Machine Learning Journal manuscript No.
(will be inserted by the editor)

Learning Perceptually Grounded Word Meanings From Unaligned Parallel Data

Stefanie Tellex · Pratiksha Thaker · Joshua Joseph · Nicholas Roy

Received: date / Accepted: date

Abstract In order for robots to effectively understand natural language commands, they must be able to acquire meaning representations that can be mapped to perceptual features in the external world. Previous approaches to learning these *grounded* meaning representations require detailed annotations at training time. In this paper, we present an approach to grounded language acquisition which is capable of jointly learning a policy for following natural language commands such as "Pick up the tire pallet," as well as a mapping between specific phrases in the language and aspects of the external world; for example the mapping between the words "the tire pallet" and a specific object in the environment. Our approach assumes a parametric form for the policy that the robot uses to choose actions in response to a natural language command that factors based on the structure of the language. We use a gradient method to optimize model parameters. Our evaluation demonstrates the effectiveness of the model on a corpus of "pick up" and "go to" commands given to a robotic forklift by untrained users.

Keywords robotics · language · machine learning · probabilistic graphical models

1 Introduction

In order for robots to robustly understand human language, they must have access to representations capable of mapping between symbols in the language and aspects of the external world which are accessible via the robot's model of its environment. Previous symbolic approaches have represented word meanings as symbols in some specific symbolic language, either programmed by hand [Winograd, 1971, MacMahon et al., 2006] or learned [Matuszek et al., 2010, Chen and Mooney, 2011, Liang et al., 2011]. Because word meanings

are represented as symbols, rather than perceptually grounded features, the mapping between these symbols and the external world must still be defined *a priori*.

Although symbols are required to deal with some linguistic concepts such as negation, quantifiers, and determiners, it is also necessary to map between words in the language and non-symbolic, perceptual aspects of the external world. Language grounding approaches address this problem by mapping words in the language to *groundings* in the external world [Mavridis and Roy, 2006, Hsiao et al., 2008, Kollar et al., 2010, Tellex et al., 2011]. Groundings are the specific physical concepts that are referred to by the language and can be objects (e.g., a truck or a door), places (e.g., a particular location in the world), paths (e.g., a trajectory through the environment), or events (e.g., a sequence of actions taken by the robot).

Recent work has demonstrated how to learn grounded word meanings from a parallel corpus of natural language commands paired with groundings in the external world [Tellex et al., 2011]. However, learning the model parameters required that the parallel corpus be augmented with additional annotations specifying the alignment between specific phrases in the language and corresponding groundings in the external world. Figure 1 shows an example command from the training set paired with these alignment annotations, represented as arrows pointing from each linguistic constituent to a corresponding grounding. Approaches that do not require these augmentations, such as Branavan et al. [2009] or Vogel and Jurafsky [2010], do not capture the nested hierarchical structure of language.

Our aim is to relax these annotation requirements and develop an algorithm that learns perceptually grounded, compositional word meanings from an *unaligned* parallel corpus. By focusing on human-robot interaction domains, where the human partner is giving the robot instructions for some task, we can assume that the language specifies a high-level action that the robot should perform. For supervision, the algorithm receives only knowledge of the correct action, rather than requiring individual labels for all the linguistic constituents during training, as in previous work. Our system takes as input a state/action space for the robot defining a space of possible groundings and available actions in the external world. In addition it requires a training corpus of natural language commands paired with a demonstration of the robot correctly executing the action in the environment. For example, an entry in the corpus consists of a natural language command such as "Pick up the tire pallet" given to a robotic forklift, paired with a video or log of the robot's actions as drives to the tire pallet, inserts its forks, and raises it off the ground, drives to the truck, and sets it down.

