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

Mycal Tucker, Derya Aksaray, Rohan Paul, and Nicholas Roy

***Abstract*— abstract goes here**

I. Introduction

Robots must be able to understand natural language if they are to efficiently collaborate with humans. In order to achieve this goal, the problem of associating phrase with real-world objects and actions, commonly referred to as the grounding problem, has emerged as a focus of new research in robotics (cite G3,DCG, etc.).

While traditional solutions to the grounding problem may be trained to ground a fixed set of phrases to a fixed set objects, such methods fail to reason intelligently about unknown phrases or objects that have never been encountered in training. Furthermore, current techniques often assume that the location of the object being grounded to is known (e.g. phrases refer to objects that are currently perceived or localized within a known map). As a result, a robot will pick the most likely perceived grounding rather than exploring its surroundings.

To assume that language will be drawn from a fixed set and to assume that language only refers to known, perceived objects is unsafe in the real world. Humans regularly draw upon context-specific lexicons that the general population does not recognize; however, training a robot to know every possible meaning of every possible word is infeasible and inefficient. At the same time, humans often refer to objects whose locations are fundamentally unknown. Attempting to reason over the space of all possible maps, however, is similarly computationally infeasible.

In this paper, a new model called the Distributed Correspondence Graph - Unknown Phrase, Unknown Percept - Away (DCG-UPUP-Away), takes steps toward relaxing both of these assumptions by 1) explicitly modeling unknown phrases and unknown percepts and 2) creating hypothetical objects. These two changes yield a model that correctly grounds a large variety of phrases in challenging environments while autonomously learning new words and symbols. This anecdotal evidence is supported by a simulation study using commands generated by Amazon Mechanical Turk users. Overall, a turtlebot correctly grounds such commands roughly 80% of the time while learning new concepts in an unsupervised manner, despite initially being trained to recognize fewer than half of the phrases or objects.

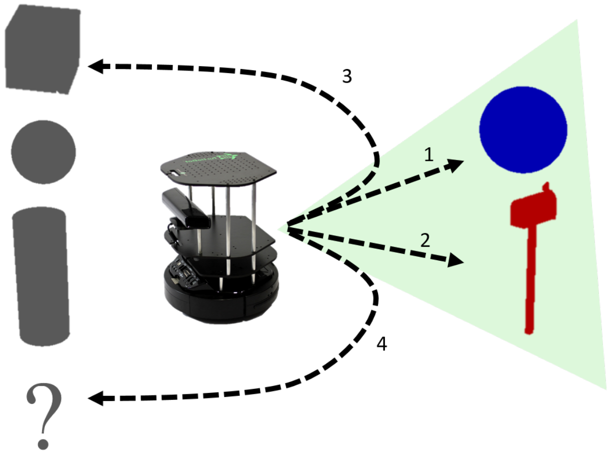


Fig. 1: In this example, DCG-UPUP-Away has been trained only with cubes, spheres, and cylinders. DCG-UPUP-Away may ground phrases to known perceived objects (1), unknown perceived objects (2), known hypothesized objects (3), or unknown hypothetical objects (4).

The remainder of this paper is organized as follows. Background information about probabilistic graphical models used for the grounding problem is introduced in Section II. The technical approach used in developing DCG-UPUP-Away is presented in Section III. The model is evaluated in Section **??**. Existing research in natural language robotics and human-robot interaction that complements this work are reviewed in Section V. Finally, Sections VI reviews the contributions of this paper as well as areas for future research.

II. Background

The work in this paper falls within the field of natural language grounding, which addresses the problem of correctly determining how phrases relate to the real world (e.g. the phrase “go to the cube” means approaching a physical cube). In the general formulation of the grounding problem, three sets must be considered: $\Gamma = \{\text{symbolic actions and objects}\}$ is the set of groundings, which represents what phrases may be grounded to; $\Lambda = \{\text{English phrases}\}$ is the set of phrases, which represents what phrases natural language sentences may be composed of; and $\Upsilon = \Gamma^O \times \{\text{object attributes}\}$ is the cartesian product of the set of objects (i.e. $\Gamma^O \subset \Gamma$) and the object

All authors are with the Computer Science and Artificial Intelligence Laboratory (CSAIL), Massachusetts Institute of Technology, Cambridge, MA 02139, USA {mycal,daksaray,rohanp,nickroy}@csail.mit.edu

It appears as if acknowledgments sometimes go here, but there’s also the acknowledgment section at the end. What’s the difference?

attributes (e.g. color, location), which represents the world model in which the phrases may be grounded.

