning the factor function f is an extremely difficult problem as it must accept sentences of arbitrary

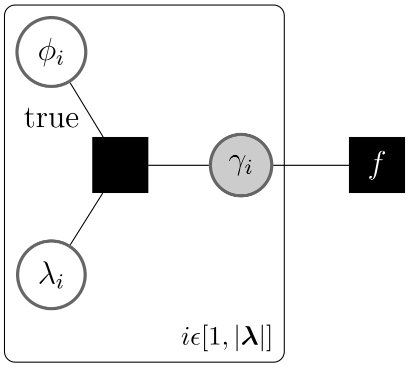


Figure 2-3: Probabilistic graphical model for the grounding problem without any assumptions. For each word λ_i within a command of $|\Lambda|$ words, the correspondence variable ϕ_i is set to true and the most likely grounding γ_i is found.

length. Second, the number of computations required to find the most likely value of $\boldsymbol{\gamma}$ grows exponentially as the length of the commands increases linearly because the grounding of each word depends on the grounding of each other word. It is these two issues that motivate the development of new models that, by making assumptions about conditional independence of groundings, allow for efficient computation of correct groundings.

2.1.1 Generalized Grounding Graphs

The Generalized Grounding Graph (G^3) is a model that addresses both issues [54]. The main insight in G^3 is that the syntactic hierarchy of natural language commands can be used to generate a sparsely-connected graphical model of Spatial Description Clauses (SDCs) [31]. SDCs will be described in more detail later, but for now they may be thought of as 4-element tuples that contain subjects, actions, landmarks, and relations between the subject and the landmark. The hierarchy of SDCs is revealed by examining the parse tree of natural language commands. Parse trees show how words relate and are generated by rules for combining parts of speech into larger phrases or clauses until an entire sentence is composed. A sample parse tree is shown in Figure 2-4.

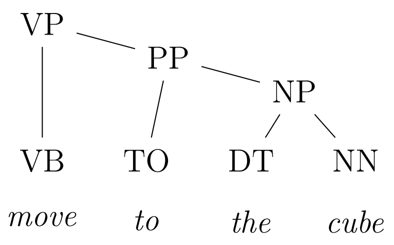


Figure 2-4: Parse tree for the sentence “move to the cube’.’

This parse tree confirms one’s intuition about language: in the command “move to the cube,” the verb “move” means the same thing whether one is told to move to a cube or a sphere. However, exactly what meaning “move” will take depends on whether one should be moving to or away from an object. Thus, grounding “move” ought to depend on the preposition phrase as a whole, but perhaps not the noun directly. In other words, once one understands where the cube is, one must no longer consider what a cube is. Such reasoning is reflected perfectly in the edge structure of the parse tree: “move” and “cube” are conditionally independent given the preposition phrase (PP). In fact, several other papers have exploited parse tree structure for even more efficiency gains and more complex symbol representations [13].

From this notion of conditional independence, it is possible to merge the original, fully-connected grounding graph in Figure 2-3 with the parse tree to generate an efficient grounding graph in Figure 2-5.

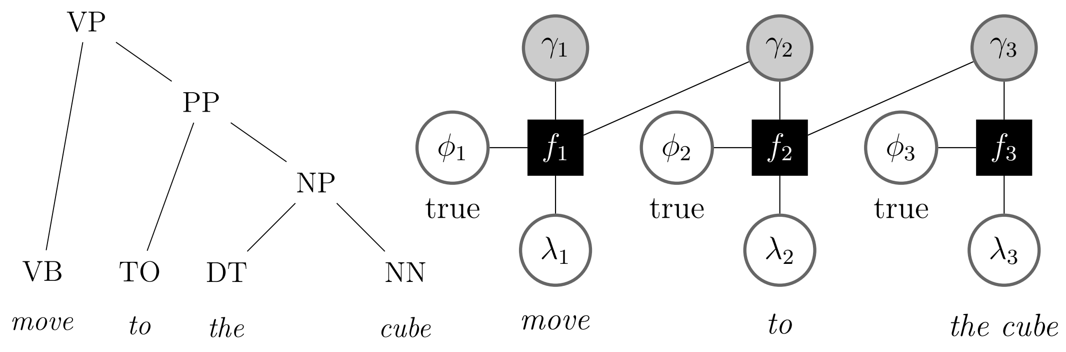


Figure 2-5: G^3 factor graph using the parse tree on the left. The gray γ nodes are unknown, while the white ϕ and λ nodes are set to true and phrases, respectively.

Now, instead of having to simultaneously consider every grounding, computing the optimal grounding for a word only requires local computation about that word and child groundings (dictated by the parse tree). This factored approach is reflected in the final form of the equation solved by G^3, shown in Equation 2.3. The domains for the variables used in Equation 2.3 remain the same as the domains written in Table 2.1 for known phrases, known action and object groundings, binary correspondence variables, and a perceived world model.

$$P(\boldsymbol{\phi} = \mathbf{True}|\boldsymbol{\lambda}, \Gamma, \Upsilon) = P(\boldsymbol{\phi} = \mathbf{True}|SDCs(\boldsymbol{\lambda}), \Gamma, \Upsilon) \tag{2.2}$$

$$= \frac{1}{Z}\prod_i \Psi(\phi_i, SDC_i, \Gamma_{c_i}, \Upsilon) \tag{2.3}$$

The final form of Equation 2.3 states that the probability of a correct grounding equals the product of a feature function Ψ evaluated for every correspondence variable ϕ_i, spatial description clause SDC_i, child groundings Γ_{c_i}, and the world model Υ (all normalized by the partition function Z). This locality — that the overall probability can be computed through the product of many functions that only depend on a single phrase, the SDC for that phrase, and child groundings — is precisely what combats the otherwise exponential growth of naïve approaches.

The feature function Ψ, is a log-linear function that associates binary functions of language (e.g. if a word is present) with binary functions of groundings (e.g. if a box is present). These binary functions (also called features) are combined in a weighted sum according to learned weights $\boldsymbol{\mu}$. The exact form of Ψ, therefore, is shown in Equation 2.4:

$$\Psi(\phi_i, SDC_i, \Gamma_{C_i}, \Upsilon) = \exp\Big(\sum_{f\epsilon F} \mu_f f(\phi_i, SDC_i, \Gamma_{C_i}, \Upsilon)\Big) \tag{2.4}$$

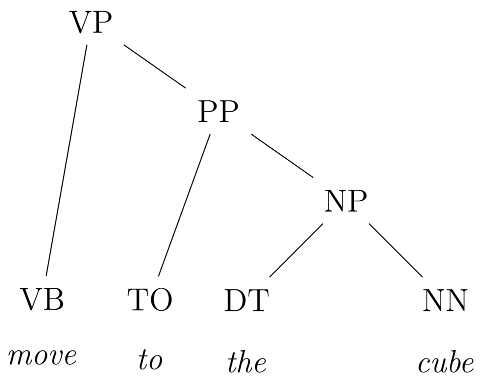
The training procedure for G^3 learns the weights μ_f for each binary feature f from the set of all hand-coded features F. Weights are learned using the Limited-memory Broyden-Fletcher-Goldfarb-Shanno (L-BFGS) by exploiting the convexity of the function Ψ [35]. Thus, the overall Ψ function is merely the exponential of a weighted combination of binary features that return 0 or 1, depending on what phrase or grounding is being used.

2.1.2 Distributed Correspondence Graphs

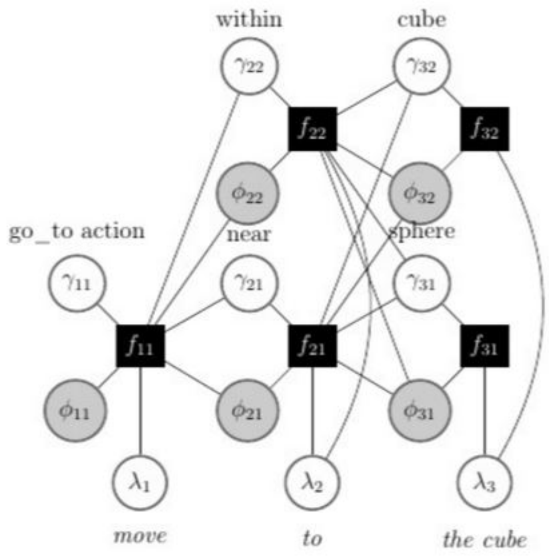
While G^3 formally drastically improves the efficiency of conducting inference on a probabilistic graphical model to solve the grounding problem, such inference on the graph occasionally remains quite slow, especially when phrases refer to groups of objects (e.g. "drive between the cube, the sphere, and the cylinder."). The Distributed Correspondence Graph (DCG) addresses this problem in two ways - by building a larger, but ultimately more efficient, graphical model and inference procedure and by reasoning over constraints instead of paths and objects [29].

First, the structure of DCG evolves from G^3 so that, rather than fixing ϕ to true and then searching over Γ to find the most likely groundings, grounding nodes γ_{ij} may be introduced into the graph, each set to a possible grounding, and then search is conducted over the most likely settings of ϕ_{ij}. The final DCG model shown alongside the same parse tree for the command "move to the cube" is shown in Figure 2-6.

Although the graphical model may appear complex, it is actually quite simple. Each parsed phrase appears in a λ_i node at the bottom of the graph. Each λ_i node is connected to a factor linking the phrase to a correspondence variable ϕ_{ij} and a grounding variable γ_{ij}. Each column represents possible groundings for the phrase; in this simplified example, the domain of possible actions is restricted to a single `go_to` action, the domain of preposition groundings is restricted to `within` or `near`, and the domain of noun groundings is restricted to `cube` or `sphere`. Finally, the edges between columns, going from each factor in one column to each ϕ_{ij} and γ_{ij} in the



(a) Parse tree for the sentence "move to the cube."



(b) DCG factor graph for the command "move to the cube." Unknown correspondence variables ϕ_{ij} are gray; known groundings γ_{ij} and phrases λ_i are white.

Figure 2-6: DCG graphical model with inter-column structure generated from the parse tree on the left

next column, represent the reliance on child groundings. In this case, because the parse tree is binary, each column (except the rightmost one) has exactly one child grounding, so there are only edges between adjacent columns.

The structure of DCG is further revealed by recreating the original graphical model in plate notation, as in Figure 2-7. The similarities between G^3 and DCG are further revealed by the minimal structure. Of particular note, is the domain of each γ_{ij}, as each plate is repeated by letting γ_{ij} range over Γ.

Before defining the domains of the variables in the DCG graph, however, it is important to explore the second idea in DCG: reasoning over constraints rather than objects allows for extremely efficient inference when considering sets of objects. Recall that the entire reason for G^3 is that it breaks the exponential computational costs of a fully connected grounding graphical model. While G^3 achieves this goal most of

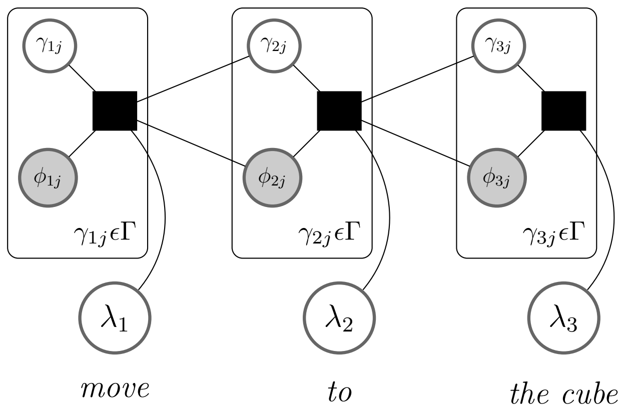


Figure 2-7: DCG graphical model redrawn in plate notation.

the time, it fails when considering sets of objects (e.g when grounding the command "go between the cube and the sphere"). As the number of related objects grows, the G^3 computational costs increase exponentially.

Therefore, DCG replaces the domains of the grounding variables in G^3 (previously the set of all known actions and objects) with a set of constraints. A constraint is formally defined as a function of the robot state, a fixed input, and time interval that returns true if the state matches the input during the specified time interval and false otherwise. For example, a constraint might specify that the robot must be within 1 meter of a cube between 5 and 10 seconds from now. Now that DCG grounds to constraints rather than specific paths or objects, grounding and planning have been effectively split into two problems: first DCG determines what constraints are encoded in the language command, then a planner may accept those constraints and generate a series of actions that satisfy all the constraints. In fact, the idea of dividing natural language planning into a grounding phase followed by a planning phase has subsequently appeared in several other papers for robotic navigation or recipe-following [37, 5, 9].

It is now finally possible to formally define the problem DCG tries to solve and the domains of the variables it uses. Recall that DCG fixes grounding variables and solves for correspondence variables, so Equation 2.1 from G^3 may be transformed to

Equation 2.5:

$$\boldsymbol{\phi}^* = \underset{\boldsymbol{\phi} \epsilon \Phi^{|\boldsymbol{\lambda}|}}{\arg\max} \, P(\boldsymbol{\phi} | \boldsymbol{\lambda}, \boldsymbol{\gamma}, \Upsilon). \tag{2.5}$$

The equation can be paraphrased as saying that the most likely correspondence variables $\boldsymbol{\phi}$ can be found by finding the correspondence variables that maximize the likelihood of phrases and groundings, given a model of the world. By using the same factorization trick from G^3 assuming independence of child groundings, Equation 2.5 may be rewritten as Equation 2.6, where Equation 2.7 is used to emphasize that DCG uses the same system of learned log-linear feature functions to compute the most likely correspondence variables.

$$\boldsymbol{\phi}^* = \underset{\boldsymbol{\phi} \epsilon \Phi^{|\boldsymbol{\lambda}|}}{\arg\max} \prod_{i=1}^{n} p(\phi_{ij} | \gamma_{ij}, \lambda_{ij}, \Gamma_{c_{ij}}, \Upsilon) \tag{2.6}$$

$$= \underset{\boldsymbol{\phi} \epsilon \Phi^{|\boldsymbol{\lambda}|}}{\arg\max} \frac{1}{Z} \prod_{i} \Psi(\phi_{ij}, \gamma_{ij}, \Gamma_{c_{ij}}, \Upsilon) \tag{2.7}$$

Formal definitions of the domains used in DCG for solving these equations are shown in Table 2.2.

Table 2.2: The domains of variables in DCG. Groundings fall within the space of pre-defined relations (e.g. near, within, etc.) crossed with known objects; phrases must only use known words; the world model only includes perceived objects; correspondence variables are ternary.

$R = \{\text{set of hand-coded relations}\}$
$\Gamma = R \times \Gamma_{KO}$
$\Lambda = \Lambda_K$
$\Upsilon = \Upsilon_P$
$\Phi = \{Active, Inverted, Inactive\}$

Note how the domain of groundings has been changed from actions and objects to constraints. Because of this change, the correspondence variables have in turn been transformed from binary to ternary variables that can take on the values of Active, Inactive, or Inverted. Setting ϕ_{ij} to active in DCG is much like setting ϕ_i to true in G^3; it means the constraint is correctly grounded. For example, the phrase "near the cube" has an active correspondence with the region near a cube. Conversely, the phrase "far from the cube" has an inverted correspondence with the region near the cube because "near" and "far" are antonyms. Finally, correspondence variables are set to inactive if the language has no bearing on a particular constraint (e.g. "near the cube" has no relation to the constraint of being near a sphere as well).

In this work, DCG-UPUP-Away will continue to modify the structure of the DCG graph by expanding the domain of possible phrase to include unknown nouns and the domain of possible groundings to include unknown objects. Furthermore, the world model will be adapted to contain hypothetical objects that may exist outside the robot's field of view. Finally, additional binary features will be introduced to the log-linear model so that DCG-UPUP-Away may reason intelligently within the expanded domains.

2.2 Human-Robot Dialog

While DCG-UPUP-Away builds most directly on G^3 and DCG, other work in natural language robotics has guided the work in this thesis. Research into human-robot natural language dialog and question-asking indicates a broad area for future improvements to DCG-UPUP-Away. Instead of silently executing commands, as is done in traditional language grounding research, a natural extension is having robots that talk back. For example, should a human issue an ambiguous command (e.g. "pick up the bottle" when facing two bottles), a robot might ask for more information rather than silently executing a low-confidence grounding.

Such scenarios are considered in [46]. Ros et al. broadly approach resolving language ambiguity using two techniques. First, a robot attempts to model the human's perspective on the scene to determine which objects may be visible to the human. This technique relies on insights from child development studies that show how children employ such reasoning on their own and has been successfully used in other robotics literature [39, 40, 56, 55]. Importantly, this first strategy requires no additional dialog. The second strategy, however, relies on the robot asking a human for more information. For example, the robot may ask for spatial relations or object features in distinguishing between objects. Choosing exactly which question to ask, of course, requires reasoning about what information best discriminates among potential groundings.