To learn from an unaligned corpus, we derive a new training algorithm for the Generalized Grounding Graph (G^3) framework [Tellex et al., 2011] that performs stochastic gradient descent in the model parameters, based on the policy gradient method described by Branavan et al. [2009]. Our aim is to find model parameters such that, as the robot executes the actions predicted by the model, its probability of choosing the correct action is maximized. We

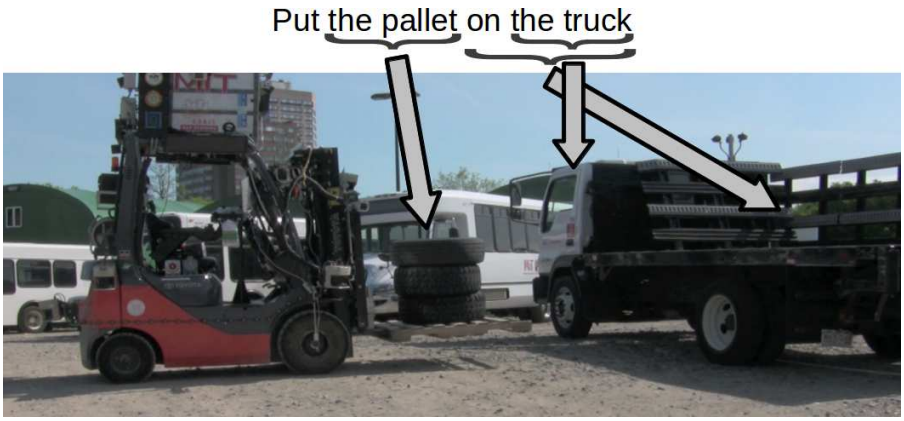
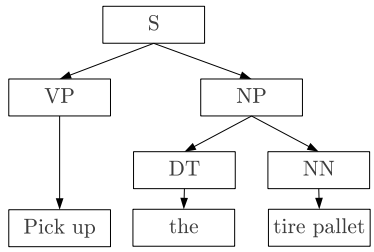


Fig. 1: Sample entry from an aligned corpus, where mappings between phrases in the language and groundings in the external world are explicitly specified as arrows. Learning the meaning of "the truck" and "the pallet" is challenging when alignment annotations are not known.

assume a parametric form for the model that factors according to the linguistic structure of the natural language command. The system searches for model parameters that maximize expected reward using stochastic gradient descent. By factoring the distribution over actions according to the structure of language, we can compute an appropriate gradient update for each factor, allowing the system to infer groundings for each linguistic constituent even without direct supervision. We evaluate our model in a human-robot interaction domain using a dataset of natural language commands given to a robotic forklift, collected from untrained users on the internet. Commands direct a robotic forklift to pick up objects or drive to locations in the environment. The evaluation demonstrates that, given a natural language command issued in a particular context, the model is able to infer actions for the robot as well as mappings between noun phrases in the command and objects in the environment, despite having no direct supervision for noun phrase groundings during training.

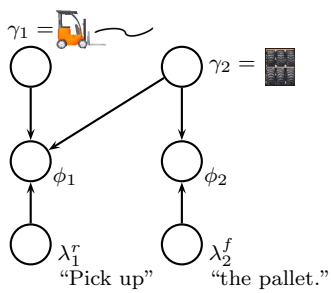
2 Background

We briefly review the G^3 framework, introduced by Tellex et al. [2011]. In order for a robot to understand natural language, it must be able to map between words in a command such as "Pick up the tire pallet," and corresponding aspects of the external world. Each constituent phrase in the command refers to a particular object, place, or action that the robot should take in the environment; together we refer to these as *groundings*. The aim of the G^3 framework is to find the most probable groundings, $\gamma_1 \ldots \gamma_N$, given a parsed natural lan-



st

(a) Parse tree.



(b) Grounding graph.

Fig. 2: Parse tree and grounding graph for the command, "Pick up the tire pallet." Random variables and edges are created in the graphical model for each constituent in the parse tree. The λ variables correspond to language; the γ variables correspond to groundings in the external world. Edges in the graph are created according to the parse structure of the command

guage command Λ, and the robot's model of the environment, M:

$$\underset{\gamma_1 \ldots \gamma_N}{\operatorname{argmax}}\, p(\gamma_1 \ldots \gamma_N | \Lambda, M) \tag{1}$$

The environment model M consists of the robot's location along with the locations and geometries of objects in the external world. A robot computes the environment model using sensor input. The computed model defines a space of possible values for the grounding variables, $\gamma_1 \ldots \gamma_N$. Formally, each γ_i is a tuple, (g, t, p), where:

- g is a bounding prism. It is expressed as a set of points which define a polygon, $(x_1, y_1), \ldots, (x_N, y_N)$, together with a height, z.
- t is a set of pre-defined textual tags, $\{tag_1, \ldots, tag_M\}$, which are the output of perceptual classifiers.
- $p \in \mathbb{R}^{T \times 7}$ is a sequence of T points. Each point is a pose for the robot. It consists of a tuple $(\tau, x, y, z, roll, pitch, yaw)$ representing the location and

orientation of the grounding at time τ. Locations between two times are interpolated linearly.