Given a natural language command $\boldsymbol{\lambda}$, which is a vector of phrases from the set Λ and has a length of $|\boldsymbol{\lambda}|$, the general grounding problem can be formulated as a probability maximization problem

$$\boldsymbol{\gamma}^* = \underset{\boldsymbol{\gamma} \in \Gamma^{|\boldsymbol{\lambda}|}}{\arg\max}\, p(\boldsymbol{\gamma}|\boldsymbol{\lambda}, \Upsilon), \tag{1}$$

where $\boldsymbol{\gamma} \in \Gamma^{|\boldsymbol{\lambda}|}$ is a vector of groundings with a length of $|\boldsymbol{\lambda}|$, and Υ denotes the world model. In this formulation, the optimal vector of groundings $\boldsymbol{\gamma}^*$ is the one with maximum likelihood, given a command $\boldsymbol{\lambda}$ and a world model Υ.

In practice, the domains of Γ, Λ, and Υ in (1) typically include elements from previously seen examples. For example, rather than allowing the set of phrases Λ to include all words in a dictionary, Λ is generally assumed to only contain words that have appeared in the training examples. Moreover, solving (1) is a hard combinatorial optimization problem due to the diversity in language and world. One way to tackle this issue is to construct probabilistic graphical models based on the linguistic structure of the commands. For example, the Generalized Grounding Graph (G3) model [REF] is a factor graph that is trained from a corpus of labeled examples to ground language commands with objects, locations, and paths.

An alternative model is the Distributed Correspondence Graph (DCG) [REF], which infers the most likely set of planning constraints from language commands (rather than grounding phrases to particular actions, objects, or paths as in G3 model). In particular, DCG has the following properties: First, DCG decouples motion planning from the grounding problem. To this end, it discards the notion of grounding to specific actions or objects by instead only grounding to constraints relative to known objects (e.g. the area near a cube). Then, it passes the constraints to a motion planner. Second, DCG allows only the phrases composed entirely of the words that have already been encountered in training. Third, DCG only considers the perceived objects rather than a full model of the world including the unobserved objects. Fourth, ternary correspondence variables ϕ_{ij} are introduced to represent whether the i^{th} phrase λ_i from the overall command $\boldsymbol{\lambda}$ corresponds to a grounding γ_{ij}. For a given phrase λ_i, ϕ_{ij} is set to *Active* if λ_i refers to constraint γ_{ij} (e.g. the area near the cube), *Inverted* if λ_i refers to the opposite of γ_{ij} (e.g. the area far from the cube), or *Inactive* if λ_i has no bearing on γ_{ij} (e.g. the area near a sphere). Fifth, for the sake of computational efficiency, the overall inference is factored using conditional independence according to the structure of the parse tree of $\boldsymbol{\lambda}$.

Accordingly, the optimization problem solved over a DCG model becomes

$$\boldsymbol{\phi}^* = \underset{\phi_{ij} \in \boldsymbol{\phi}}{\arg\max} \prod_{i}^{|\boldsymbol{\lambda}|} \prod_{j}^{|\Phi_i|} p(\phi_{ij}|\gamma_{ij}, \lambda_i, \Gamma_{c_{ij}}, \Upsilon_{KP}), \tag{2}$$

where $\lambda_i \in \Lambda_{KN}$ is the i^{th} phrase in command $\boldsymbol{\lambda}$ and Λ_{KN} is the set of phrases with known (previously seen) words; Φ_i is the set of correspondence variables of λ_i, $\phi_{ij} \in \Phi_i$ is the j^{th} correspondence variable of λ_i; γ_{ij} is the j^{th} grounding of λ_i; $\Upsilon_{KP} = \{\text{object attributes}\} \times \Gamma_{KP}$ is the world model consisting of the object attributes and the set of known perceived symbolic objects Γ_{KP}; and $\Gamma_{c_{ij}}$ is the set of child groundings of γ_{ij}. Note that $\Gamma_{c_{ij}}$ may be formally defined as the set of groundings for the leftmost descendants of the immediate children (barring itself) of the parent of phrase λ_i in the parse tree of the natural language command.