Deits et al. explore precisely that issue of how to choose which questions to ask in [18]. As in much research in robotic question-asking, a balance must be struck between too many questions and not enough questions while simultaneously determining what sorts of question to ask [22, 48]. The authors in [18] use the entropy of the probability distribution over groundings to estimate the grounding uncertainty. Higher entropy, therefore, leads to more questions. As expected, asking questions leads to a greater grounding accuracy rate for all sorts of questions asked.

While the above-mentioned research largely uses hand-coded templates for generating questions, other work explores the topic of learning how to compose questions automatically. One system, CEDERIC, mimics human learning and uses only two bases cases to ask questions as ranging from confirmations (e.g. "Is this view ok?") to specifications (e.g. "I have several schools to choose between. Which one do you mean?") [20]. Conversely, significant research has been conducted on generating natural language statements instead of questions. Such systems that use inverse semantics have a broad range of applications including directing humans to their destination, explaining what a robot is about to do, and automatic translation [17, 51, 36].

Such research provides a clear path for future improvements upon DCG-UPUP-Away, which offers no dialog. In fact, [18] interfaces nicely with G^3 and thus may presumably be easily adapted to operate with DCG or DCG-UPUP-Away.

2.3 Learning Semantic Meanings

Setting aside actual human-robot interaction, a large corpus of research focuses entirely on techniques for learning the semantic meanings of language; presumably once a robot learns what language means, grounding such language becomes much easier. Much of the work in this field draws heavily on observations of children, who exhibit prodigious learning rates of 1 new word every 3 days when they are only 4 months old to 12 words a day when they are 8 years old [21, 2]. Furthermore, such learning appears to require minimal structured adult supervision; children quickly convert few experiences into more general knowledge (e.g. children do not need to see thousands, or even hundreds of cats before knowing what a cat is) [8, 15, 43, 27]. This indicates that robots may eventually be able to perform similarly and learn new concepts in a semi-supervised or unsupervised manner.

One common approach for autonomous language learning provides a robot with semantic representations of the world that must be associated with language. Such associations may be formally expressed using predicate logic, but ultimately the problem of language acquisition is reframed as a mapping problem from words to pre-defined semantics [14, 49, 61, 23]. Unfortunately, hand-labeled representations necessarily require intensive human involvement in generating training data [41]. As a result, other work takes the opposite approach and tries to associate words directly to objects or actions without creating formal symbolic representations. For example, using online raw video data and sentences, a system is able to learn shape categories without being told ahead of time that four right angles define rectangular objects [60, 47].

Finally, other works combine ideas from both approaches to cluster raw sensor data into semantic units that are in turn associated with language [12, 25, 38]. These models appear particularly impressive. For example, the system developed in [25], called the Transportable Word Intension Generator (TWIG) manages to learn prepositions such as "above" and "below" and deitic pronouns like "I" and "you." This system, like DCG-UPUP-Away, performs best when provided with training examples that include one or fewer unknown words.

Although this work is extremely impressive, its domain is restricted to learning through natural language statements as opposed to commands. For example, a robot may be shown a cone and told “this is a cone,” but the robot will not be expected to approach a cone when told “go to the cone.” Thus, the robot always remains in a sort of learning mode (or switches between mutually exclusive learning and execution modes) as opposed to DCG-UPUP-Away, which attempts to simultaneously learn and execute language commands.

2.4 Grounding Spatial Relations

Rather than trying to solve the general language grounding problem or teach robots arbitrary statements, some researchers have focused in particular on the language of spatial relations. Spatial relations are ubiquitous in language — not only are they directly applicable to language directions such as “go to the cube to the left of the sphere,” but they are often used in more abstract thoughts as well (e.g. “their ideologies fall on opposite sides of the spectrum”).

Early research by Jackendoff focuses on explicit categorization of all spatial relations by examining prepositional phrases, verbs, and actions [30]. Building upon such work, the fields of linguistics and artificial intelligence have closely examined how spatial relations in language may shed light on internal representations of the world [33, 34, 28]. Only a handful of elementary spatial relations, it is argued, are needed to completely represent any natural language describing the relative position of two objects. Although such works provide an elegant model for understanding how humans may think about spatial language, they do not directly translate the theory into practical behavior for robotics. In fact, some research indicates that humans use spatial relations differently when addressing other humans as opposed to robots [42].

In order to directly address this gap, others apply the idea of composing simple spatial relations to real-world robots [1]. For example, a mobile industrial robot is given primitive relations (e.g. above, below, left, right, at, near, etc.) and manages to combine the relations to parse more complex statements such as "near the block on the left." Furthermore, the system is able to translate its symbolic understanding of these relations into a potential field representing where, in metric space, an object is likely to be located [52]. Other work combines spatial relations with inverse semantics to allow a robot to describe its environment using to a human in natural language by saying things like "there is an object on the right" [50].

Finally, some research explicitly couples spatial relations to the grounding problem. In fact, the predecessor to the probabilistic graphical models G^3 and DCG that motivates this work relies heavily on Spatial Description Clauses (SDCs) [31]. Natural language commands are executed by first parsing the language into a hierarchy of SDCs, and then finding a sequence of actions that best satisfies the SDCs. Each SDC alone is relatively simple: it is composed of a figure (the subject of the sentence), a verb (the action), a landmark (an object), and a spatial relation (a relation between the figure and the landmark). Recently, SDCs, and spatial relations more generally, have been used to adapt DCG or build entirely new graphical models [44, 32]. The modified DCG model, for example, manages to ground abstract concepts like "the row on the left" when used facing collections of scattered blocks on a tabletop [44].

Ultimately, the work on spatial relations in natural language illustrates a different approach to the grounding problem than the approach used here. Although this thesis explicitly does not investigate phrases with spatial relations, one could clearly incorporate some of the tools from this field to disambiguate between objects and improve the robustness of DCG-UPUP-Away.

2.5 Object Hypotheses for Unknown Environments

Another field of study closely tied to the field of language grounding explores the topic of inferring information about unknown environments using language. For example, if a robot is placed in a room and told to “pick up the ball just outside the door,” a robot should be able to easily deduce that a ball is located near the door. A more formal framing of that intuition indicates that grounding systems should reason over a belief distributions over objects rather than a deterministic known map.

Duvallet et al. use such a framework to propose a latent map that is partially observed by language [19]. Thus, in the example of a ball outside the door, the phrase “pick up the ball outside the door” generates a region of high probability near the door and low probability further away. Sampling from this distribution, as well as updating the distribution as more observations are made, yields a useful map to plan in. Similarly, other work generates distributions over maps or exactly placing objects in unknown environments if their locations are uniquely described [57, 59].

In the absence of localizing language, however, these sophisticated techniques provide relatively little information about where an object may be located. Therefore, because this research is restricted to simple phrases that lack spatial relations, DCG-UPUP-Away uses a far simpler technique for hypothesizing objects in unknown locations. Future development, however, could incorporate some of the methods presented for more intelligent reasoning about unperceived objects.

Chapter 3

Grounding to Unknowns

ἔοκια γοῦν τούτου γε σμικρῶ τινι αύτῶ τούτω σοφώτερος εἶναι, ὅτι ἅ μὴ οἶδα ούδὲ οἶομαι είδέναι

"Indeed I seem to be wiser than him in this little matter in that I don't think I know what I don't know"

—Plato's Apology

The DCG model developed by Howard et al. efficiently grounds some phrases, but it is only capable of reasoning about objects and phrases that the system has already been trained on [29]. For example, if DCG is trained with the phrase "chair" and recognizes chairs in the real world, it can subsequently operate in an environment with chairs. DCG is unable, however, to intelligently reason about new phrases or percepts in the absence of relevant training. In the same scenario as previously described (DCG has been trained with chairs, but only chairs), if a robot is placed in a room full of chairs and a single sofa and is told "drive to the sofa," an empty grounding set is returned. In other words, because DCG cannot reason about the phrase "sofa," it returns that "sofa" does not ground to anything. Given the context, however, any human would be able to deduce that "sofa" refers to the only unknown object in the scene - the sofa.

In this chapter, the DCG model is extended to DCG-UPUP (short for Distributed Correspondence Graph - Unknown Phrase Unknown Percept). The main technical insight is that explicitly recognizing when phrases or percepts are unknown allows for intelligent learned behavior in new scenes. The remainder of this chapter is divided among recognizing when phrases are unknown, recognizing when percepts are unknown, training the model to associate unknown phrases to unknown percepts, and permanently learning new symbols. By the end of chapter, DCG-UPUP will be able to handle exactly the chair-sofa DCG failure case described previously.

3.1 Unknown Phrases

A necessary step in learning to associate unknown phrases with unknown percepts is first recognizing when a phrase is unknown. For the sake of simplicity, development is limited to recognizing unknown nouns (extensions to verbs will be discussed in the future work section and has been briefly examined in [10]). If DCG-UPUP is able to recognize when a noun is unknown, it will have significant insight into what the noun should ground to (specifically, not an object it already knows). Just like a high-schooler taking a multiple-choice vocabulary quiz when he recognizes all the words except for one, DCG-UPUP can use a learned process of elimination to ground unknown nouns to unknown objects. None of this is possible, however, without first recognizing when nouns are unknown.

Before immediately recognizing unknown nouns, one must define what it means for a phrase to be unknown. In the context of the grounding problem, a natural definition arises based on whether or not the system has been trained on how to ground that noun. Thus, if the system has been trained to ground a phrase to a known type, then that specific phrase is considered known – otherwise the phrase is labeled unknown. The subtlety of requiring known nouns to ground to a known type (as opposed to merely appearing in the training examples anywhere) comes from the necessity of including training examples in which unknown nouns ground to unknown objects. The nouns used in such examples should not subsequently be classified as known words.

Keeping track of which nouns are known, based on the above definition, is straightforward. The training procedure for the LLM (the log-linear model used for DCG) is augmented to also generate a set of all the nouns seen in training that were grounded to a known object type. (For now, the requirement that the phrase grounds to a known type may seem odd given that no unknown types have been introduced yet, but that will change in the next section.) The built up set of known phrases, henceforth referred to as κ, therefore is a snapshot at any given moment of exactly which nouns are known and may be thought of as part of the world model Υ. Note that κ grows over the course of the LLM training, so a phrase that is unknown at the beginning training may be known by the end.

For the purposes of causing DCG to respond to unknown nouns, the final step in recognizing a noun as unknown is adding a feature to the LLM that returns true when an unknown phrase is used. Using the definition of unknown phrases and the construction of the κ set, one may easily see that the f_{uphrase} feature should fire (i.e. return true) precisely when a noun is not found within κ. Therefore, when the feature is first created, it accepts κ as an argument that it keeps track of. Later, when training the LLM, f_{uphrase} is queried with a phrase, at which point the feature returns true if the noun is within κ and false otherwise. This new feature - f_{uphrase} - is similar to many of the other features already used to train the LLM; from a theoretical standpoint, however, it is the only feature that is self-reflective, in a sense, and requires a model of its own intelligence. The formal definition of f_{uphrase} is shown in Equation 3.1:

$$f_{\text{uphrase}}(\phi_{ij}, \gamma_{ij}, \lambda_i, \Gamma_{c_{ij}}, \Upsilon) = \begin{cases} 0, & \text{if } \lambda_i \in \kappa \\ 1, & \text{else} \end{cases} \tag{3.1}$$

3.2 Unknown Percepts

In parallel to detecting when a phrase is unknown, DCG-UPUP must recognize when a percept is unknown. Initially, the raw data used to perceive the world are color images taken by a robot. Generally, though, these data can be anything (e.g. sonar readings as well) as long as it is possible to translate the data into an intermediate stage in which object types and locations are recorded. In this context, a single percept is defined as an instantaneous estimate of the object type and location; it is assumed that it is possible to generate such percepts from raw data. Therefore, recognizing unknown percepts becomes the simpler task of detecting unknown objects rather than reasoning over the raw data directly.

Once again, before attempting to recognize unknown percepts, one must first define what it means for a percept to be "unknown." Just as phrases are defined as unknown if they never appear in training examples, one may similarly say that objects of a type that never appeared in training data are unknown. For example, if an object recognizer has been trained to recognize cubes, spheres, and cylinders, any instance of a cube, sphere, or cylinder would be known but a cone would be unknown. Of course, such a definition merely translates the problem of deciding whether an object is unknown into the problem of deciding how well it matches a known category. There are myriad ways to quantify how well objects match known categories: support vector machines (SVMs) report distances from boundaries while other multi-class detectors report probability distributions over known classes [26, 7]. Generally, the field of novelty detection — determining when data belong to a new class instead of an existing one — remains an active field of research [6, 45].

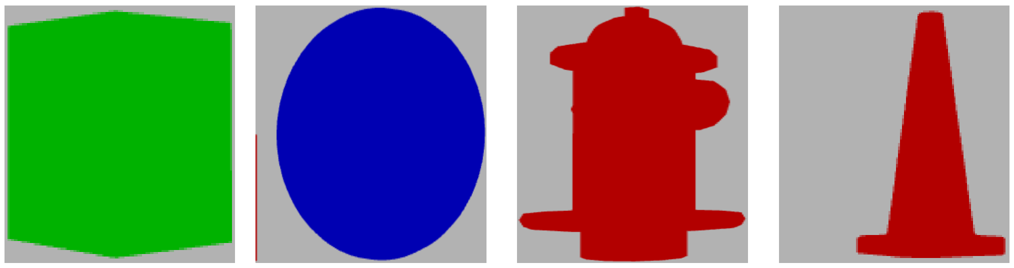
In this particular implementation, an image classifier based on the Histogram of Orientated Gradients (HOG) is used as part of a three-step method [16]. First, an object proposal toolbox from [62] generates tight bounding boxes around objects in an RGB image of the scene. In theory, any classification-agnostic object proposal software may be used, including [58] or [24]. Second, each bounding box is input into a standard SVM classifier offered by MATLAB that uses HOG features as well as the aspect ratio of the bounding box. Third, given the reported distances from the boundaries in the SVM, if the classifier appears uncertain (as defined later in this section), the object is labeled as "unknown" – otherwise the most likely classification is used.

The confidence of the classifier is approximated by examining the difference between the signed margin of the most likely classification and the second most likely classification. Since the SVM is a discriminative model among non-overlapping classes, the first margin is always positive and the second is always negative. (Using mutually exclusive classes is motivated in part by [25] which uses decision trees to represent phrase meanings.) Thus, objects that fall squarely within a category generate large differences while objects near classification borders yield small differences. Comparing this difference to a threshold allows one to say whether or not an object is unknown. As the number of classes grows, however, classification margins necessarily shrink, so some method of normalizing the threshold by the number of categories is required. The general form of the final classification equation used in this thesis is shown in Equation 3.2:

$$\chi = \begin{cases} \text{"unknown"}, & \text{if } m_{\text{diff}} < \frac{m_{\text{thresh}}}{(n-1)^{Z_{\text{exp}}}} \\ \chi_{\text{svm}}, & \text{otherwise} \end{cases} \tag{3.2}$$

where χ is the final classification, n is the number of classification categories, m_{diff} is the difference between the margins of the best and second-best classifications, and χ_{svm} is the classification that maximizes the margin from the SVM. Finally, by playing with intercept m_{thresh} and normalization parameter Z_{exp} until sample cases pass, one may generate the actual equation used for determining the unknown cutoff.

In this research, the best values for m_{thresh} and Z_{exp} were found to be 0.6 and 1.2, respectively. These numbers were determined though cross-validation of training data; because object recognition is not the emphasis of this research, theoretical motivations for these values are left to future research. The values for m_{thresh} and Z_{exp} were generated by systematically testing the classifier with several images of both known and unknown objects. Figure 3-1 shows sample test images of a cube, sphere, and fire hydrant. Below each image are the classification margins generated by a classifier trained only on cubes, spheres, and cylinders.



(a)

cube	**0.2281**
cylinder	-0.2281
sphere	-0.5657

A test image of a cube, with margins.

(b)

cube	-0.2959
cylinder	-0.6862
sphere	**0.2959**

A test image of a sphere, with margins.

(c)

cube	-0.0584
cylinder	-0.3859
sphere	**0.0584**

A test image of a fire hydrant (unknown), with margins.

(d)

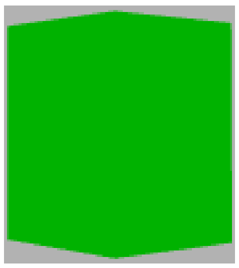
cube	-0.0443
cylinder	-0.1274
sphere	**0.0443**

A test image of a construction (unknown), with margins.

Figure 3-1: Sample generated test images. The bold classifications are the most likely known classification type. Note how the unknown fire hydrant and cone have very small margins, resulting in a final classification of "unknown."