Groundings are objects, places, paths or events in the external world. An object such as a pallet is represented in the map as a three-dimensional geometric shape, along with a set of symbolic tags that might be produced by an object classifier, such as "pallet" or "truck." A place grounding represents a particular location in the map, and would have no tags and a zero-length trajectory. A path grounding consists of the trajectory of a point-based prism. An event grounding might consist of the robot and its trajectory over time, as it picks up a pallet.[1] The type of a grounding variable is inferred from the associated syntactic constituent using a template-based algorithm. We assume that the robot has access to the environment model whenever it infers groundings for a natural language command. For brevity, we omit M from future equations in this section.

To learn the distribution in Equation 1, one standard approach is to factor it based on certain independence assumptions, then train models for each factor. Natural language has a well-known compositional, hierarchical argument structure [Jackendoff, 1983], and a promising approach is to exploit this structure in order to factor the model. The G^3 framework takes as input a natural language command and uses the parse structure of the language to define the random variables and factors in a graphical model.

However, if we define a directed model over these variables (e.g., $p(\gamma_1 \dots \gamma_N | \Lambda)$) we must assume a possibly arbitrary order to the conditional γ_i factors. For example, for a phrase such as "the tire pallet near the other skid," we could factorize in either of the following ways:

$$p(\gamma_{\text{tires}}, \gamma_{\text{skid}} | \Lambda) = p(\gamma_{\text{skid}} | \gamma_{\text{tires}}, \Lambda) \times p(\gamma_{\text{tires}} | \Lambda) \tag{2}$$

$$p(\gamma_{\text{tires}}, \gamma_{\text{skid}} | \Lambda) = p(\gamma_{\text{tires}} | \gamma_{\text{skid}}, \Lambda) \times p(\gamma_{\text{skid}} | \Lambda) \tag{3}$$

Depending on the order of factorization, we will need different conditional probability tables that correspond to the meanings of words in the language. To resolve this issue, another approach is to use Bayes' Rule to estimate $p(\Lambda | \gamma_1 \dots \gamma_N)$, but this approach would require normalizing over all possible words in the language Λ. Another alternative is to use an undirected model such as a conditional random field, but this approach is intractable because it requires normalizing over all possible values of all γ_i variables in the model, including continuous attributes such as location and size.

To address these problems, the G^3 framework introduces a correspondence vector Φ to capture the dependency between $\gamma_1 \dots \gamma_N$ and Λ. Each entry in $\phi_i \in \Phi$ corresponds to whether linguistic constituent $\lambda_i \in \Lambda$ corresponds to the groundings associated with that constituent. For example, the correspondence variable would be $True$ for the phrase "the tire pallet" and a grounding of an actual pallet containing tires, and $False$ if the grounding was a different object,

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

such as a generator pallet or a truck. We assume that $\gamma_1 \dots \gamma_N$ are independent of Λ unless Φ is known. Introducing Φ enables factorization according to the structure of language with local normalization at each factor over a space of just the two possible values for ϕ_i. At inference time, these locally normalized factors can be simply multiplied together without the need to compute a global normalization constant, as would be required for a Markov random field or conditional random field.