For example, consider Figure 2 that shows the parse tree of a simple command, and Figure 3 that shows the corresponding DCG graphical model. The child grounding of the phrase "move" is the grounding for the phrase "to." Similarly, the child grounding of the phrase "to" is the grounding for the phrase "the cube" (determiners such as "the" may be collapsed into their nouns). Thus, in this example, each grounding has exactly one child grounding, yielding the inter-plate structure in Figure 3. Examining the parse tree also reveals why the factorization in Equation 2 is reasonable: the meaning "move" should be conditionally independent of the noun "cube" given the prepositional phrase. After all, the correct grounding of the word "move" is an action that does not depend on whether the target is a cube or a sphere, but it does depend on the position of the cube.

Finally, one must consider the factor function $\Psi : \Phi \times \Gamma \times \Lambda \times \Gamma \times \Upsilon \rightarrow \mathbb{R}$ within each plate that determines the most likely configuration of each $\phi_{ij} \in \Phi$ given $\gamma_{ij} \in \Gamma$, $\lambda_i \in \Lambda$, $\Gamma_{c_{ij}} \subset \Gamma$, and $\Upsilon_{KP} \subset \Upsilon$. Accordingly, one may rewrite (2) as follows:

$$\boldsymbol{\phi}^* = \underset{\phi_{ij} \in \boldsymbol{\phi}}{\arg\max} \frac{1}{Z} \prod_{i}^{|\boldsymbol{\lambda}|} \prod_{j}^{|\Phi_i|} \Psi(\phi_{ij}, \gamma_{ij}, \lambda_i, \Gamma_{c_{ij}}, \Upsilon_{KP}), \tag{3}$$

where Ψ is a log-linear model (LLM) composed of a weighted combination of hand-coded binary functions as

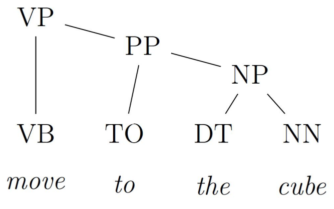


Fig. 2: Parse tree for the command "move to the cube"

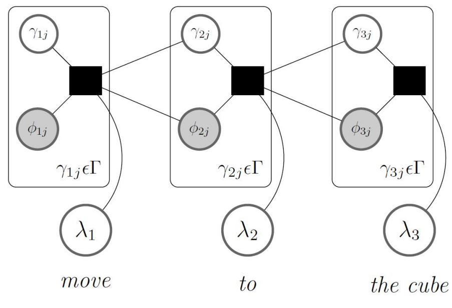


Fig. 3: DCG graphical model for the parse tree in Figure 2

follows:

$$\Psi(\phi_{ij},\gamma_{ij},\lambda_i,\Gamma_{c_{ij}},\Upsilon_{KP}) = \exp\Big(\sum_{f\epsilon F}\mu_f f(\phi_{ij},\gamma_{ij},\lambda_i,\Gamma_{c_{ij}},\Upsilon_{KP})\Big), \tag{4}$$

where each binary function f belongs to a set of hand-coded binary features that evaluate specific traits about a grounding (e.g. whether the word "cube" appears in $\boldsymbol{\lambda}$), and μ_f is the weighting of each f. In this work, the weights μ_f are learned in a training procedure via the Limited-memory Broyden-Fletcher-Goldfarb-Shanno algorithm.

III. Technical Approach

In the previous section, it has been stated that the DCG model can be efficiently used in grounding problems where the objects and phrases are known and the phrases are grounded to only perceived objects. In this paper, the proposed model DCG-UPUP-Away enables the solution of a more generalized grounding problem such that 1) the phrases and objects may be known or unknown, and 2) the phrases may be grounded to objects that are out of perception. To this end, the following sections detail how to ground unknown phrases or objects, how to incrementally learn new objects and phrases, how to hypothesize groundings out of perception, and how to involve adjective attributes into grounding to avoid ambiguities.