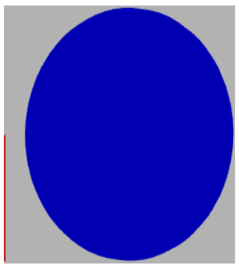
A new image classifier, trained on cubes, spheres, cylinders, and hydrants results in different behavior: this time the fire hydrant is classified as a known type, but the cone remains unknown. Classification results and margins for that classifier are shown in Figure 3-2.

The final modification made to the object recognizer was adding color recognition. Because the HOG classifier is trained using color-invariant features, only the object type (e.g. "cube") is reported. Given the simplicity of the simulated environment, however, it is easy to compute the mean RGB values across the bounding box of each object and report the greatest color value. Thus, the final object classifier returns the object type as well as color. The utility for color as a grounding heuristic will


(a)

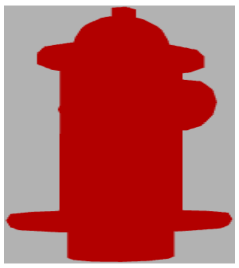
cube	**0.1160**
cylinder	-0.1160
hydrant	-0.4624
sphere	-0.2915

A test image of a cube, with margins.


(b)

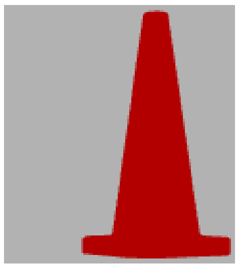
cube	-0.1856
cylinder	-0.3956
hydrant	-0.3853
sphere	**0.1856**

A test image of a sphere, with margins.


(c)

cube	-0.2775
cylinder	-0.4418
hydrant	**0.2497**
sphere	-0.2497

A test image of a fire hydrant, with margins.


(d)

cube	-0.0435
cylinder	-0.0730
hydrant	-0.0852
sphere	**0.0435**

A test image of a construction (unknown), with margins.

Figure 3-2: Sample generated test images. The bold classifications are the most likely known classification type. The classifier has been trained on cubes, spheres, cylinders, and hydrants, so only the cone is unknown.

be discussed in detail in Chapter 5, but the intuition is that color allows for DCG-UPUP-Away to differentiate between two objects of the same type, as long as they are different colors. This is particularly useful for differentiating between two unknown objects of different colors.

3.2.1 Alternate Approaches to Object Recognition

It is important to emphasize that, the overall approach presented in this thesis is agnostic to which particular object recognizer is used. Thus, while a specific object recognizer and unknown-detection mechanism have been proposed here, any object recognizer that can recognize objects and report when they are unknown is acceptable. In fact, an object recognizer based on Speeded Up Robust Features (SURF) was first used in DCG-UPUP [4]. That particular recognizer returns a probability distribution over the object classes to indicate the most likely object type. When developing with the SURF-based recognizer, rather than using the difference of the top-two confidences to detect unknowns, the normalized entropy of the distribution was used, where normalized entropy is defined in Equation 3.3 as follows:

$$\text{norm\_entropy} = \frac{-1}{\text{num\_classes}} \sum_{i=1}^{\text{num\_classes}} p(class_i) * log_{\text{num\_classes}}(p(class_i)). \quad (3.3)$$

When the SURF object recognizer is given an object that does not clearly belong to a known category, the probability distribution tends to approach the uniform distribution across all categories. Thus, the entropy increases, likely indicating an ambiguous and therefore unknown object. While either one of the changes to the object classifier (using SURF features instead of HOG features and using a probability distribution instead of classification margins) could have been used, neither seemed to distinguish well between objects when using fewer than 10 training images per class, as is done in this thesis.

The focus of this research was on a model that assumes a perfect perception system that returns object classifications (whether the true object class or “unknown”). In fact, while evaluating the full DCG-UPUP-Away system, the majority of the failure cases arose from improper object classification. As a result, the majority of experimental results were generated using manual object recognition — humans, rather than an object classification program, labeled objects for use by DCG-UPUP-Away. Thus, there is potential to explore alternative features or classification frameworks for an automated object classifier to make DCG-UPUP-Away completely autonomous.

3.3 Training Unknown Phrase - Unknown Percept Associations

Recall that the goal of DCG-UPUP is to learn that unknown phrases often refer to unknown objects, much like how humans use process of elimination in unfamiliar situations. Thus, it is insufficient merely to recognize phrases and percepts as unknown; the DCG-UPUP model must be trained to ground unknown phrases to unknown objects. Although this may appear to be a daunting task, nearly all the necessary framework is already in place.

First, we must introduce a notion of an unknown symbol. Much like the symbol for cubes relates phrases (e.g. "cube" and "cuboid") to percepts (e.g. images of cubes), the unknown symbol relates unknown phrases to percepts of unknown objects. Concretely, this is done by associating a feature that fires where an unknown phrase is detected with a feature that fires when an unknown object is detected. The necessary work for these features has already been presented, however. For one thing, f_{uphrase} already detects when phrases are unknown. Secondly, the original version of DCG already uses features that detect what object type is being grounded to. All that is necessary, therefore, is to train the association between f_{uphrase} firing and a feature for object type "unknown_type". This trained association defines the unknown symbol.

In total, 7 positive training examples were added to the 40 existing positive training examples to train the unknown_phrase - unknown_percept association. (Negative training examples are generated automatically by changing the correspondence variable settings, as is done in DCG.) Each example uses a new word that has never been grounded to a known type (e.g. "hammer," "mallet," "bottle," and "crate"). Within each training example, instead of the noun grounding to a known data type such as cube_type, the noun is grounded to unknown_type. Crucially, by varying which unknown noun is used but continuing to ground to the unknown_type, the LLM learns that the exact phrase does not matter - only whether or not the noun appears in κ, the set of known words.

These changes alone are enough to solve the sofa and chairs problem presented at the beginning of this section. In brief, the situation involves a robot that has only been trained with chairs but is facing a sofa and several chairs being told "go to the sofa." DCG-UPUP behaves exactly as desired:

1. The object detector segments out the sofa and chairs from its input image;

2. The object classifier returns that each chair is a chair and the sofa is an unknown object;
3. The f_{uphrase} feature returns that the noun “sofa” is unknown;
4. The trained DCG-UPUP model recognizes that unknown phrases are often associated with unknown objects and returns that “sofa” is referring to the sofa.

DCG-UPUP has learned the notion of unknown symbols. Unknown phrases are associated with unknown percepts, exactly as desired. There remain, however, limitations to this result: DCG-UPUP assumes that phrase only refer to objects that are currently perceived, and there is currently no way to differentiate between two unknown objects (e.g. a cone and a fire hydrant).

3.4 Technical Changes

Although a high level view of DCG-UPUP has been presented, the real difference between DCG and DCG-UPUP is revealed by looking at the domains of the variables in the new DCG-UPUP graphical model. First, it appears that the underlying equation that both DCG and DCG-UPUP are trying to solve remains the same.

Table 3.1: Identical equations are used for DCG and DCG-UPUP

DCG	$\boldsymbol{\phi}^* = \underset{\phi_{ij} \epsilon \Phi^{\lvert\boldsymbol{\lambda}\rvert}}{\arg\max} \prod_{i=1}^{n} p(\phi_{ij} \mid \gamma_{ij}, \lambda_{ij}, \Gamma_{c_{ij}}, \Upsilon)$
DCG-UPUP	$\boldsymbol{\phi}^* = \underset{\phi_{ij} \epsilon \Phi^{\lvert\boldsymbol{\lambda}\rvert}}{\arg\max} \prod_{i=1}^{n} p(\phi_{ij} \mid \gamma_{ij}, \lambda_{ij}, \Gamma_{c_{ij}}, \Upsilon)$

The crucial difference is the domains for λ_i, γ_{ij}, and ϕ_{ij}. In the Table 3.2, the domains for language and groundings are shown to illustrate the changes made to DCG.

In addition, one may see how the two additional features for detecting unknown phrases and unknown objects are incorporated into the LLM. Recall that DCG uses

Table 3.2: Variable and domain definitions for DCG-UPUP. Unknown objects (Γ_{UO}) and phrases with unknown nouns (Λ_U) are now permitted.

$\Gamma = \Gamma_{KA} \cup \Gamma_{KO} \cup \Gamma_{UO}$
$\Lambda = \Lambda_K \cup \Lambda_U$
$\Upsilon = \Upsilon_P$
$\Phi = \{True, False\}$

a log-linear model that combines many binary features to associated phrases with objects. The main equation for DCG-UPUP may thus be rewritten, as was done for DCG, as a product of Ψ, the log-linear function that is learned, for each possible grounding. This time, instead of only using the features from DCG, though, Ψ includes the two extra unknown features. This change is reflected in Equation 3.5.

$$\boldsymbol{\phi}^* = \underset{\boldsymbol{\phi} \epsilon \Phi^{|\boldsymbol{\lambda}|}}{\arg\max} \frac{1}{Z} \prod_i \Psi(\phi_{ij}, \gamma_{ij}, \Gamma_{c_{ij}}, \Upsilon) \tag{3.4}$$

$$\Psi = \exp\Big(\sum_{f \epsilon F_{\text{DCG}}} \left[\mu_f f(\phi_{ij}, \gamma_{ij}, \lambda_i, \Gamma_{c_{ij}}, \Upsilon)\right] + \sum_{f \epsilon F_{\text{Unknown}}} \left[\mu_f f(\phi_{ij}, \gamma_{ij}, \lambda_i, \Gamma_{c_{ij}}, \Upsilon)\right]\Big) \tag{3.5}$$

Finally, one may assemble the probabilistic graphical model for DCG-UPUP, shown in Figure 3-3, using the expanded domains. The possibility for grounding to an unknown object appears in the graph as an extra plate on top of the rightmost column, containing the noun.

By varying the natural language command and expanding the plate notation from Figure 3-3, one may see how DCG-UPUP is solved for both known and unknown nouns in Figure 3-4. In this scenario, DCG-UPUP has been trained with cubes, spheres, and cylinders. The robot faces a cube, sphere, and cone, which is recognized as unknown.

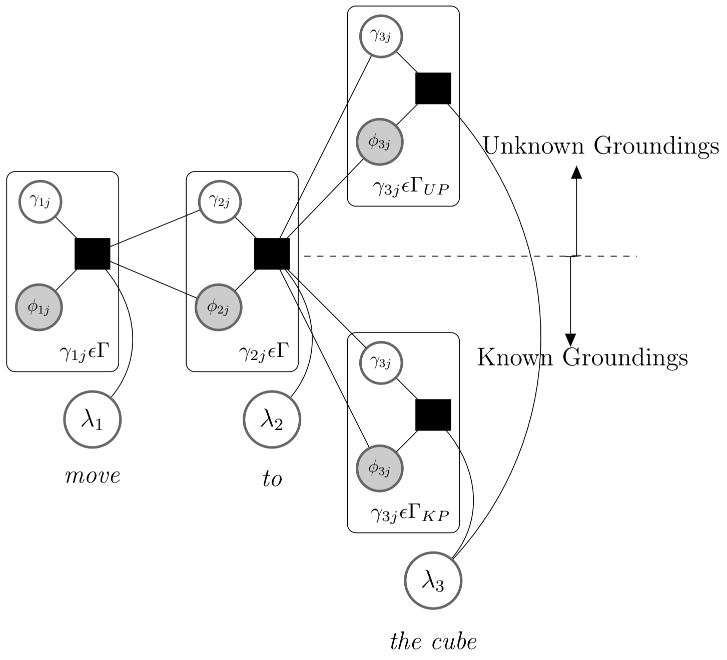


Figure 3-3: The DCG-UPUP probabilistic graphical model. Note how the domain of possible object groundings has been expanded to include unknown perceived objects (Γ_{UP}) in addition to known perceived objects (Γ_{KP}) from before.

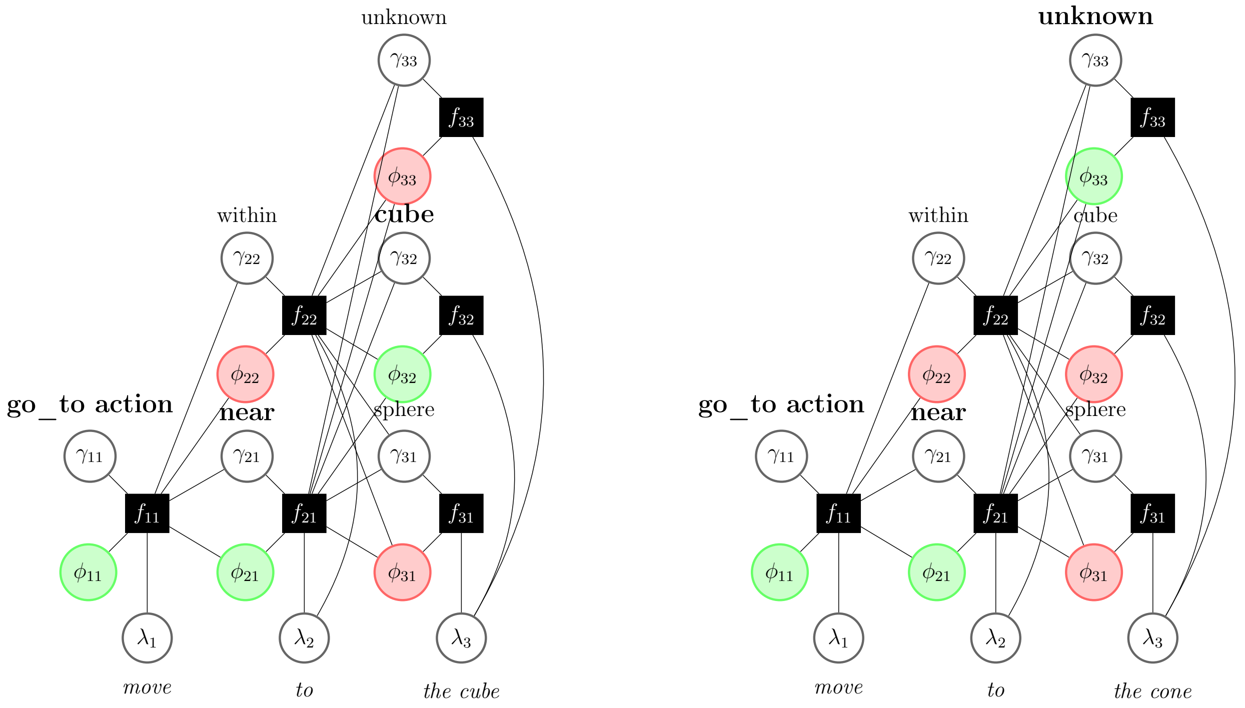


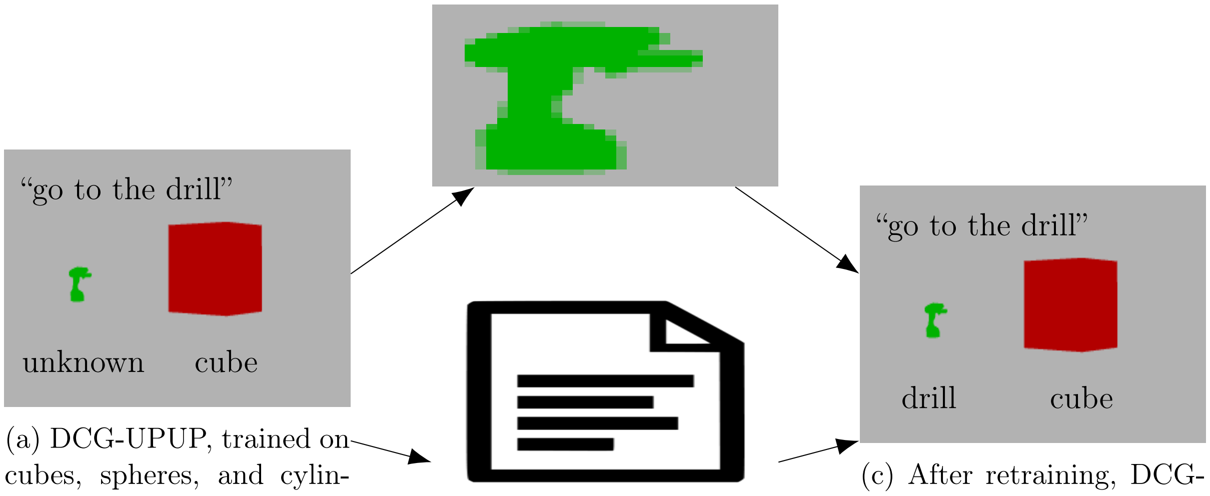
Figure 3-4: DCG-UPUP grounds a known phrase "cube" to a known object on the left but grounds an unknown phrase "cone" to an unknown object on the right. Green ϕs show true correspondences for the solved graph; red ϕs show the converse.