Using the correspondence variable, we can write:

$$\underset{\gamma_1 \dots \gamma_N}{\operatorname{argmax}}\, p(\gamma_1 \dots \gamma_N | \Phi, \Lambda) \tag{4}$$

To perform inference, the value of the correspondence variable vector Φ is $True$, indicating that the inference is searching for groundings which correspond to the words in the natural language command. When the correspondence vector takes on the value of $True$, this maximization is identical to equation 1, except that Φ is listed as a given variable. Listing Φ as a given variable makes explicit the assumption that all linguistic constituents and associated groundings correspond. This inference is equivalent to maximizing the joint distribution of all groundings $\gamma_1 \dots \gamma_N$, Φ and Λ:

$$\underset{\gamma_1 \dots \gamma_N}{\operatorname{argmax}}\, p(\gamma_1 \dots \gamma_N, \Phi, \Lambda). \tag{5}$$

Note that even though Equation 5 uses the joint distribution, Λ and Φ are fixed, and we optimize over $\gamma_1 \dots \gamma_N$. Next we rewrite as a conditional distribution on Φ multiplied by a prior:

$$\underset{\gamma_1 \dots \gamma_N}{\operatorname{argmax}}\, p(\Phi | \Lambda, \gamma_1 \dots \gamma_N) p(\Lambda, \gamma_1 \dots \gamma_N) \tag{6}$$

We assume that Λ and $\gamma_1 \dots \gamma_N$ are independent when Φ is not known, as in the graphical model shown in Figure 2, yielding:

$$\underset{\gamma_1 \dots \gamma_N}{\operatorname{argmax}}\, p(\Phi | \Lambda, \gamma_1 \dots \gamma_N) p(\Lambda) p(\gamma_1 \dots \gamma_N) \tag{7}$$

This independence assumption is justified because if we do not know whether $\gamma_1 \dots \gamma_N$ correspond to Λ, then the language does not tell us anything about the groundings.

Finally, for simplicity, we assume that any object in the environment is equally likely to be referenced by the language, which amounts to a constant prior on $\gamma_1 \dots \gamma_N$.[2] We ignore $p(\Lambda)$ since it does not depend on $\gamma_1 \dots \gamma_N$, leading to:

$$\underset{\gamma_1 \dots \gamma_N}{\operatorname{argmax}}\, p(\Phi | \Lambda, \gamma_1 \dots \gamma_N) \tag{8}$$

[2] In the future, we plan to incorporate models of attention and salience into this prior.

We factor the model according to the hierarchical, compositional linguistic structure of the command:

$$p(\Phi|\Lambda, \gamma_1 \dots \gamma_N) = \prod_i p(\phi_i|\lambda_i, \gamma_{i_1} \dots \gamma_{i_k}) \tag{9}$$

The specific random variables and dependencies are automatically extracted from the parse tree and constituent structure of the natural language command; the details of this factorization are described formally by Tellex et al. [2011]. Parses can be extracted automatically, for example with the Stanford Parser [de Marneffe et al., 2006] or annotated using ground-truth parses. We call the resulting graphical model the *grounding graph* for a natural language command. Figure 2 shows the parse tree and graphical model generated for the command "Pick up the pallet." The random variable ϕ_2 is associated with the constituent "the pallet" and the grounding variable γ_2. The random variable ϕ_1 is associated with the entire phrase, "Pick up the pallet" and depends on both grounding variables: γ_1, which is the action that the robot takes, and its argument, γ_2, which is the object being manipulated. The λ_i variables correspond to the text associated with each constituent in the parse tree.

We assume that each factor takes a log-linear form with feature functions f_j and feature weights θ_j.

$$p(\Phi|\Lambda, \gamma_1 \dots \gamma_N) = \prod_i \frac{1}{Z} \exp\left(\sum_j \theta_j f_j(\phi_i, \lambda_i, \gamma_{i_1} \dots \gamma_{i_k})\right) \tag{10}$$

This function is convex and can be optimized with gradient-based methods [McCallum, 2002]. Training data consists of a set of natural language commands together with positive and negative examples of groundings for each constituent in the command. We refer to this dataset as an *aligned* corpus because each constituent in the command is aligned with a corresponding grounding in the environmental context. In contrast, the aim of the current work is to remove the requirement for this extra alignment annotation.

Features correspond to the degree to which the $\gamma_1 \dots \gamma_N$ correctly ground λ_i. These features define a *perceptual representation* in terms of a mapping between the grounding and words in the language. For example, for a relation such as "on," a natural feature is whether the grounding corresponding to the head noun phrase is supported by the grounding corresponding to the argument noun phrases. However, the feature $supports(\gamma_i, \gamma_j)$ alone is not enough to enable the model to learn that "on" corresponds to $supports(\gamma_i, \gamma_j)$. Instead we need a feature that also takes into account the word "on:"

$$supports(\gamma_i, \gamma_j) \wedge (\text{“on”} \in \lambda_i) \tag{11}$$

Thus features consist of the Cartesian product of perceptual features such as *supports* crossed with the presence of words in the linguistic constituent associated with the corresponding factor in the grounding graph.