A. Symbol for Unknown [DA: Grounding Unknown Phrases or Objects]

here I introduce the notion of an unknown symbol
Here's the factored equation:

$$\boldsymbol{\phi}^* = \arg\max_{\boldsymbol{\phi}\epsilon\Phi^{|\boldsymbol{\lambda}|}} \frac{1}{Z}\prod_{ij}\Psi(\phi_i,\gamma_{ij},\lambda_i,\Gamma_{c_{ij}},\Upsilon_{KP}\cup\Upsilon_{UP}), \tag{5}$$

And the features:

$$\Psi() = \exp\Big(\sum_{f\epsilon F_{\text{DCG}}}\mu_f f(\phi_{ij},\gamma_{ij},\lambda_i,\Gamma_{c_{ij}},\Upsilon_{KP}\cup\Upsilon_{UP}) + ... \tag{6}$$

$$\sum_{f\epsilon F_{\text{Unknown}}}\mu_f f(\phi_{ij},\gamma_{ij},\lambda_i,\Gamma_{c_{ij}},\Upsilon_{KP}\cup\Upsilon_{UP})\Big). \tag{7}$$

B. Incremental Unsupervised Learning

Here I say how we use the unknown symbol to incrementally learn more symbols. There's no need for extra equations, but I could include something to show how domains are expanded. I think that's taken care of in the pseudocode, though.

C. Hypothesized Groundings out of Perception

Grounding out of world. Here are the equations:

$$\boldsymbol{\phi}^* = \arg\max_{\boldsymbol{\phi}\epsilon\Phi^{|\boldsymbol{\lambda}|}} \frac{1}{Z}\prod_{ij}\Psi(\phi_i,\gamma_{ij},\lambda_i,\Gamma_{c_{ij}},\Upsilon_{KP}\cup\Upsilon_{UP}\cup\Upsilon_{KH}\cup\Upsilon_{UH}), \tag{8}$$

And the features:

$$\Psi() = \exp\Big(\sum_{f\epsilon F_{\text{DCG}}}\mu_f f(\phi_{ij},\gamma_{ij},\lambda_i,\Gamma_{c_{ij}},\Upsilon_{KP}\cup\Upsilon_{UP}\cup\Upsilon_{KH}\cup\Upsilon_{UH}) + ... \tag{9}$$

$$\sum_{f'\epsilon F_{\text{Unknown}}}\mu_{f'} f'(\phi_{ij},\gamma_{ij},\lambda_i,\Gamma_{c_{ij}},\Upsilon_{KP}\cup\Upsilon_{UP}\cup\Upsilon_{KH}\cup\Upsilon_{UH}) + ... \tag{10}$$

$$\mu_{f_H} f_H(\phi_{ij},\gamma_{ij},\lambda_i,\Gamma_{c_{ij}},\Upsilon_{KP}\cup\Upsilon_{UP}\cup\Upsilon_{KH}\cup\Upsilon_{UH})\Big). \tag{11}$$

Then I can include this pseudocode:

Algorithm 1 My algorithm

```
procedure DCG-UPUP-AWAY
    $M \leftarrow$ new DCG-UPUP-Away
    $M.\Gamma \leftarrow$ init_groundings
    $M.F \leftarrow$ init_features
    $T \leftarrow$ init_training
    $M$.train($M.\Gamma, M.F, T$)
    while true do
        $\Upsilon \leftarrow$ perceive_objects(camera, $M.\Gamma$)
        $\Upsilon \leftarrow \Upsilon +$ hypothesize_objects($M.\Gamma$)
        $\boldsymbol{\lambda} \leftarrow$ get_nl_command()
        $[\boldsymbol{\phi}^*, \boldsymbol{\gamma}^*] \leftarrow M$.ground($\boldsymbol{\lambda}, \Upsilon$)
        if $\boldsymbol{\gamma}^*$.is_hypothesized() then
            spin_in_place()
        else
            drive_to($\boldsymbol{\gamma}^*$)
        $T_u \leftarrow$ gen_unsupervised_training($\boldsymbol{\lambda}, \boldsymbol{\gamma}^*, \Upsilon$)
        if is_unknown($\boldsymbol{\lambda}$)&!$\boldsymbol{\gamma}^*$.is_hypothesized() then
            $\gamma' \leftarrow$ new_grounding($\boldsymbol{\lambda}, \boldsymbol{\gamma}^*, \Upsilon$)
            $M.\Gamma \leftarrow M.\Gamma + \gamma'$
            $M.F \leftarrow M.F + f_{\text{word}}(\boldsymbol{\lambda}[\text{noun}])$
            $M.F \leftarrow M.F + f_{\text{obj}}(\boldsymbol{\gamma}^*[\text{obj}])$
            $T_u \leftarrow$ replace_unknown($T_u, \gamma'$)
        $T \leftarrow T + T_u$
        $M$.train($M.\Gamma, M.F, T$)
```

D. Adjective-Attribute Heuristics

Here I describe how color can be used to bias the correct grounding.