3.5 Unsupervised Learning and Concept Acquisition

In the previous section, DCG-UPUP infers that unknown phrases are associated with unknown objects, so when it is given the phrase “sofa” and sense a sofa for the first time, it claims that “sofa” must be referring to the sofa. In the absence of an active learning mechanism, however, the next time DCG-UPUP sees a sofa or hears the phrase “sofa,” it will once again treat the percept and phrase as unknown. This is undesirable. Instead, ideally the robot would remember what sofas look like and that the word “sofa” refers to sofas. In other words, the robot should learn about a symbolic sofa.

Perhaps the simplest way to teach DCG-UPUP about the new symbol is to generate new training examples, with the relevant new phrase and a new object type, and create a new LLM trained with the original and new training files. (It may be possible to efficiently update the LLM rather than rebuilding it from scratch, but that would only change the runtime of the program, not its overall behavior.) Generating the training examples, too, is relatively straightforward: the output of DCG-UPUP, supplemented with a description of the full environment, exactly matches the information contained within a standard training example. In other words, DCG-UPUP can autonomously generate new training examples that will permanently teach it new symbols.

There are, of course, several subtleties to take care of. First, the object recognizer must be retrained to correctly classify the new object, but that can be achieved by saving a snapshot of each object and relabeling and retraining whenever a new type is created. Second, DCG-UPUP must incrementally expand the set of known object types in include new types as they are encountered. Finally, the new object type, and the new phrase, must be paired with new features that detect them (e.g. a feature to detect when the new phrase is used or a new feature that detects if the object is perceived). The rest of the learning (e.g. updating the set of known words κ to contain the new phrase) occurs automatically when the LLM is retrained. Figure 3-5 shows the workflow from command execution, through generating unsupervised training examples, to execution of a new command after retraining.



(a) DCG-UPUP, trained on cubes, spheres, and cylinders, is told to "go to the drill" and recognizes the "drill" and the physical drill as unknown.

(b) A snapshot of the drill is saved for retraining the object classifier, and a new training example is written from the grounding output of DCG-UPUP.

(c) After retraining, DCG-UPUP has learned to recognize drills and that "drill" grounds to drills.

Figure 3-5: Unsupervised learning allows DCG-UPUP to first recognize drills as unknown and later as their own type.

In order to improve the robustness of both the language grounding and object recognition models, this unsupervised learning technique is used after every natural language command has been executed to completion. (The LLM and object recognizer are retrained only when manually commanded.) This delayed retraining is particularly useful for the object recognizer, which often struggles to correctly classify objects based on only a single sample image of an object type. Instead, if the robot saves several snapshots of an object as it drives towards it, the robot may generate a more complete and robust image training set. Sample snapshots in Figure 3-6 from a real-world trial in which a turtlebot was given the command "go to the cone" show how DCG-UPUP stores images of the cone as it approaches its destination.

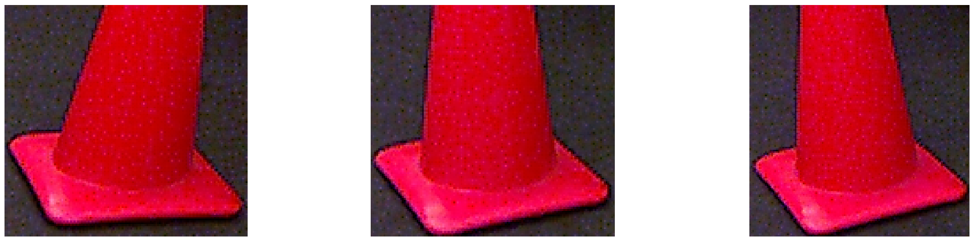


Figure 3-6: DCG-UPUP saves snapshots of the object it has grounded a command to, thus providing training examples for the object recognizer.

Finally, it is important to note there are dangers with this unsupervised learning approach. As with any fully unsupervised learning technique, concept drift risks worsening the performance of the system over time. For example, if DCG-UPUP has been trained to recognize cubes, spheres, and cylinders and hears the command "go to the fire hydrant," the unknown phrase "fire hydrant" will ground to the first unknown object that the robot sees. If that object is not a fire hydrant but a cone, after retraining, the object recognizer will subsequently perceive cones as fire hydrant types. This error will only worsen as the robot continues to ground phrases since "fire hydrant" will no longer be an unknown phrase, and cones will be perceived as fire hydrant types.

The obvious mitigation for concept drift is transforming the unsupervised learning into semi-supervised learning with humans in the loop. A human user, for example, could confirm or deny proposed new training examples. In fact, significant research in information-theoretic dialog has already been conducted and would fit well with this work [18]. Such research, however, must wait for future work, discussed in Chapter 7.

3.6 Synonyms

In its current proposed form, DCG-UPUP may be easily adapted to yield one other important result: by assuming that phrases only refer to objects the robot can currently perceive, DCG-UPUP can learn synonyms. Unfortunately, this assumption will be violated in Chapter 4 when phrases may ground to objects that are not perceived. Resolving the ambiguity of synonyms and unseen objects must be left for future work. For the remainder of this section, however, one may see how synonyms may be included within the DCG-UPUP system.

Synonyms easily fit into the grounding framework already presented. When grounding a synonym, the object type is already known, but the phrase is different. If the robot may assume that phrases only refer to objects it can currently perceive, one must only consider two possible cases: 1) the robot perceives one or more unknown object or 2) the robot does not perceive any unknown objects. In the first case, DCG-UPUP favors grounding unknown phrases to unknown objects, so it will try to choose the best candidate unknown object. In the second case, DCG-UPUP must ground an unknown phrase to a known object, so, similarly, must choose the best candidate known object. Heuristics used for breaking such ties (among either known or unknown objects) are presented in Chapter 5.

The behavior in the second case is exactly what is necessary for learning synonyms. An unknown noun (e.g. "ball") is associated with a known object (e.g. a sphere). By using the same unsupervised learning trick of retraining on grounding outputs, the new phrase may be permanently learned.

Unfortunately, as stated previously, the critical issue that merits further investigation is deciding whether an unknown phrase refers to an unknown object (perceived or unperceived) or a known object. Incorporating additional spatial clues from language (e.g. "pick up the ball in the front") or gestures (e.g. pointing to the sphere) could help solve this problem through disambiguation. Subsequently modifying DCG-UPUP to learn synonyms, following the procedure described above, would be straightforward.

Chapter 4

Grounding out of Perception

Regardless of how well the robot may ground to objects it perceives, performance of DCG-UPUP is limited unless the robot can ground to objects that it cannot perceive. For example, if a robot is told "go to the cube behind the wall," if the robot can see a wall but cannot see behind it, the robot should deduce approximately where the cube probably is. In an even simpler scenario, imagine that a robot that has been trained to ground cubes and spheres is facing only cubes and is told "go to the sphere." Once again, the robot should be able to deduce that in such a scenario, it must explore outside its current field of view rather than finding the most sphere-like cube. A new version of DCG-UPUP, called DCG-UPUP-Away (for Distributed Correspondence Graph - Unknown Phrase Unknown Percept - Away) is presented that explicitly allows a robot to ground to objects outside its current field of view.

The majority of this chapter will discuss various approaches initially developed to adapt DCG-UPUP for out-of-world groundings. Two model changes, although potentially promising for future research, are considered but ultimately discarded for a more elegant and robust approach that involves "hypothesizing" objects outside the field of view. Finally, this chapter concludes by explaining how natural language feedback from the robot to the human could be integrated for more robust performance.

4.1 Alternate Approaches to Grounding Out-of-World

4.1.1 Default Out-Of-World Groundings

One possible approach for deciding when to ground to out-of-world objects is based on a sort of default behavior: if the confidence of the best grounding falls below a certain threshold, the robot should explore the world it cannot perceive. This simple method has the advantage of not changing the DCG-UPUP model and therefore scarcely affecting runtime. Furthermore, it appears to reflect how people may decide to look for objects; if one is told to find a cube yet sees nothing that looks like a cube, say, one might walk around until finding a good match.

Although such behavior may seem to reflect how humans think, in fact, upon further consideration, this appearance disintegrates. Default behavior and human behavior diverge when phrases either match several objects very well, or no objects at all. Consider, for example, telling a robot to "pick up the cube" when it is facing 20 cubes. If every cube is equally likely to be the correctly grounded object, then the probability of the most likely grounding will be around 5%. If the threshold for an out-of-world grounding is greater than 5%, therefore, the robot will erroneously decide to look for a cube other than the 20 it currently sees. Clearly, a similarly difficult environment may be constructed for any threshold, leading to failure.

In addition, this approach is unattractive simply because it is not a learned behavior. Ideally, DCG-UPUP-Away would inherently decide where to ground to not through a set of pre-defined rules, but rather through seeing several training examples and generating its own rules. Having a default threshold violates this principle.

4.1.2 Symbolic Representation of Out-Of-World Groundings

Another possible approach is to have a symbol, *out_ of_ world* that represents all possible out-of-world groundings. Therefore, if DCG-UPUP-Away grounds to *out_ of_ world*, the robot decides to explore a new part of the environment.

This appears to be a satisfactory solution that matches the thresholding idea in the previous section, but with the theoretical advantage of being a learned behavior. In order to teach DCG-UPUP-Away when to ground out of the world, training examples are constructed in which nouns that do not match perceived groundings are grounded to the *out_of_world* symbol.

There remain, however, two problems with this approach. First, using a single symbol to represent all possible object types outside the current field of view seems like an inaccurate representation of how people decide whether or not a phrase refers to something they cannot see. Secondly, after actually implementing this approach, performance remains frustratingly poor. It appears as if the underlying problem is that, when the LLM is trained to ground many different phrases to the same *out_of_world* symbol, distinctions among the different phrases fade. Thus, phrases such as "cube" and "sphere" start grounding to the same groundings, even if a cube or sphere is perceived. Clearly, such behavior is unacceptable.

4.2 "Hypothesized" Objects Approach

The solution used for the final version of DCG-UPUP-Away addresses the problems in the previous two approaches. The main idea is to create a hypothesized object for each possible object type (all the known types as well as the unknown type, introduced in DCG-UPUP). Note that this means each perceived object has a hypothesized counterpart of the same type. If DCG-UPUP-Away grounds to a hypothesized object rather than an actual object that the robot perceives, the robot decides to explore new areas. This approach matches the two design criteria proposed: first, it is a learned behavior and, second, it matches human intuition by comparing how well a phrase grounds to hypothesized objects and perceived objects.

In order to implement these changes, DCG-UPUP is altered before any inference is actually conducted. First, DCG-UPUP-Away populates a world model Υ based on what objects are currently perceived. This is exactly what happens in DCG and DCG-UPUP. Second, DCG-UPUP-Away iterates through each possible object type (maintained as a growing set, as explained in Chapter 3) and adds a single hypothesized instance of each object to the world model. For the sake of computational speed and simplicity, only a single hypothesized instance of each object type is added, but doing so assumes that natural language commands will not refer to several out-of-world objects. This assumption can be relaxed by creating several hypothesized objects of the same type, but is outside the scope of this research.

After adding the hypothesized objects to the world model, inference proceeds as in DCG-UPUP, but with the new world model. If the grounded object is a hypothesized object, the robot must explore its surroundings; otherwise it can see the grounded object. Currently, "exploring" surroundings consists of slowly rotating in place, thus revealing more of the surrounding environment. As DCG-UPUP-Away perceives additional objects, it attempts to ground the command within the updated world model. Future work could easily yield more sophisticated behavior by adding exploratory heuristics or even using language to guide search (as in [19] and [3]).

Intuitively, this approach finally matches how people decide whether or not a phrase refers to something they cannot currently see. Upon hearing a phrase, it is natural to compare how well that phrase matches each object currently seen (grounding within the world) or each object not seen (grounding outside the world). For example, when facing a cube and hearing the phrase "go to the cube," the phrase "cube" matches the cube better than, say, an imaginary sphere outside the field of view. Conversely, when facing a cube and hearing the phrase "go to the sphere," the phrase "sphere" does not match the cube but would match a sphere very well if it existed outside the field of view, so the natural decision is to look elsewhere for the sphere.

There is, however, one subtlety worth considering: a phrase may refer to an object that is currently outside the field of view, but within the field of view there may be an object of that same type. Imagine, for example, a robot located between two cubes, only looking at one. When told “go to the cube,” what should the robot do? On the one hand, the cube in front of it is a high probability grounding for the phrase “cube.” At the same time, a hypothetical cube would match “cube” with equally high probability!

One way to resolve this subtlety is to introduce an additional feature and training examples into DCG-UPUP-Away. The feature – f_{hypobj} – fires when the grounded object is hypothesized, but does not fire if the object is perceived. Only 6 positive training examples demonstrating tie-breaking between real and hypothesized objects (e.g. “go to the cube” grounds to a perceived cube rather than a hypothesized cube) are necessary to train DCG-UPUP-Away to favor real objects.

4.2.1 Modifications to the DCG Model

Just as when comparing DCG-UPUP to DCG, one may compare DCG-UPUP-Away to DCG by examining the domains of variables used in solving the grounding problem. Once again, the underlying equation used for finding the optimal grounding remains the same.

Table 4.1: Identical equations are used for DCG, DCG-UPUP, and DCG-UPUP-Away

DCG	$\boldsymbol{\phi}^* = \underset{\phi_{ij} \epsilon \Phi^{\|\boldsymbol{\lambda}\|}}{\arg\max} \prod_{i=1}^{n} p(\phi_{ij} \| \gamma_{ij}, \lambda_{ij}, \Gamma_{c_{ij}}, \Upsilon)$
DCG-UPUP	$\boldsymbol{\phi}^* = \underset{\phi_{ij} \epsilon \Phi^{\|\boldsymbol{\lambda}\|}}{\arg\max} \prod_{i=1}^{n} p(\phi_{ij} \| \gamma_{ij}, \lambda_{ij}, \Gamma_{c_{ij}}, \Upsilon)$
DCG-UPUP-Away	$\boldsymbol{\phi}^* = \underset{\phi_{ij} \epsilon \Phi^{\|\boldsymbol{\lambda}\|}}{\arg\max} \prod_{i=1}^{n} p(\phi_{ij} \| \gamma_{ij}, \lambda_{ij}, \Gamma_{c_{ij}}, \Upsilon)$

This time, the domains of Γ and Υ have grown larger to reflect the set of possible hypothesized objects. The resulting graphical model for DCG-UPUP-Away is shown in plate notation in Figure 4-1, with the domains of the variables in Table 4.2. In this complete model, nouns may ground to known and perceived objects, unknown and perceived objects, known and hypothetical objects, or unknown and hypothetical objects.

Table 4.2: Variable and domain definitions for DCG-UPUP-Away. Unknown phrases (Λ_U) and unknown groundings (Γ_{UO}) remain, and hypothetical objects are added to the world model (Υ_H).

$\Gamma = \Gamma_{KA} \cup \Gamma_{KO} \cup \Gamma_{UO}$
$\Lambda = \Lambda_K \cup \Lambda_U$
$\Upsilon = \Upsilon_P \cup \Upsilon_H$
$\Phi = \{True, False\}$

Finally, one may revisit the LLM used for associating phrases with objects. Whereas DCG-UPUP introduces two additional features (one for detecting unknown phrases and one for detecting unknown objects), the only feature added for DCG-UPUP-Away is f_{hypobj}. Thus, the computation of Ψ may be rewritten as in Equation 4.2. The LLM now clearly uses the original features from DCG, the two features for associating unknown phrases with unknown objects, and the feature for detecting hypothesized objects.

$$\boldsymbol{\phi}^* = \underset{\phi \epsilon \Phi^{|\boldsymbol{\lambda}|}}{\arg\max} \frac{1}{Z} \prod_i \Psi(\phi_{ij}, \gamma_{ij}, \Gamma_{c_{ij}}, \Upsilon) \tag{4.1}$$

$$\begin{aligned}\Psi = \exp\Big(\sum_{f \epsilon F_{\text{DCG}}} & \big[\mu_f f(\phi_{ij}, \gamma_{ij}, \lambda_i, \Gamma_{c_{ij}}, \Upsilon)\big] + \\ & \sum_{f' \epsilon F_{\text{Unknown}}} \big[\mu_{f'} f'(\phi_{ij}, \gamma_{ij}, \lambda_i, \Gamma_{c_{ij}}, \Upsilon)\big] + \\ & \qquad \sum_{f'' \epsilon F_{\text{Hypothesized}}} \big[\mu_{f''} f''(\phi_{ij}, \gamma_{ij}, \lambda_i, \Gamma_{c_{ij}}, \Upsilon)\big]\Big)\end{aligned} \tag{4.2}$$

4.3 Potential Improvement through Natural Language Feedback

As mentioned at the end of Chapter 3, DCG-UPUP originally allowed for learning synonyms, but only under the assumption that phrases refer exclusively to objects that are currently perceived. That assumption is relaxed in DCG-UPUP-Away, and a new problem arises: how can one distinguish between a synonym for a known object that is present in the world and a new phrase referring to a new object that is out of the field of view of the robot. In fact, without outside information, it is impossible to distinguish between the two cases.