The system follows natural language commands by optimizing the objective in Equation 8. It carries out approximate inference by performing beam search over $\gamma_1 \dots \gamma_N$. It searches over possible bindings for these variables in the space of values defined in the environment model M. It then computes the probability of each assignment using Equation 8; the result is the maximum probability assignment of values to all the variables $\gamma_1 \dots \gamma_N$. Although we are using $p(\Phi|\Lambda, \gamma_1 \dots \gamma_N)$ as the objective function, Φ is fixed, and the $\gamma_1 \dots \gamma_N$ are unknown. This approach is valid because, given our independence assumptions, $p(\Phi|\Lambda, \gamma_1 \dots \gamma_N)$ corresponds to the joint distribution over all the variables given in Equation 5. We discretize the space of possible groundings to make this search problem tractable. If no correct grounding exists in the space of possible values, then the system will not be able to find the correct value; in this case it will return the best value that it found.

3 Approach

Our goal is to learn model parameters for the G^3 framework from an unaligned corpus of natural language commands paired with robot actions. Previously, the system learned model parameters Θ using an aligned corpus in which values for all grounding variables, $\gamma_1 \dots \gamma_N$, were known at training time, and annotators provided both positive and negative examples for each factor. If the alignments between language and grounding variables are not known, then Equation 9 becomes ill-posed. Specifically, if the alignments are not known, then it is not known which grounding variables to instantiate for the $\gamma_{i_1} \dots \gamma_{i_k}$ that match each λ_i. If the weights are known, then at inference time, one can search for the alignments of λ_i to γ_i that maximize the posterior likelihood of Φ. During training, however, learning the weights can no longer be posed as the same convex supervised learning problem. Providing the alignments in the training data is a fundamental limit of the scalability of the approach. Examples of natural language commands matched with demonstrated robot actions can be easily collected from untrained users at a large scale using crowdsourcing such as Amazon Mechanical Turk. In contrast, labeling alignments requires trained annotators, takes more time to label each example, and raises issues of inter-rater agreement.

We now describe how to relax the annotation requirement so that only the top-level action for each command needs to be observed in order to train the model. We are given a corpus of D training examples. Each example d consists of a four-tuple, $(\Lambda^d, \Gamma^d, M^d, g_a^d)$ where:

- Λ^d is a parsed natural language command.
- Γ^d is a vector of grounding variables $\gamma_1 \dots \gamma_N$.
- M^d is an environmental context, consistent of a set of objects with a location, geometry, and set of perceptual tags.
- g_a^d is a demonstration of the robot correctly executing the natural language command.

Each environmental context will have a different set of objects and available actions for the robot. For example, in the robotic forklift domain, one training example might contain a command given in an environment with a single tire pallet; another might contain a command given in an environment with two box pallets and a truck. We define a discrete space of actions for the robot: it can drive near objects in the environment, pick them up, and put them down in discrete locations. The domain of possible values for a grounding variable $\gamma \in \Gamma^d$ is specified by the environmental context, but the actual values for these variables are not known.

We augment each example with a key piece of supervision: a demonstration of the robot correctly following a natural language command. This demonstration corresponds to an observed value g_a^d for a specific grounding variable in the graph. This variable, γ_a, corresponds to the action sequence the robot should execute when following the entire natural language command. We refer to this variable as the *top-level action* variable, since it corresponds to the root node of the parse tree for a natural language command. Our approach uses the supervision obtained from generating the correct top-level action to generate the gradient for all factors in the grounding graph. Each environmental context defines a low-level state/action space for the robot. A state consists of the robot's location and the location of all known objects in the environment. An action consists of moving to one of a few predefined locations (for example, near each object in the environment), or picking up manipulable objects in the environment. These actions are the space of possible values for the variable γ_a^d.