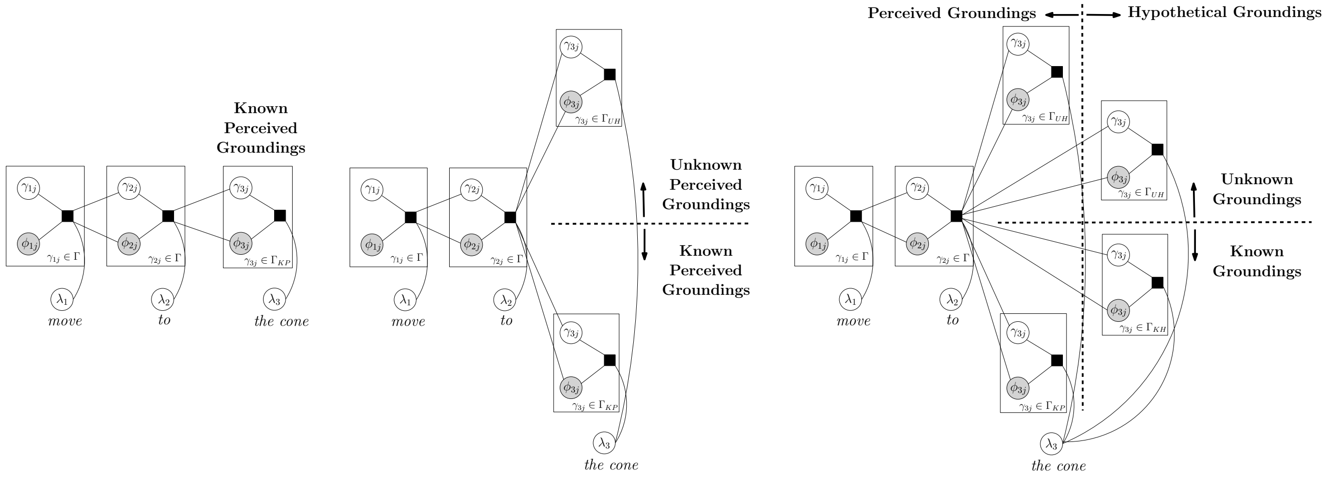
Here are the equations. The factored equation is the same as before, so I will leave it out. The features:

$$\Psi() = \exp\Big(\sum_{f\epsilon F_{\text{DCG}}}\mu_f f(\phi_{ij},\gamma_{ij},\lambda_i,\Gamma_{c_{ij}},\Upsilon_{KP}\cup\Upsilon_{UP}\cup\Upsilon_{KH}\cup\Upsilon_{UH}) + ... \tag{12}$$

$$\sum_{f'\epsilon F_{\text{Unknown}}}\mu_{f'} f'(\phi_{ij},\gamma_{ij},\lambda_i,\Gamma_{c_{ij}},\Upsilon_{KP}\cup\Upsilon_{UP}\cup\Upsilon_{KH}\cup\Upsilon_{UH}) + ... \tag{13}$$

$$\mu_{f_H} f_H(\phi_{ij},\gamma_{ij},\lambda_i,\Gamma_{c_{ij}},\Upsilon_{KP}\cup\Upsilon_{UP}\cup\Upsilon_{KH}\cup\Upsilon_{UH}) + ... \tag{14}$$

$$\sum_{f''\epsilon F_{\text{Color}}}\mu_{f''} f''(\phi_{ij},\gamma_{ij},\lambda_i,\Gamma_{c_{ij}},\Upsilon_{KP}\cup\Upsilon_{UP}\cup\Upsilon_{KH}\cup\Upsilon_{UH})\Big). \tag{15}$$

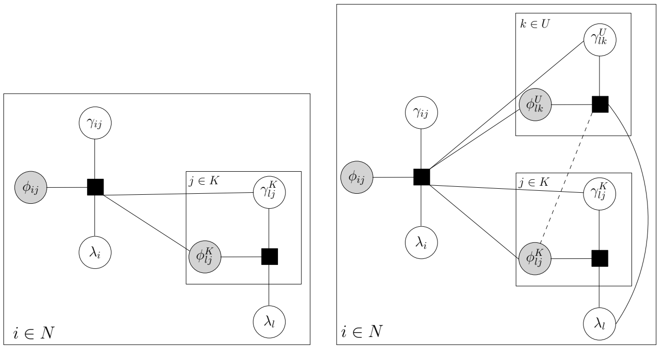


(a) The DCG model

(b) The DCG-UPUP model

(c) The DCG-UPUP-Away model

Fig. 4: The graphical models constructed for the command "*move to the cone*".