One possible way to resolve this problem is through natural language feedback from human users. For example, instead of finding the single most likely grounding, DCG-UPUP-Away could propose two groundings: the most likely grounding to a perceived object, and the most likely grounding to a hypothetical object. The human could then specify which of the two groundings to choose.

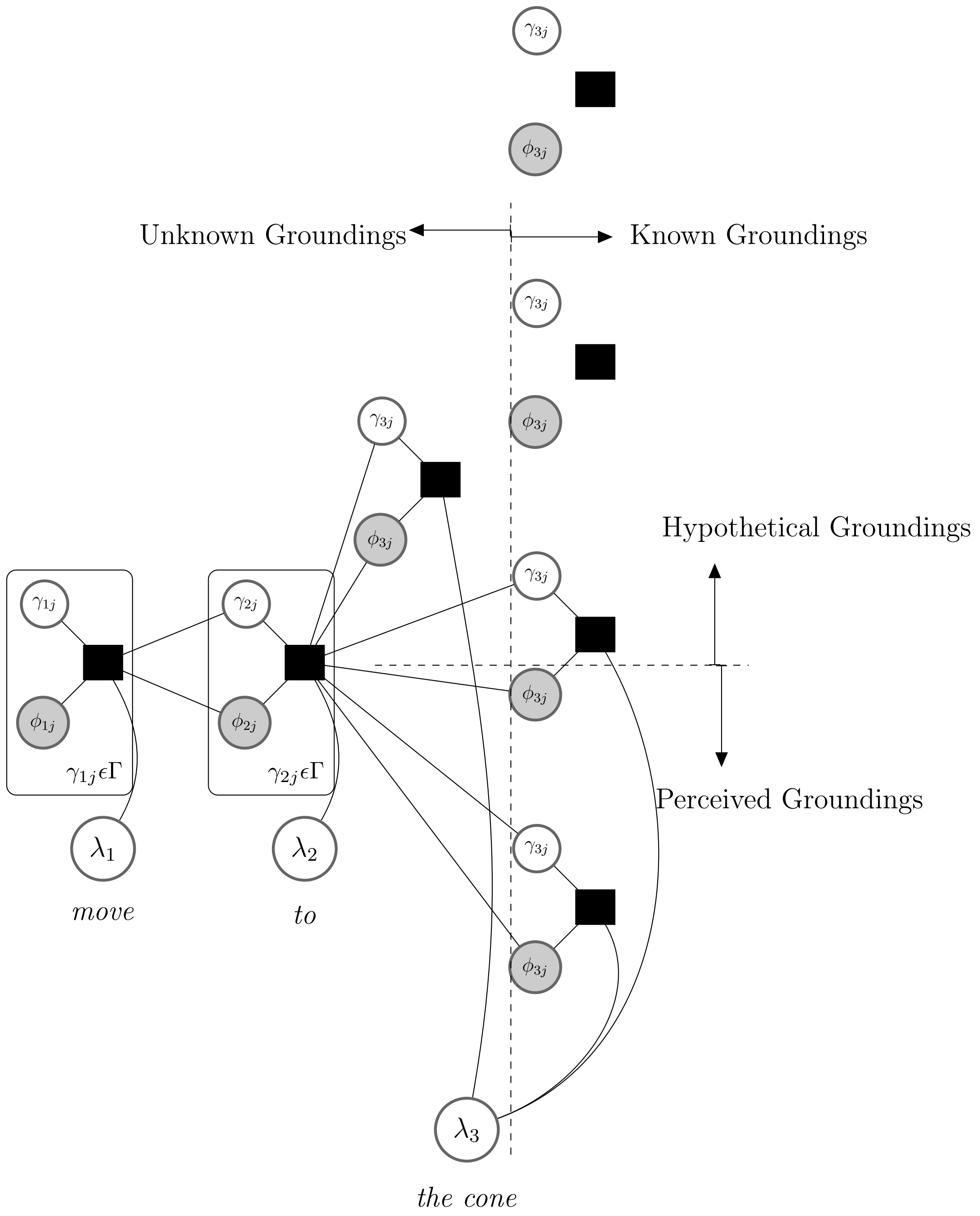


Figure 4-1: The graphical model for DCG-UPUP-Away. The four types of objects — known perceived, unknown perceived, known hypothetical, and unknown hypothetical — are labeled as Γ_{KP}, Γ_{UP}, Γ_{KH}, and Γ_{UH}, respectively.

Chapter 5

Resolving Object Ambiguity through Heuristics

In this chapter, grounding accuracy is improved by reasoning with object properties and natural language adjectives. DCG-UPUP-Away, as developed so far, solves the problems initially described in the introduction: a robot may now reason about unknown objects and phrases, autonomously learn new symbols, and even reason about objects that it cannot perceive. However, DCG-UPUP-Away occasionally grounds to the wrong object when multiple objects of the same type are present. For example, if two cubes are perceived simultaneously when given the command “go to the cube,” the model assigns equal likelihoods to grounding to either of the cubes and may therefore mistakenly choose the wrong cube to approach. Perhaps more importantly, if DCG-UPUP-Away perceives two unknown objects (e.g. a mailbox and a cone) and is told to “go to the cone” when the phrase “cone” is unknown, it is just as likely to ground to the mailbox as the cone. Due to the unsupervised learning techniques used for symbol acquisition, this has potentially long-term effects. Therefore, in order to increase robustness for experimental validation and to indicate a broad area for future work, DCG-UPUP-Away has been modified to use context referents in language and object property information from the real world to determine the correct grounding. Specifically, DCG-UPUP-Away uses color adjectives to disambiguate between objects with different colors.

5.1 Form of Color Heuristic

The role of the color heuristic is to differentiate between two objects of the same type: for example, a red cube and a blue cube. More generally, any heuristic relating an object's properties to language (e.g. "small" and and object's size) assists in disambiguating among objects. For a deployable robot in the real world, many such heuristics may be necessary, as natural language routinely references object properties, not just their type.

Nearly all heuristics considered for DCG-UPUP-Away match the form of the color heuristic: an association is learned between natural language (e.g. "red") and an object property (e.g. its RGB value). Any heuristic that ignores natural language but favors certain groundings over others makes that assumption that those objects are generally more likely to be grounded to. An example of this is the hypothesized heuristic, introduced in Chapter 4. In general, when deciding between grounding to two identical objects, one of which is perceived and the other of which is hypothesized, it is more likely that the language grounds to the perceived object. Therefore, the hypothesized heuristic only requires one feature that determines whether or not an object is hypothesized. More generally, though, heuristic features that rely on both language and object properties require at least two features: one to detect the language part, and one the object property part. By including these two features it is possible to learn their association.

5.2 Color Heuristic Features

The remainder of this chapter will explain how the color heuristic is included within DCG-UPUP-Away. Only 3 main changes are necessary: 1) a f_{word} feature must be added for each adjective that contains color information (e.g. "red"), 2) a new feature, f_{color}, must be developed to fire if an object's color matches the feature's argument (e.g. red), and 3) a f_{color} feature must be added for each color that an object can take on (in this case restricted to a set of common colors) passed in as an argument, *color_arg*. The mathematical formulation of f_{color} is shown in Equation 5.1:

$$f_{\text{uphrase}}(\phi_{ij}, \gamma_{ij}, \lambda_i, \Gamma_{c_{ij}}, \Upsilon) = \begin{cases} 1, & \text{if } \gamma_{ij}.color == color\_arg \\ 0, & \text{else} \end{cases}. \tag{5.1}$$

5.2.1 Color Language Feature

On the language detection side, minimal changes are required. The feature for detecting when a particular word is used in a phrase, f_{word}, has already been developed in DCG. Therefore, all that is required is to add features to detect what color phrases appear. In this particular development, that meant adding f_{word} with the arguments "red", "green", and "blue".

There is one further change necessary for a sentence containing color adjectives to be parseable in the first place: the parser must be updated to allow for adjectives. Recall that parse trees are necessary in order to infer the structure of the graphical models in G^3, DCG, and DCG-UPUP-Away. It is possible to add an adjective or even multiple adjectives in front of nouns, however, by introducing a recursive parse tree element with a base case of an adjective followed by a noun. This small modification is all that is necessary to parse phrases with potentially infinite adjectives.

5.2.2 Color Object Feature

Just as language features are added to detect phrases like "red," object property features must be added to detect what colors objects are. Adding such features requires two changes: 1) objects must contain a color property and 2) a feature must be created that accepts a color as an argument and fires if a grounded object matches the color.

Updating objects to keep track of color is relatively simple. There are myriad ways to represent color (e.g. RGB or HSV), but for simplicity, in this instance, development was limited to representing color as an object property that could take on four possible values: "red," "green," "blue," and "na." Updating the world model Υ to reflect objects' colors is similarly straightforward. The object recognizer introduced in Chapter 3 is modified to also detect the most prevalent RGB value within the bounding box of each object by thresholding the color histogram of the overall image. If no color channel value is noticeably greater than the others, the object color is set to "na".

Given that the world model maintains information about each object's color, one may develop a feature that fires when the grounded object matches a specific color. The feature, called f_{color}, is almost exactly like an existing feature, f_{objtype}, which checks what type of object the grounded object is. Instead of checking the object-type property, however, f_{color} checks the color property. Therefore, three f_{color}'s were added to the feature set with arguments "red," "green," and "blue." By including several training examples in which phrases like "go to the red cube" grounds to a red cube instead of a blue cube, DCG-UPUP-Away can learn the association between color adjectives and color properties.

Although f_{color} as described so far has been developed to accomplish most of the desired goals of disambiguating between objects of the same type, one key problem remains: given that DCG-UPUP-Away hypothesizes objects in order to ground to objects it cannot perceive, and given that color now influences groundings, it appears as if it is necessary to hypothesize every object in every color. Such a procedure could be computationally prohibitively expensive, since it involves searching over the cross of the object type space and the color space. Instead, only a minor modification is made to f_{color}. Previously, f_{color} returned true if and only if the grounded object's color matches the feature's argument (e.g. f_{color}("blue") firing only if the grounded object is blue). Now, f_{color} returns true if the grounded object's color matches the feature's argument or if the object's color is "na." As a result, if hypothetical objects are created with colors set to "na," every feature color will fire – as if the object has the correct color to match the language, whatever that color may be. In practice, this approach works quite well and efficiently solves the problem of simultaneously solving for color and object type.

Having presented the main ideas behind incorporating the color heuristic, one may formally see how the inference procedure in DCG-UPUP-Away is changed. The probabilistic graphical model itself remains unchanged because the domains of groundings stays the same (objects merely contain color information as well). 7 additional features have been introduced, though: 3 for detecting the words "red," "blue," and "green", 3 for detecting object colors red, blue, and green, and 1 for detecting the object color "na." This appears in the LLM equations as shown in Equation 5.3.

$$\boldsymbol{\phi}^* = \underset{\boldsymbol{\phi} \epsilon \Phi^{|\boldsymbol{\lambda}|}}{\arg\max} \frac{1}{Z} \prod_i \Psi(\phi_{ij}, \gamma_{ij}, \Gamma_{c_{ij}}, \Upsilon) \tag{5.2}$$

$$\begin{aligned}\Psi = \exp\Big(\sum_{f \epsilon F_{\text{DCG}}} & \big[\mu_f f(\phi_{ij}, \gamma_{ij}, \lambda_i, \Gamma_{c_{ij}}, \Upsilon)\big] + \\ & \sum_{f' \epsilon F_{\text{Unknown}}} \big[\mu_{f'} f'(\phi_{ij}, \gamma_{ij}, \lambda_i, \Gamma_{c_{ij}}, \Upsilon)\big] + \\ & \sum_{f'' \epsilon F_{\text{Hypothesized}}} \big[\mu_{f''} f''(\phi_{ij}, \gamma_{ij}, \lambda_i, \Gamma_{c_{ij}}, \Upsilon)\big] + \\ & \qquad \sum_{f''' \epsilon F_{\text{Colors}}} \big[\mu_{f'''} f'''(\phi_{ij}, \gamma_{ij}, \lambda_i, \Gamma_{c_{ij}}, \Upsilon)\big]\Big)\end{aligned} \tag{5.3}$$

Another possible way to solve this problem would be to first solve for color (independent of object type) and then solve for object type. This would simultaneously solve the computational problem of searching jointly over color and object type while matching human intuition. In many ways, solving for color and solving for object type seem like independent problems, which bolsters the argument for this alternate approach. Such an approach would involve introducing an additional type of symbol representing color and is therefore left as a promising area for future research.

Chapter 6

Evaluation

In this chapter, a framework for evaluating DCG-UPUP-Away and the results found from the proposed experiments are presented. This section will demonstrate that the DCG-UPUP-Away model behaves as expected randomly generated simulation trials as well as a proof-of-concept trial run on hardware.

The main metric of success is whether or not DCG-UPUP-Away grounds to the correct object in a scene. One additional datum worth considering, however, is how many symbols DCG-UPUP-Away correctly knows at any given time (e.g. it was initially trained to ground cubes, spheres, and cylinders, but now it also knows how to ground cones). Both metrics will be considered, although the primary focus is grounding accuracy.

6.1 Experimental Design

Although the examples in the previous section demonstrate some of the new capabilities developed in DCG-UPUP-Away, two questions remain unanswered: 1) do these capabilities apply to real-world scenarios and 2) how does DCG-UPUP-Away perform in such scenarios? To answer these questions, an experiment was designed specifically to test DCG-UPUP-Away using language generated by the human subjects in randomly generated environments.

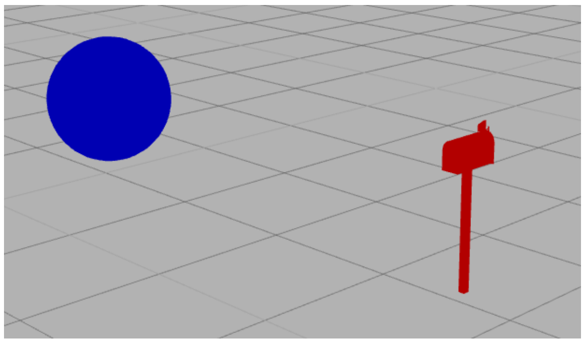
The overarching idea of the evaluation procedure falls into 4 phases:

1. Randomized environment generation
2. Natural language command generation
3. DCG-UPUP-Away training
4. DCG-UPUP-Away evaluation

6.1.1 Simulated Environment Generation

In the first phase, 10 random environments are generated by selecting a subset of objects from a library of 24 possible objects. These 24 objects are generated by combining one of 8 object types (cube, sphere, cylinder, cone, hydrant, handle, mailbox, or drill) with one of 3 possible colors (red, green, or blue). Each object is selected to be added to the environment with a 15% probability. Each added object is placed at a random location generated from a uniform distribution over a 4 by 4 meter square around the robot (excluding a 2 by 2 meter square directly around the robot). Note that the global orientations of objects are not randomized - given that the robot will be placed in the middle of the scene and that objects are randomly located, though, the orientation of an object relative to the robot will therefore automatically be randomized. Finally, environments are rejected if they contained zero or one objects or if an object is completely hidden behind another object when viewed from the starting position of the robot. Figure 6-1 shows samples from the dataset of simulated environments.

Generating environments in such a manner maintains two important facts about the environments. First, the robot cannot know ahead of time what objects are present in the world. Second, objects may be placed out of the initial field of view of the robot. These two facts, coupled with the fact the robot will only be trained


(a) A generated random environment containing a blue sphere and a red mailbox.

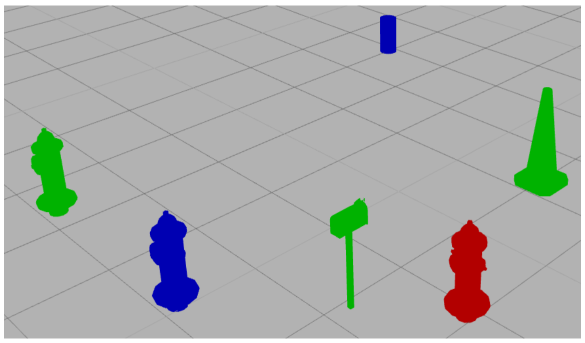

(b) A different generated random environment containing a green hydrant, blue hydrant, green mailbox, red hydrant, green cone, and blue cylinder.