Next, we define a decision problem for each example d in the corpus. Each choice in the decision problem consists of binding a value to all grounding variables γ_i in the grounding graph. For instance, for the command, "Go to the tire pallet," the grounding variables would consist of γ_{pallet} (mapping to the phrase "the tire pallet") and γ_{go} (mapping to "Go to the tire pallet.") Actions in the decision problem consist of binding an object to γ_{pallet} and an action taken by the robot in the physical world to γ_{go}. For example, in an environment with a tire pallet (pallet 1) and a box pallet (pallet 2), the decision problem would contain the following actions:

action 1 γ_{go} = [Drive to location 1], γ_{pallet} = [pallet 1]
action 2 γ_{go} = [Drive to location 1], γ_{pallet} = [pallet 2]
action 3 γ_{go} = [Drive to location 2], γ_{pallet} = [pallet 1]
action 4 γ_{go} = [Drive to location 2], γ_{pallet} = [pallet 2]
action 5 γ_{go} = [Pick up pallet 1], γ_{pallet} = [pallet 1]
action 6 γ_{go} = [Pick up pallet 1], γ_{pallet} = [pallet 2]
action 7 γ_{go} = [Pick up pallet 2], γ_{pallet} = [pallet 1]
action 8 γ_{go} = [Pick up pallet 2], γ_{pallet} = [pallet 2]

We define a reward function for our decision problem based on choosing the correct grounding g_a^d for the top-level action variable, $\gamma_a \in \Gamma^d$ for a training

example d:

$$r(\Gamma^d, g_a^d) = \begin{cases} 1 & \text{if } \gamma_a = g_a^d \\ -1 & \text{otherwise} \end{cases} \tag{12}$$

If location 1 is assumed to be close to pallet 1, then actions 1 and 2 would receive +1 reward, and the rest would receive −1 reward. However, if pallet 1 is a tire pallet, and pallet 2 is a box pallet, then only action 1 gets all variables correct. Conversely, actions 3, 5, and 7 obtain negative reward for choosing the wrong value for γ_{go}, even though the grounding for γ_{pallet} is correct.[3]

Our aim is to learn model parameters to jointly predict top-level actions, which are directly rewarded, as well as values for other grounding variables, even though no reward signal is observed directly for these variables. Our hypothesis is that learning to predict other grounding variables is necessary in order to choose correct actions. For example, consider the command in the example given above, "Go to the tire pallet." If the robot grounds the wrong object, such as a box pallet, to the phrase "the tire pallet," then it will compute features associated with the wrong landmark object, leading to predicting incorrect actions. For a robot to predict correct actions, it must model the factored structure of language and propagate the error signal to each term. Over many examples the system uses this signal to infer values for the lower-level grounding variables which enables it to infer a policy for following natural language commands.

We define a sampling distribution to choose values for the grounding variables in the model using the G^3 framework with parameters Θ:

$$p(\Gamma^d | \Phi^d, \Lambda^d, M^d, \Theta) \tag{13}$$

Here the values of Λ^d are known, and the values of the correspondence variable vector Φ are fixed to $True$. Our aim is to find model parameters that maximize expected reward when drawing values for Γ^d from $p(\Gamma^d | \Phi^d, \Lambda^d, M^d, \Theta)$:

$$\underset{\Theta}{\operatorname{argmax}} \sum_{d \in D} E_{p(\Gamma^d | \Phi^d, \Lambda^d, M^d, \Theta)} r(\Gamma^d, g_a^d) \tag{14}$$

Expanding the expectation we have:

$$\underset{\Theta}{\operatorname{argmax}} \sum_{d} \sum_{\Gamma^d} r(\Gamma^d, g_a^d) \times p(\Gamma^d | \Phi^d, \Lambda^d, M^d, \Theta) \tag{15}$$

Here Γ^d ranges over all possible values for the grounding variables $\gamma_1 \ldots \gamma_N$, given the environmental context, M^d.

We can factor this distribution as in the G^3 framework described in Section 2. We use stochastic gradient ascent in the space of model parameters [Branavan et al., 2009] to maximize expected reward. We assume the

[3] We use a +1/-1 reward function rather than a 0-1 loss function because the gradient update will be scaled by the magnitude of the reward function, so a 0-1 loss function will perform no gradient update when the model infers the correct action.

Input:
Initial values for parameters, Θ^0.
Training dataset, D.
Number of iterations, T.
Step size, α.