(a) Without explicit unknown symbols

(b) With explicit unknown symbols

Fig. 5: Factor graph representations. Input instruction is parsed into N phrases where λ_l represents a child phrase of parent phrase λ_i. Superscripts K and U denote known and unknown variables, respectively.

IV. Evaluation

DCG-UPUP-Away is evaluated in two experiments. First, a simulated turtlebot within randomly generated simulated environments is given a series of user-generated natural language commands. Second, a physical turtlebot is given specific commands in a laboratory environment in order to demonstrate novel behaviors enabled by DCG-UPUP-Away. Both experiments assume a perfect object recognizer that translates raw sensor data into a world model Υ that may be used by DCG-UPUP-Away, as well as an initial set of hand-labeled training examples for training the LLM to ground cubes, spheres, and cylinders. In all trials, training DCG-UPUP-Away with 53 positive examples took less than 1 minute on a Lenovo Thinkpad X1 Carbon (TODO get more specs), and grounding a command took under 40 seconds.

A. Experimental Setup

The simulated testing environments are randomly generated in Gazebo. 10 worlds are created, and each is populated with a random collection of objects in randomized locations. There are 8 possible object types (including cubes, spheres, and cylinders) in 3 possible colors, for a total of 24 objects. Each object has a 15% chance of being added to a given map. Using such a procedure to generate environments, when coupled with the limited field of view of the turtlebot, caused 87% of objects to be placed outside the initial field of view of the robot, demonstrating the need for the ability to ground commands to hypothesized objects.

After generating the 10 maps, screenshots of world with a single object highlighted are uploaded to Amazon Mechanical Turk. For each image, users were instructed to write a command "for approaching the highlighted object." These image-command pairs were saved for evaluating whether a robot, when placed in the corresponding simulated world and given the natural language command, successfully approaches the correct object. An example screenshot, with an annotation supplied by a user, is shown in Figure 6.

10 image-command pairs are randomly selected without replacement from the pool of all pairs. The 10 ordered pairs constitute one trial; each specific pair is dubbed one iteration. 30 trials are generated, each consisting of 10 iterations, for a total of 300 evaluations. When executing a trial, the turtlebot is first trained on the initial, hand-curated training set. The turtlebot is then given the natural language command from the first iteration, and then retrained using the initial data supplemented by unsupervised training examples generated by the first iteration. The retrained turtlebot is given the command from the next iteration, and appropriately retrained after each execution until all 10 iterations have been executed.

The metrics we consider are the grounding accuracy (how likely DCG-UPUP-Away is to correctly ground a phrase)

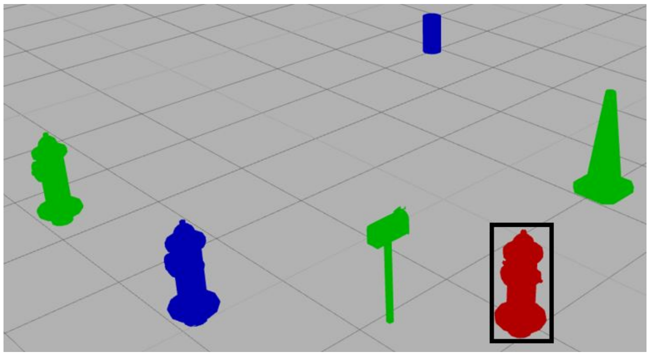


Fig. 6: A simulated world with a highlighted object presented on Amazon Mechanical Turk, labeled by a user as "Move to the red fire hydrant."

and the number of known symbols. We further divide the grounding accuracy results to examine when phrases are grounded to known, unknown, or learned objects.

B. Grounding Accuracy

The primary metric used in evaluating the success of DCG-UPUP-Away is the grounding accuracy: how likely is the turtlebot to correctly execute the natural language command. Because the turtlebot is retrained between iterations, grounding accuracy may change as a function of iteration number. In fact, mean grounding accuracy remains between 70% and 90% across all iterations, as shown in Figure 7.