Figure 6-1: Sample generated test environments

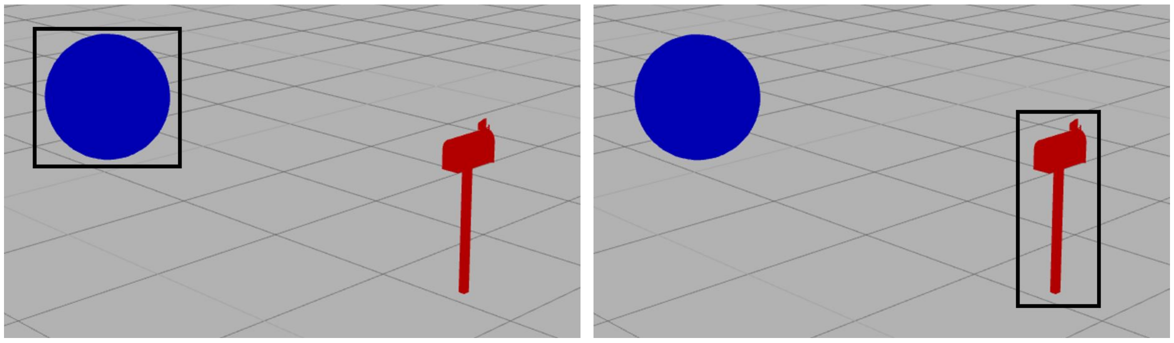
with cubes, spheres, and cylinders, means that both the unknown-phrase, unknown-percept and out-of-world grounding capabilities will be tested.

6.1.2 Natural Language Command Generation

After generating the environments, each object within each environment is paired to a natural language command. Such commands may then be used to tell the robot how to approach any given object in any environment, and the response of the robot can later be assessed for accuracy.

Given the 10 particular generated environments, there are 39 objects total. Screenshots of each environment are taken and then modified to highlight one particular object. Figure 6-2 shows two sample images of the same environment with different objects highlighted.

These images with highlighted objects are then uploaded to a form on Amazon Mechanical Turk for natural language annotation. Online users are shown one such image and asked to "Please write the command for approaching the highlighted object." Users are also provided with a sample image of a green cylinder, red cube, and



(a) Highlighting a blue sphere in a sample environment. (b) Highlighting a red mailbox in a sample environment.

Figure 6-2: Example environments with highlighted objects.

blue sphere in which the cylinder is highlighted. The sample image is accompanied by the command "go to the green cylinder."

In total, 390 annotations are collected: 10 for each of the 39 images. Allowing 10 annotations for each image mitigates the problem of improperly formed annotations. For example, one user routinely used spatial relations in annotations (e.g. saying "go to the object on the left") despite the instructions explicitly prohibiting such phrases. In addition, because the true goal of collecting the natural language commands is to generate a series of consistently labeled images, ideally one user should label all 39 images. In fact, several users stopped labeling after only a handful of images, rendering their annotations useless. Ultimately, however, several users followed nearly all the instructions (although occasionally alternated between labeling spheres as "sphere" and "circle") and labeled each of the 39 images. The commands of the first such user were used for the rest of the experiment. Thus, although only a small fraction of the annotations were used, Amazon Mechanical Turk served its purpose by generating commands from non domain experts.

6.1.3 DCG-UPUP-Away Initial Training

Often, experiments designed to assess natural language understanding use a randomly selected subset of their test data as a training set [54, 29]; in this experiment, however, a pre-defined corpus of 55 example phrase groundings are used. If this experiment used a subset of the test data, the likelihood of assessing performance when grounding to unknowns would be extremely small, because few objects or phrases would remain unknown.

The only way to guarantee that some phrases and percepts will remain unknown is by leaving them out of the training set. Therefore, an initial grounding data set is constructed that only grounds cubes, spheres, and cylinders (other nouns, such as "mallet" and "pipe" are also used, but only to train grounding to unknowns). The majority of the groundings are used for training how to ground "cube," "sphere," and "cylinder" as well as how to use color to disambiguate between objects. In addition, 8 of the examples are used to introduce synonyms to the known types (e.g. "circle" grounds to a sphere type and "box" grounds to a cube type). These synonym training examples introduce a degree of vocabulary robustness that was ultimately used to ground real users' phrases. Finally, similar to the initial set of grounding training data, a set of images of cubes, spheres, and cylinders is set aside as the initial training for the object recognizer.

After training DCG-UPUP-Away with the initial training data, the system has learned the symbols for cubes, spheres, and cylinders. Subsequent trials could use any of 8 possible objects, meaning the system at first knows only $\frac{3}{8}$ of symbols in the testing environments.

6.1.4 DCG-UPUP-Away Evaluation

The simulated environments, natural language commands, and initial training data are all that is necessary for evaluating DCG-UPUP-Away. Two evaluation mechanisms are developed: one full end-to-end system with an automatic object recognizer, and one system in which object recognition is decoupled from the main system and performed manually (researchers hand-label each object as its object type or unknown when it is perceived). Although the former experiment evaluates DCG-UPUP-Away in a more realistic setting, the inaccuracy of the object recognizer introduces errors that mask whether or not DCG-UPUP-Away, the model, functions properly. Moreover, the object recognition module is fully separable from the language grounding module, so future object recognition work may be easily incorporated into the overall system.

Both methods (named "automatic" and "manual" for the automatic object recognition and manual object recognition systems, respectively) are used to determine their grounding accuracy rates over a series of natural language commands. The rate indicates how likely is it that DCG-UPUP-Away will ground to the right object for many commands even as it performs unsupervised learning between commands.

The exact evaluation procedures for both methods are nearly identical. First, for both methods, DCG-UPUP-Away was trained on the initial training set. The automatic method additionally trained an object recognizer based on a set of initial training images. Second, a series of environment and natural-language command pairs was uniformly randomly selected without replacement from the 39 possible commands. Each series is called one "trial"; thus a trial consists of several evaluations within different environments. Within a trial, a single evaluation is referred to as an "iteration." For the automatic method, which is considerably slower than the manual method, each trial contained 5 environments, so 5 iterations; for the manual method, each series contained 10 iterations. Third, a turtlebot was placed in each of the simulated environments, in order, and given the corresponding natural language command. Figure 6-3 shows a screenshot of the turtlebot placed in one such environment.

Between iterations in a given trial, DCG-UPUP-Away is retrained using the last generated unsupervised training example. For the automatic method, success for a given iteration is determined by whether or not the last camera view of the robot contains exactly one object and it is the correct robot. For the manual method, suc-

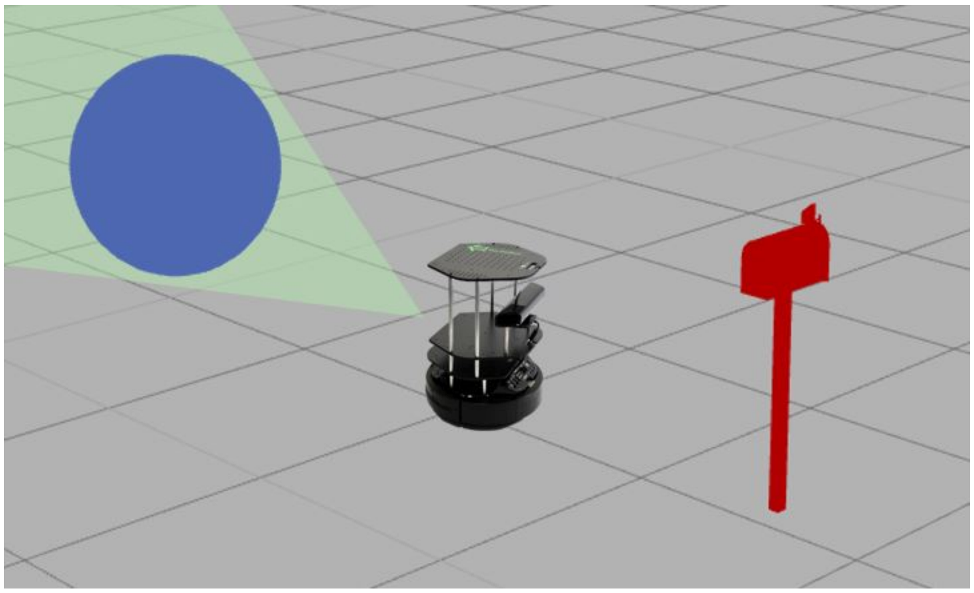


Figure 6-3: The turtlebot located in a simulated environment used for evaluation of DCG-UPUP-Away. Note how, given its orientation and position, the turtlebot perceives the blue sphere but has no way of knowing of the red mailbox.

cess is determined by manually reading through the grounding output file. Alongside recording an iteration's success rate, how many symbols DCG-UPUP-Away has correctly learned is also logged. After finishing a full trial of iterations, all unsupervised training examples are erased, and the entire evaluation process starts over with initial training.

6.2 Experiment Results and Analysis

The results have been divided into 4 groups. In the first, the grounding accuracy rate using manual object recognition is calculated as a function of iteration number within a given series. In the second, the number of correctly known symbols (ranging from 3 to 8), is plotted as a function of iteration number as well. In the third, the symbol and accuracy data are combined to show how accuracy rates increase as a function of symbols learned. Finally, in the fourth, preliminary data is presented for accuracy rate using automatic object recognition as a function of iteration number.

6.2.1 Grounding Accuracy across a Series

The primary result shown in this experiment is that the accuracy of DCG-UPUP-Away remains relatively constant at 80% over series of 10 iterations. The exact data showing this result are shown in Figure 6-4. The data used for this plot were generated using manual object recognition, for 30 trials, each with 10 iterations.

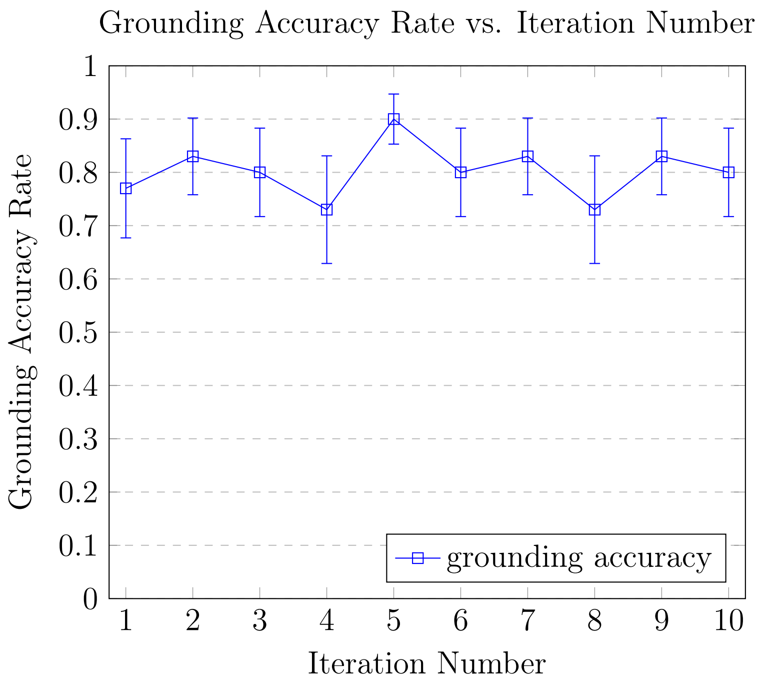


Figure 6-4: Relatively constant ground accuracy rate of DCG-UPUP-Away as a function of iteration number.

There are two main results shown in this plot. First, the baseline accuracy ranges between 70% and 90%. Averaged across all iterations and all trials (for a total of 300 individual evaluations), 80% of commands result in a successful grounding. Hidden within that 80% are a majority of iterations in which the object being referred to is unknown (not a cube, sphere, or cylinder), out of the field of view of the robot, or both. This feeds nicely into the second main point: grounding accuracy remains high across all iteration numbers. In other words, despite only using completely unsupervised learning within a trial, DCG-UPUP-Away is as likely to succeed after 9 iterations as after only training on hand-curated initial training data. Furthermore, during the course of retraining on unsupervised data, DCG-UPUP-Away learns new symbols.

6.2.2 Symbol Learning across a Series

The next metric worth examining is exactly how many symbols DCG-UPUP-Away learns over the course of 10 iterations. After all, if DCG-UPUP-Away is initially trained on cubes, spheres, and cylinders but only knows those three symbols at the end of a trial, maintaining the high grounding accuracy across iterations is not impressive. In fact, that is not the case. Figure 6-5 shows how, on average, the number of correctly learned symbols grows from 3 (cube, sphere, cylinder) in the first iteration to over 6 by the last iteration (the iteration number goes to 11 to allow for retraining after the 10th iteration).

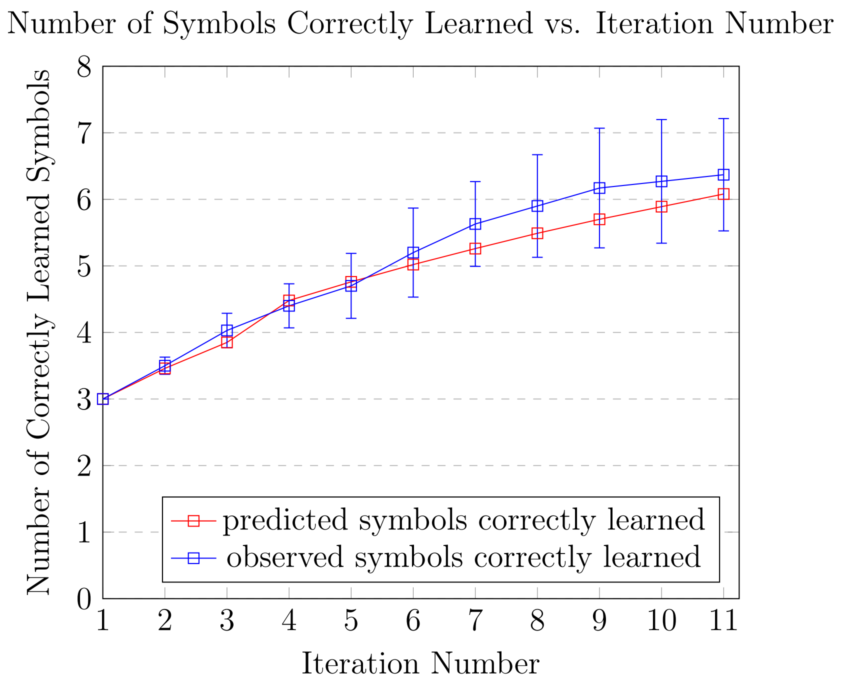


Figure 6-5: Increasing number of correctly learned symbols as a function of iteration number. DCG-UPUP-Away is initially trained to know 3 symbols of the 8 symbols used in generating environments.

At the very least, the data in Figure 6-5 validate that DCG-UPUP-Away performs as expected. The blue line, showing actual data, tracks the red line, showing algebraically computed expected values, quite well. (The algebraically computed expected values are generated using simple combinatorics reflecting a maximum of 8 possible known symbols.) The data are admittedly heteroscedastic; variance in the number of correctly learned symbols increases as a function of iteration number. In fact, this trend of growing variance may be unavoidable. After all, each series starts with exactly 3 known symbols and then randomly acquires new symbols, which introduces additional variance.

The most important result from Figure 6-5, however, is that DCG-UPUP-Away reliably learns a few symbols correctly. Despite only being initialized with knowledge of cubes, spheres, and cylinders, DCG-UPUP-Away may be placed in scenes containing up to 5 other unknown symbols: fire hydrants, cones, door handles, mailboxes, and power drills. On average, after 10 iterations, DCG-UPUP-Away has learned at least 3 of these symbols, completely autonomously. In fact, in 10% of the trials, DCG-UPUP-Away learned all 5 unknown symbols correctly by the last iteration.

Furthermore, if human feedback is ever incorporated into DCG-UPUP-Away, the average number of correctly learned symbols will likely increase dramatically. The most common failures, both in grounding accuracy and symbol learning, occurred when DCG-UPUP-Away had to pick between two unknown objects of the same color, grounded to the wrong one, and subsequently misclassified that object for all remaining trials.

6.2.3 Improved Accuracy through Symbol Acquisition

Having determined that DCG-UPUP-Away learns symbols, the natural next question is whether doing so improves grounding accuracy. In other words, does knowing more symbols make DCG-UPUP-Away more likely to correctly ground a natural language command. Therefore, rather than plotting accuracy as a function of iteration number within a trial, Figure 6-6 shows that knowing more symbols is positively correlated with improved accuracy ($r = 0.654$).