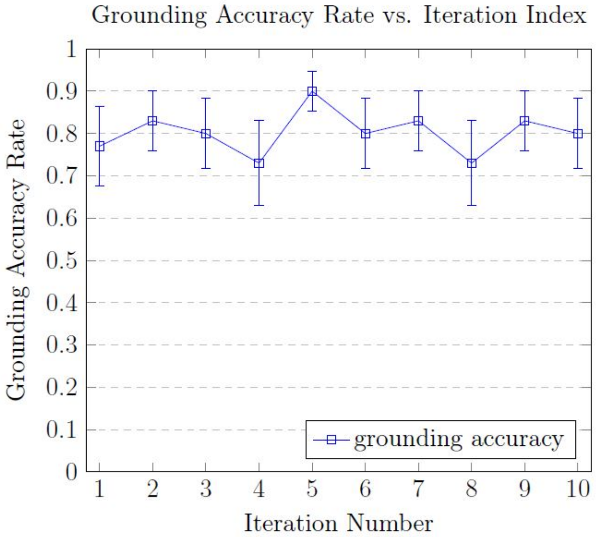


Fig. 7: this is the overall grounding accuracy

Although the overall grounding accuracy remains relatively constant, the underlying behavior within DCG-UPUP-Away changes over the course of a trial. In Figure 8, we plot 3 curves, showing what fraction of correctly grounded phrases refer to known objects, unknown objects, or learned objects as a function of iteration number. In the first iteration, nearly 70% of correctly grounded commands refer to known objects, but by the 10th iteration that number has fallen to nearly 10%, replaced almost entirely by correctly grounding to learned objects.

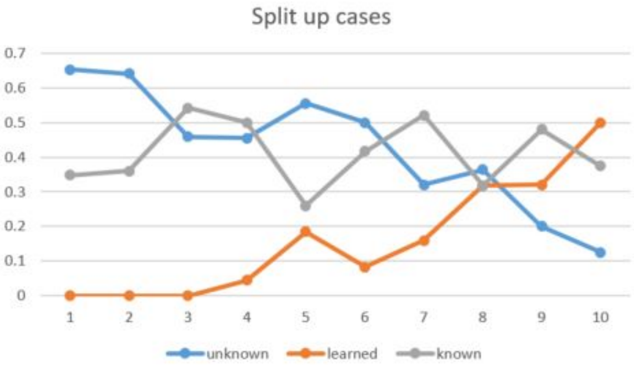


Fig. 8: this is split up (must replot well)

C. Learned Symbols

In order to better examine the learning behavior exhibited by DCG-UPUP-Away, the other metric considered is the number of correctly known symbols. (Symbols may be incorrectly learned by associating a phrase with the wrong sort of object.) Initially, the turtlebot is trained with cubes, spheres, and cylinders, but environments may contain up to 5 additional object types (fire hydrants, drills, mailboxes, door handles, and traffic cones). Whenever an unknown phrase is grounded to such an unknown object, the turtlebot learns the new symbol. Thus, one may calculate the expected number of known symbols as a function of iteration number using combinatorics to count how many unknown objects are present. The recorded number of correctly learned symbols are plotted in Figure 9 in blue, as well as the expected number in red.

(Asking for advice: the x axis ranges from 1 to 11 because I can retrain after the 10th iteration and increase the mean number of symbols learned. It's a small increase, though, so I wouldn't be too upset cutting off the 11th iteration, though.)

As expected, the blue curve starts at 3 (for the cube, sphere, and cylinder), and stochastically monotonically increases. In 10% of trials, all 8 symbols were correctly learned; in other trials DCG-UPUP-Away incorrectly grounded unknown phrases (and therefore learned an incorrect symbol) or the 10 iterations collectively never referred to the five initially unknown objects, preventing DCG-UPUP-Away from ever learning the new symbol. Furthermore, learning symbols correctly improves grounding accuracy: for each additional correctly learned symbol, the turtlebot is over 4% more likely to correctly ground a command. (TODO generate p values via ANOVA.)

D. Hardware Demonstration

Outside the simulation environment, DCG-UPUP-Away was tested on a physical turtlebot in a laboratory setting. The turtlebot was placed facing a cylinder (known) and a cone (unknown). In addition, a cube (known) and a crate

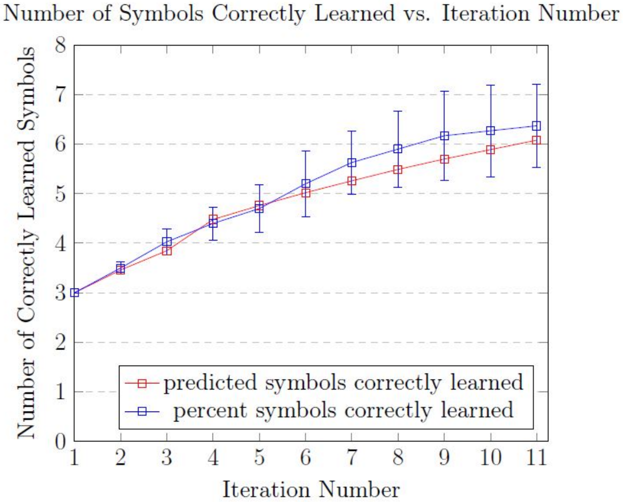


Fig. 9: this is how we show that we learn symbols. I will update this label (and caption) of this figure. also must use std error bars instead of variance

(unknown) were located behind the turtlebot. All objects were labeled with AR-track tags, which, in conjunction with (cite AR track package) was used in order to generate the world model Υ from a kinect mounted on the turtlebot.

Four natural language commands were used to demonstrate the full range of abilities of DCG-UPUP-Away. First, the turtlebot was given the command “move towards the cube.” The turtlebot successfully drove to the cube, demonstrating a correct grounding to a known, perceived object.

Second, the turtlebot was given the command “move towards the cone.” The turtlebot drove to the cone, demonstrating that it perceived the cone as unknown, recognized the phrase “cone” as unknown, and grounded the unknown phrase to the unknown object. Thus, a command was correctly grounded to an unknown, perceived object.

Third, the turtlebot was given the command “move towards the cube.” The turtlebot rotated in place until the cube came in view, and then approached the cube. In other words, the command was first grounded to a known, hypothesized object and then, once then cube was perceived, to a known, perceived object.

Fourth, the turtlebot was given the command “move towards the crate.” Once again, the turtlebot rotated in place, this time until it saw the crate, whereupon it drove to the crate. This demonstrates two important behaviors: 1) the turtlebot must have learned what a cone was, otherwise the unknown phrase (“crate”) would have been grounded to the cone and 2) the turtlebot grounded the command to an unknown, hypothesized object until the crate was perceived. The execution of this last command, including images of the physical behavior of the turtlebot as well as the model used for grounding, is shown in Figure 10.

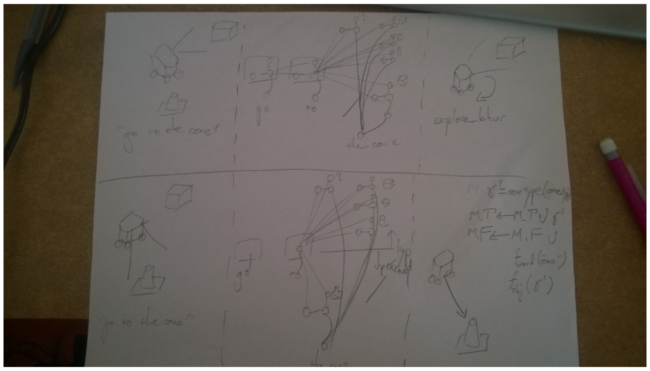


Fig. 10: this is a sketch of the 6 subfigures that demonstrate hypothesized groundings on hardware and in the model, and shows how it learns. I’d like this to go across the top of the page.

E. Limitations

Although the successful execution of natural language commands has been discussed in great depth, the failures of DCG-UPUP-Away are equally important in characterizing the limits of its performance.

Specifically, the most obvious limitation is that DCG-UPUP-Away assumes that, in the absence of additional information, unknown phrases refer to the first perceived unknown object. Strategies to relax this assumption by associating language adjectives with object properties have been explored in Section (TODO color section), but so far only a relatively simple method has been used.

In addition, DCG-UPUP-Away assumes a one-to-one correspondence between unknown phrases and unknown objects; it cannot, for example, learn synonyms by grounding unknown phrases to known types.

V. Related Works

related work goes here

VI. Conclusion

conclusion goes here. this includes future work

Acknowledgments

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