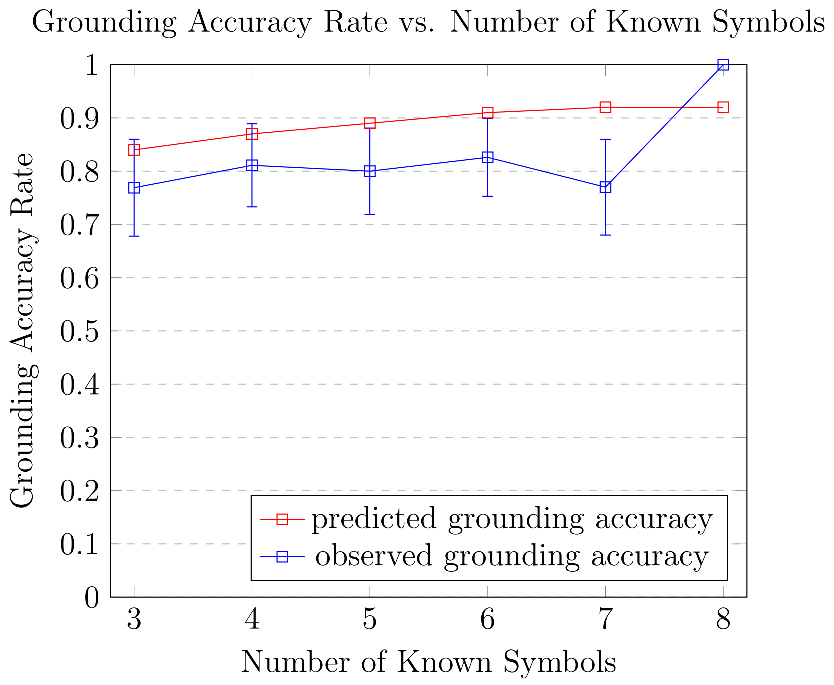


Figure 6-6: DCG-UPUP-Away grounding accuracy as a function of number of known symbols

It should be emphasized that Figure 6-6 plots grounding accuracy as a function of total number of symbols learned, even if such symbols are learned incorrectly. For example, if DCG-UPUP-Away grounds the phrase “hydrant” to a cone because both hydrants and cones are unknown (and the robot sees the hydrant before it sees the cone), DCG-UPUP-Away will think that it has learned a symbol for hydrant. If one plots grounding accuracy as a function of correctly learned symbols, however, one generates Figure 6-7.

In this figure, the correlation between learning symbols correctly and grounding more accurately is even tighter ($r = 0.857$). This substantial improvement in accuracy motivates still further future research in human feedback - if humans could tell a robot that the robot learned a symbol incorrectly, the gap in performance between Figure 6-6 and Figure 6-7 would shrink.

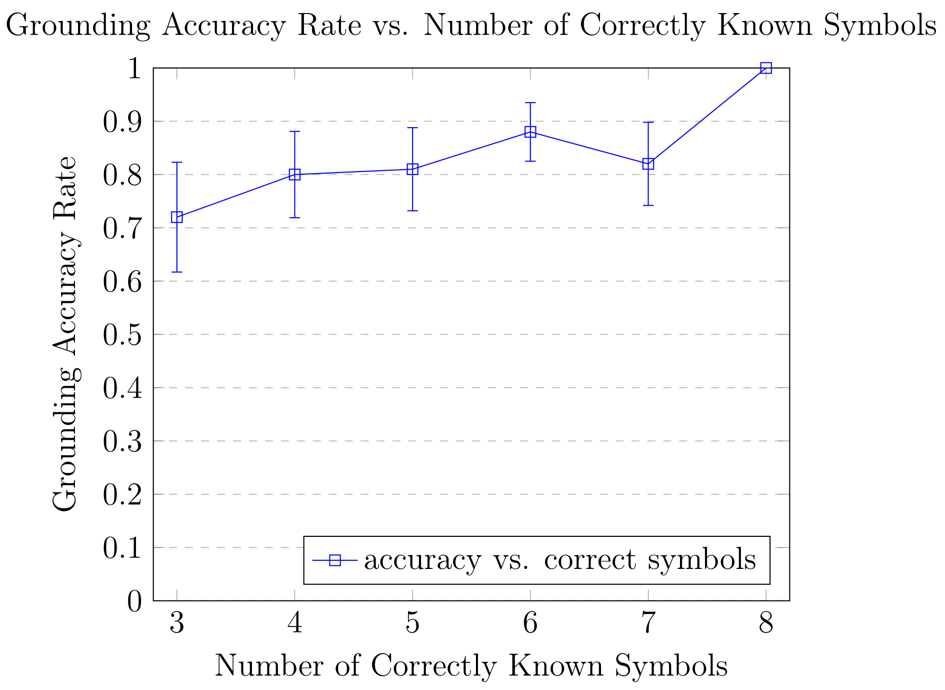


Figure 6-7: Grounding accuracy as a function of number of correctly known symbols

Even without human feedback, Figure 6-7 gives an additional insight into how DCG-UPUP-Away may be improved without modification of the actual model. Recall that the main source of grounding error is the result of ambiguous environments in which there are several unknown objects of the same color. If DCG-UPUP-Away makes a grounding error in such an environment and subsequently learns a symbol incorrectly, the number of correctly known symbols will remain persistently low. Therefore, DCG-UPUP-Away cannot benefit from the increased grounding accuracy from the right hand side of Figure 6-7. Instead, if one is careful in initially exposing DCG-UPUP-Away to unambiguous environments at first, DCG-UPUP-Away will rapidly learn symbols correctly. As a result, grounding accuracy will likely similarly improve since grounding accuracy is greater when more symbols are correctly learned. In fact, this strategy of unsupervised learning using simple examples first and then increasingly complex examples has appeared in both human cognition and symbolic artificial intelligence research [53].

6.2.4 Accuracy Using Automatic Object Recognition

One remaining question that must be addressed is how well the previous results apply in a fully autonomous system. All the data already presented were generated using manual object recognition; humans viewed the camera output of a simulated turtlebot and listed the color and type of each visible object. In fact, incorporating an object recognizer, slightly, but nevertheless noticeably, worsens grounding accuracy. Figure 6-8 shows grounding accuracy, over 10 trials, each with 5 iterations, for runs in simulation using an object recognizer.

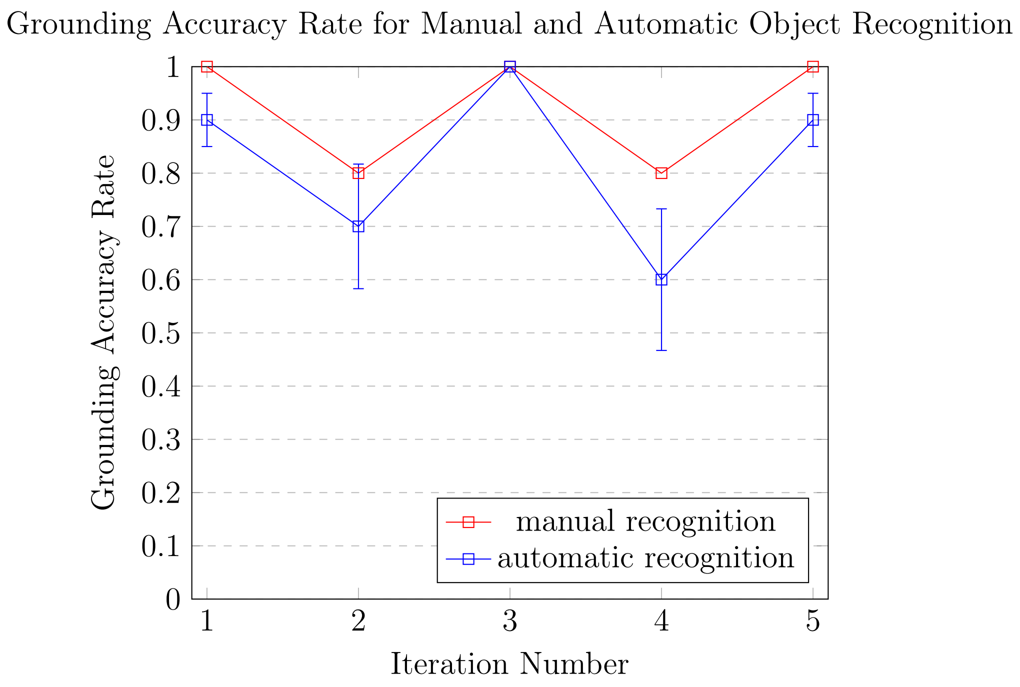


Figure 6-8: Grounding accuracy as a function of iteration number using automatic object recognition

The blue line, representing grounding accuracy for the automatic object recognizer, routinely falls below the red line, which was generated using manual object recognition for the same scenes. This graph clearly demonstrates that imperfect object recognition worsens DCG-UPUP-Away's performance. Fortunately, as previously stated, the object recognition module of DCG-UPUP-Away is completely separable, so a better object recognizer could easily be integrated into the overall system.

6.2.5 Hardware Demonstration

One final way to demonstrate the improvements offered by DCG-UPUP-Away is on a physical platform: a real turtlebot driving around a laboratory, as shown in Figure 6-9. Given the high fidelity of turtlebot simulations in Gazebo, porting DCG-UPUP-Away code from a simulation environment to a real turtlebot environment is trivial. The only real modification necessary is retraining the object recognizer to recognize real world objects instead of simulated blocks. A proof-of-concept run was divided into 3 iteration, each demonstrating a unique capability. In the run, in order to simplify the job of the object recognizer, the known object types were boxes, helmets, and soap bottles. Both the image recognizer and natural language training examples were modified for the new types.

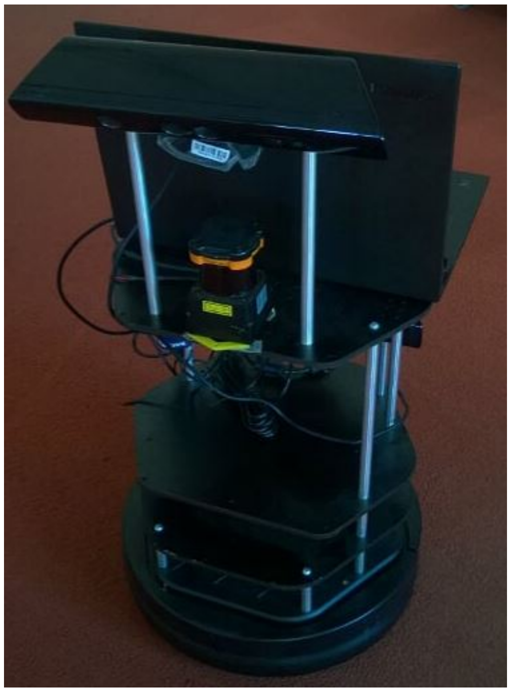


Figure 6-9: The turtlebot used to evaluated DCG-UPUP-Away in the real world. Only the top-mounted kinect was used to gather rgb images of the environment.

Hardware DCG

First, the robot demonstrates that the original DCG capabilities remain intact. The robot is placed in a scene with a box, a helmet, a bottle of soap, and a cone (the cone is unknown). After receiving the command "move towards the box," the turtlebot successfully drives to the box, as shown in Figure 6-10. This behavior demonstrates the capability to ground known phrases to known types.

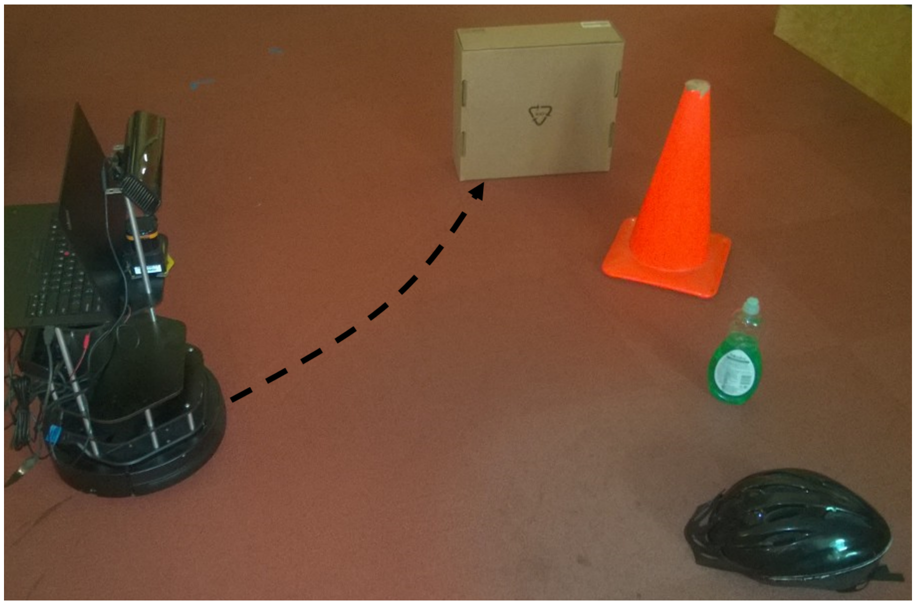


Figure 6-10: When given the command "move towards the box," the turtlebot approaches the box by grounding a known phrase to a known object.

Hardware DCG-UPUP

Second, the robot demonstrates the ability to associate unknown phrases with unknown percepts. The robot is placed in a scene with a box, a helmet, and a cone (the cone is still unknown). The turtlebot is given the command "move towards the cone." Because the cone is perceived as an unknown object and the phrase "the cone" is unknown, "cone" is grounded to the cone, and the robot drives to the cone, as in Figure 6-11. This demonstrates that DCG-UPUP has learned how to ground to unknowns.

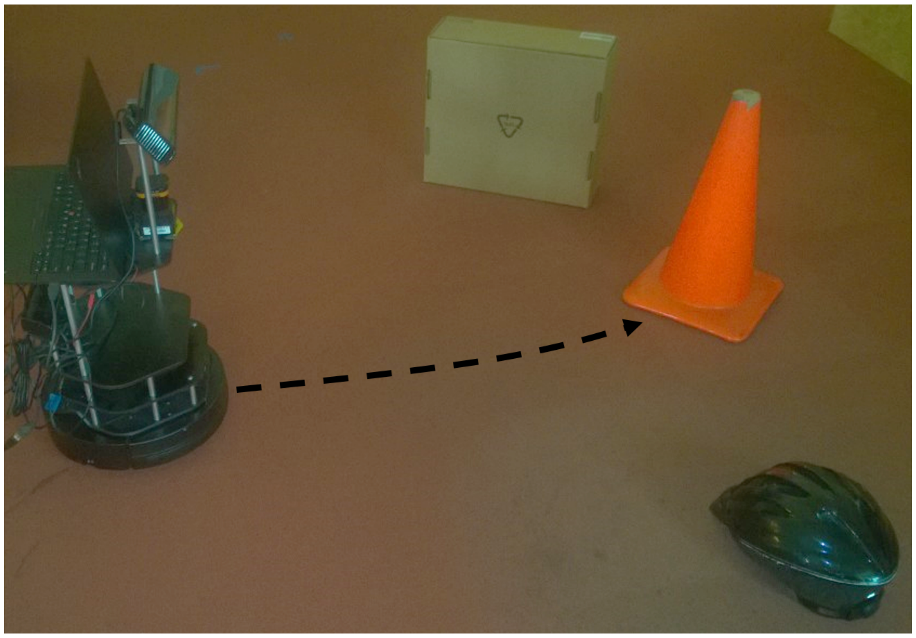


Figure 6-11: When given the command "move towards the cone," the turtlebot drives to the cone because "cone" is an unknown phrase and the physical cone is observed as unknown.

Hardware DCG-UPUP Learning

Third, the robot shows that it can permanently learn new symbols. By retraining after the second command (in which it grounded "cone" to a cone), the robot learns the symbol for a cone. In fact, when placed in a scene with a ball (which is unknown) and a cone (which is now known), the robot drives to the ball when it hears an unknown phrase such as "move to the ball." Conversely, in the same scene, if the robot gets the command "go to the cone," the robot drives to the cone. The divergent behaviors resulting from the different commands are both shown in Figure 6-12. The fact that the turtlebot behaves differently when grounding "cone" and "ball" shows that the robot must have learned what a cone is.

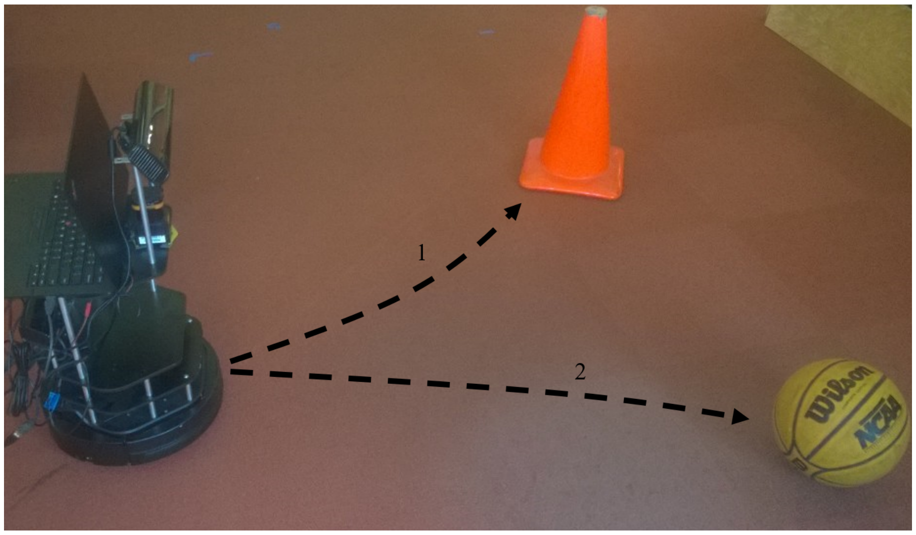


Figure 6-12: When given the command “move towards the cone,” the turtlebot follows path 1. When given the command “move towards the ball,” the turtlebot follows path 2. In order to distinguish between cones and balls, the turtlebot must have learned what a cone is.

Chapter 7

Conclusion

7.1 Contributions

There are two main contributions from the work presented in this thesis: 1) proposing a model, DCG-UPUP-Away for automatic symbol acquisition by grounding to unknowns, and 2) experimental validation of the model. The key idea behind the expanded model is that explicitly recognizing phrases or percepts as unknown allows for more intelligent behavior and unsupervised learning. In fact, the concept of recognizing unknowns could easily apply to many domains within robotics to increase robustness and learning capabilities in unknown environments.

Although such domains are left for future research, within the field of natural language grounding, DCG-UPUP-Away two novel capabilities. First, in randomly generated environments in which more than half of all objects are unknown to DCG-UPUP-Away ahead of time, DCG-UPUP-Away grounds to the correct object around 80% of the time. Second, by using naïve unsupervised learning between trials, DCG-UPUP-Away autonomously learns new symbols. This second point, in particular, appears important. One could imagine mass producing robots, training them with initial data that will be useful nearly everywhere, and then shipping off the robots to remarkably different environments. For example, one robot could go to an oil rig, while another robot, initially identical to the first, could work as a shop assistant in a tobacco store. Over time, however, the two robots will learn completely different meanings of the words "pipe" (one for pumping oil, the other for smoking out of) precisely because the phrases take on different meanings in the two contexts. This automatic divergence of knowledge because of separate experiences seems like precisely what will be necessary in robots of the future.

7.2 Recommendations for Future Work

There remains, of course, much work to be done before such robots arrive on everyone's doorstep. Some changes require minimal modifications to the model in order to incorporate more information into the grounding problem; other changes involve fundamentally restructuring the DCG-UPUP-Away framework.

First, the performance of DCG-UPUP-Away could be greatly improved by including additional sources of information. For example, online thesauruses or image catalogs could eventually lead to super-human performances. With access to the online Oxford English Dictionary, for example, DCG-UPUP-Away would never have to face the scenario of truly not knowing what a word means. Similarly, accessing Google Images could rapidly provide an object recognizer with enough training images to subsequently recognize newly learned symbols. With such corpora, DCG-UPUP-Away could search for synonyms to new phrase until it finds relates the unknown word to a phrase it already knows how to ground. Alternatively, image search could be used when grounding an unknown phrase to an unknown object: if the phrase "sofa" is unknown and the turtlebot perceives a sofa and a cone (both of which are perceived as unknown), Google Images could bias DCG-UPUP-Away to ground to the sofa by searching for "sofa" online.

Second, DCG-UPUP-Away could be fundamentally altered for even more autonomous symbol acquisition capabilities. For example, the domains of possible groundings could be increased to involve several hypothesized unknown objects (which may be necessary when trying to ground phrases such as “go to the cone next to the mailbox behind the wall.”) What could be most exciting, however, is the development of half-symbols. Recall that symbols provide a layer of abstraction between phrases and percepts. Currently, DCG-UPUP-Away only creates new symbols when grounding an unknown phrase to an unknown object. Thus, an entire symbol gets created atomically. Imagine, however, if DCG-UPUP-Away had the capability to manipulate and observe unknown objects before ever grounding a phrase to that object. Presumably, it could still group all those percepts into a symbol that has not yet been associated with a phrase. Similarly, DCG-UPUP-Away could scan pages of books and develop an extensive vocabulary before grounding words to objects. If such a system existed, it could run continually in a laboratory or even a household, only seeking input from humans occasionally to match language half-symbols to percept half-symbols. At that point, robotic behavior would start to match that of precocious children. That is a bright future for robotics and mankind, and a future that we have taken a preliminary step towards in this thesis.

Bibliography

[1] W. Adams, D. Perzanowski, and A.C. Schultz. Learning, storage and use of spatial information in a robotics domain. In *Proc. ICML 2000 Workshop Machine Learning Spatial Language*, pages 23–27, 2000.

[2] Jeremy M. Anglin, George A. Miller, and Pamela C. Wakefield. Vocabulary development: A morphological analysis. *Monographs of the Society for Research in Child Development*, 58(10):i–186, 1993.

[3] A. Aydemir, K. Sjoo, J. Folkesson, A. Pronobis, and P. Jensfelt. Search in the real world: Active visual object search based on spatial relations. In *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pages 2818–2824, May 2011.

[4] Herbert Bay, Tinne Tuytelaars, and Luc Van Gool. *SURF: Speeded Up Robust Features*, pages 404–417. Springer Berlin Heidelberg, Berlin, Heidelberg, 2006.

[5] M. Beetz, U. Klank, I. Kresse, A. Maldonado, L. Mosenlechner, D. Pangercic, T. Ruhr, and M. Tenorth. Robotic roommates making pancakes. In *Humanoid Robots (Humanoids), 2011 11th IEEE-RAS International Conference on*, pages 529–536, Oct 2011.

[6] Kristin P. Bennett and Colin Campbell. Support vector machines: Hype or hallelujah? *SIGKDD Explor. Newsl.*, 2(2):1–13, December 2000.

[7] Christopher M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2006.

[8] Paul Bloom. *How Children Learn the Meanings of Words*. MIT Press, Cambridge, MA, 2000.

[9] Mario Bollini, Stefanie Tellex, Tyler Thompson, Nicholas Roy, and Daniela Rus. *Interpreting and Executing Recipes with a Cooking Robot*, pages 481–495. Springer International Publishing, Heidelberg, 2013.

[10] S. R. K. Branavan, Harr Chen, Luke S. Zettlemoyer, and Regina Barzilay. 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: Volume*

1 - Volume 1, ACL '09, pages 82–90, Stroudsburg, PA, USA, 2009. Association for Computational Linguistics.

[11] A. Cervatiuc. Esl vocabulary acquisition: Target and approach. *The Internet TESL Journal.*

[12] Aneesh Chauhan and Luís Seabra Lopes. Using spoken words to guide open-ended category formation. *Cognitive Processing*, 12(4):341–354, 2011.

[13] I. Chung, O. Propp, M. R. Walter, and T. M. Howard. On the performance of hierarchical distributed correspondence graphs for efficient symbol grounding of robot instructions. In *Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ International Conference on*, pages 5247–5252, Sept 2015.

[14] Alexander Simon Clark. Unsupervised language acquisition: Theory and practice, 2001.

[15] Eve V. Clark. The principle of contrast: A constraint on language acquisition. In B. Macwhinney, editor, *Mechanisms of language acquisition*, pages 1–33. Lawrence Earlbaum, Hillsdale, NJ, 1987.

[16] Navneet Dalal and Bill Triggs. Histograms of oriented gradients for human detection. In *Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05) - Volume 1 - Volume 01*, CVPR '05, pages 886–893, Washington, DC, USA, 2005. IEEE Computer Society.

[17] Andrea F. Daniele, Mohit Bansal, and Matthew R. Walter. Natural language generation in the context of providing indoor route instructions. In *Proceedings of Robotics: Science and Systems*, Chicago, USA, 2016.

[18] Robin Deits, Stefanie Tellex, Pratiksha Thaker, Dimitar Simeonov, Thomas Kollar, and Nicholas Roy. Clarifying Commands with Information-Theoretic Human-Robot Dialog. *Journal of Human-Robot Interaction*, 2(2):58–79, 2013.

[19] Felix Duvallet, Matthew Walter, Thomas Howard, Sachithra Hemachandra, Jean Hyaejin Oh , Seth Teller, Nicholas Roy, and Anthony (Tony) Stentz . Inferring maps and behaviors from natural language instructions. In *International Symposium on Experimental Robotics*, June 2014.

[20] Karolina Eliasson. Case-based techniques used for dialogue understanding and planning in a human-robot dialogue system. In *In Proc. of the 20th Int. Joint Conf. on Artificial Intelligence (IJCAIâĂŹ07)*, 2007.

[21] Larry Fenson, Philip S. Dale, J. Steven Reznick, Elizabeth Bates, Donna J. Thal, Stephen J. Pethick, Michael Tomasello, Carolyn B. Mervis, and Joan Stiles. Variability in early communicative development. *Monographs of the Society for Research in Child Development*, 59(5):i–185, 1994.

[22] Terrence W Fong, Chuck Thorpe, and Charles Baur. Robot, asker of questions. *Robotics and Autonomous Systems*, 2003.

[23] Ruifang Ge and Raymond J. Mooney. A statistical semantic parser that integrates syntax and semantics. In *Proceedings of the Ninth Conference on Computational Natural Language Learning*, CONLL '05, pages 9–16, Stroudsburg, PA, USA, 2005. Association for Computational Linguistics.

[24] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2014.

[25] Kevin Gold, Marek Doniec, Christopher Crick, and Brian Scassellati. Robotic vocabulary building using extension inference and implicit contrast. *Artificial Intelligence*, 173(1):145 – 166, 2009.

[26] M. A. Hearst, S. T. Dumais, E. Osman, J. Platt, and B. Scholkopf. Support vector machines. *IEEE Intelligent Systems and their Applications*, 13(4):18–28, Jul 1998.

[27] Shirley Brice. Heath. *Ways with words : language, life, and work in communities and classrooms / Shirley Brice Heath.* Cambridge University Press Cambridge [Cambridgeshire] ; New York, 1983.

[28] Annette Herskovits. Semantics and pragmatics of locative expressions*. *Cognitive Science*, 9(3):341–378, 1985.

[29] T. Howard, S. Tellex, and N. Roy. A natural language planner interface for mobile manipulators. In *International Conference on Robotics and Automation*, June 2014.

[30] R. Jackendoff and R.S. Jackendoff. *Semantics and Cognition.* Current studies in linguistics series. de Gruyter, 1983.

[31] Thomas Kollar, Stefanie Tellex, Deb Roy, and Nicholas Roy. Toward understanding natural language directions. In *Proceedings of the 5th ACM/IEEE International Conference on Human-robot Interaction*, HRI '10, pages 259–266. IEEE Press, 2010.

[32] Thomas Kollar, Stefanie Tellex, Deb Roy, and Nicholas Roy. *Grounding Verbs of Motion in Natural Language Commands to Robots*, pages 31–47. Springer Berlin Heidelberg, Berlin, Heidelberg, 2014.

[33] Barbara Landau and Ray Jackendoff. “what” and “where” in spatial language and spatial cognition. *Behavioral and Brain Sciences*, 16:217–238, 6 1993.

[34] R.W. Langacker. *Foundations of Cognitive Grammar: Theoretical prerequisites*. Number v. 1 in Foundations of Cognitive Grammar. Stanford University Press, 1987.

[35] Dong C. Liu and Jorge Nocedal. On the limited memory bfgs method for large scale optimization. *Mathematical Programming*, 45(1):503–528, 1989.

[36] Wei Lu and Hwee Tou Ng. A probabilistic forest-to-string model for language generation from typed lambda calculus expressions. In *in Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 1611–1622, July 2011.

[37] James MacGlashan, Monica Babes-Vroman, Marie desJardins, Michael Littman, Smaranda Muresan, Shawn Squire, Stefanie Tellex, Dilip Arumugam, and Lei Yang. Grounding english commands to reward functions. In *Proceedings of Robotics: Science and Systems*, Rome, Italy, July 2015.

[38] Cynthia Matuszek, Nicholas FitzGerald, Luke S. Zettlemoyer, Liefeng Bo, and Dieter Fox. A joint model of language and perception for grounded attribute learning. In *International Conference on Machine Learning*. Omnipress, 2012.

[39] H. Moll and M. Tomasello. Twelve- and 18-month-old infants follow gaze to spaces behind barriers. In *British Journal of Developmental Psychology*, 2004.

[40] H. Moll and M. Tomasello. Level 1 perspective-taking at 24 months of age. In *Developmental Science*, 2006.

[41] Raymond J. Mooney. Learning to connect language and perception. In *Proceedings of the 23rd National Conference on Artificial Intelligence - Volume 3*, AAAI'08, pages 1598–1601. AAAI Press, 2008.

[42] Reinhard Moratz, Thora Tenbrink, John Bateman, and Kerstin Fischer. Spatial cognition iii. chapter Spatial Knowledge Representation for Human-robot Interaction, pages 263–286. Springer-Verlag, Berlin, Heidelberg, 2003.

[43] William O'Grady. *How Children Learn Language*. Cambridge University Press, 2005. Cambridge Books Online.

[44] Rohan Paul, Jacob Arkin, Nicholas Roy, and Thomas M. Howard. Efficient grounding of abstract spatial concepts for natural language interaction with robot manipulators. In *Proceedings of Robotics: Science and Systems*, AnnArbor, Michigan, June 2016.

[45] Marco A.F. Pimentel, David A. Clifton, Lei Clifton, and Lionel Tarassenko. A review of novelty detection. *Signal Processing*, 99:215 – 249, 2014.

[46] R. Ros, S. Lemaignan, E. A. Sisbot, R. Alami, J. Steinwender, K. Hamann, and F. Warneken. Which one? grounding the referent based on efficient human-robot interaction. In *19th International Symposium in Robot and Human Interactive Communication*, pages 570–575, Sept 2010.

[47] Deb K. Roy and Alex P. Pentland. Learning words from sights and sounds: a computational model. *Cognitive Science*, 26(1):113–146, 2002.

[48] N. Roy, J. Pineau, and S. Thrun. Spoken dialogue management using probabilistic reasoning. In *Proceedings of the 38th Annual Meeting of the Association for Computational Linguistics (ACL-2000)*, Hong Kong, 2000.

[49] Jeffrey Mark Siskind. Lexical acquisition in the presence of noise and homonymy. In *Proceedings of the 12th National Conference on Artificial Intelligence, Seattle, WA, USA, July 31 - August 4, 1994, Volume 1.*, pages 760–766, 1994.

[50] Marjorie Skubic, Pascal Matsakis, George Chronis, and James Keller. Generating multi-level linguistic spatial descriptions from range sensor readings using the histogram of forces. *Autonomous Robots*, 14(1):51–69, 2003.

[51] Radu Soricut and Daniel Marcu. Stochastic language generation using widl - expressions and its application in machine translation and summarization. In *in Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the ACL*, 2006.

[52] Eva Stopp, Klaus-Peter Gapp, Gerd Herzog, Thomas Laengle, and Tim C. Lueth. *Utilizing spatial relations for natural language access to an autonomous mobile robot*, pages 39–50. Springer Berlin Heidelberg, Berlin, Heidelberg, 1994.

[53] Ron Sun and C. Lee Giles. *Sequence Learning.* Springer-Verlag Berlin Heidelberg, 2001.

[54] S. Tellex, T. Kollar, S. Dickerson, M. Walter, A. Banerjee, S. Teller, and N. Roy. Understanding natural language commands for robotic navigation and mobile manipulation. In *National Conference on Artificial Intelligence*, 2011.

[55] J. G. Trafton, A. C. Schultz, M. Bugajska, and F. Mintz. Perspective-taking with robots: experiments and models. In *ROMAN 2005. IEEE International Workshop on Robot and Human Interactive Communication, 2005.*, pages 580–584, Aug 2005.

[56] J. Gregory Trafton, Nicholas L. Cassimatis, Magdalena D. Bugajska, Derek P. Brock, Farilee E. Mintz, and Alan C. Schultz. Enabling effective human-robot interaction using perspective-taking in robots. *IEEE Transactions on Systems, Man, and Cybernetics*, 35:460–470, 2005.

[57] Matthew R. Walter, Sachithra Hemachandra, Bianca Homberg, Stefanie Tellex, and Seth Teller. Learning semantic maps from natural language descriptions. In *Proceedings of Robotics: Science and Systems (RSS)*, Berlin, Germany, June 2013.

[58] Xiaoyu Wang, Ming Yang, Shenghuo Zhu, and Yuanqing Lin. Regionlets for generic object detection. In *The IEEE International Conference on Computer Vision (ICCV)*, December 2013.

[59] Thomas Williams, Rehj Cantrell, Gordon Briggs, Paul Schermerhorn, and Matthias Scheutz. Grounding natural language references to unvisited and hypothetical locations, 2013.

[60] Chen Yu and Dana H. Ballard. A multimodal learning interface for grounding spoken language in sensory perceptions. *ACM Trans. Appl. Percept.*, 1(1):57–80, July 2004.

[61] John M. Zelle and Raymond J. Mooney. Learning to parse database queries using inductive logic programming. In *AAAI/IAAI*, pages 1050–1055, Portland, OR, August 1996. AAAI Press/MIT Press.

[62] C. Lawrence Zitnick and Piotr Dollár. Edge boxes: Locating object proposals from edges. In *ECCV*. European Conference on Computer Vision, September 2014.