$\Theta \leftarrow \Theta^0$
for $t \in T$ **do**
  for $d \in D$ **do**
    $\nabla_\Theta \leftarrow \frac{\partial}{\partial \theta_k} E_{p(\Gamma^d|\Phi^d,\Lambda^d,M^d,\Theta)} r(\Gamma^d, g_a^d)$
  end for
  $\Theta \leftarrow \Theta + \alpha \nabla_\Theta$
end for
Output: Estimate of parameters Θ

Fig. 3: Training algorithm.

model takes a factored form following the G^3 framework, based on the parse structure of a natural language command. This factorization enables the computation of gradient updates to the terms corresponding to each linguistic constituent, using only the top-level reward signal: whether the model infers the correct action for the command.

First, we take the derivative of the expected reward for a single example with respect to Θ. (We drop the d subscripts for brevity.)

$$\frac{\partial}{\partial \theta_k} E_{p(\Gamma|\Phi,\Lambda,M,\Theta)} r(\Gamma, g_a) = \sum_{\Gamma} r(\Gamma, g_a) \frac{\partial}{\partial \theta_k} p(\Gamma|\Phi, \Lambda, M, \Theta) \tag{16}$$

The derivation for the derivative appears in Section 8, yielding:

$$= E_{p(\Gamma|\Lambda,\Phi,M,\Theta)} r(\Gamma, g_a) \times \left(\sum_j f_k(\phi_j, \Gamma, \Lambda, M) - E_{p(\phi'|\Gamma,\Lambda,M,\Theta)} \left[f_k(\phi', \Gamma, \Lambda, M) \right] \right) \tag{17}$$

The training algorithm is given in Figure 3. Equation 17 is exact, but due to the large space of possible groundings, we approximate the expectation over Γ by sampling values for the Γ using current model parameters. In complex environments, there are many more actions that obtain negative reward than positive reward. To compensate for this, we normalize negative reward so that the total reward from actions with negative reward equals the sum of reward from actions with positive reward.

When performing gradient updates, we use 40 random restarts with initial values for Θ drawn uniformly from $[0, 0.00001]$ and take the parameters from the run that earns the most reward on the training set. We use 0.001 for the step size parameter α. The values correspond to the magnitude of the gradient update made for each iteration and affect the speed of convergence. We set them to small values so that the algorithm converges slowly but does not overstep an optimum. Our objective function is discontinuous and is not of

a form known to converge using gradient descent. However, gradient descent in objective functions with similar form have been shown to work well in practice [Branavan et al., 2009, Vogel and Jurafsky, 2010].

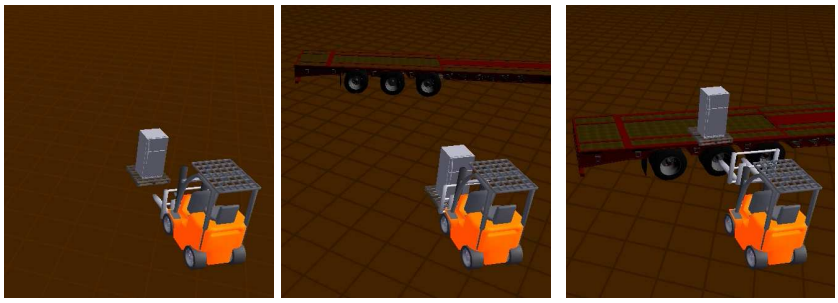
4 Results

To evaluate the learning algorithm we use a corpus of natural language commands given to a robotic forklift. We collected a corpus of natural language commands paired with robot actions by showing annotators on Amazon Mechanical Turk a video of the robot executing an action in simulation and asking them to describe in words they would use to command an expert human operator to carry out the commands in the video. We refer to each video as a scenario; we collected multiple commands for each scenario. Frames from a video in our corpus, together with commands for that scenario appear in Figure 4.

Since commands often spanned multiple parts of the video, we segment each log according to where each command begins and ends. We use 214 example commands paired with contextual information in the training set, and 93 in the test set, split randomly by scenario. That is, no scenarios in the training set appear in the test set, although scenarios consist of similar environments and the robot takes similar actions: driving around the environment and moving objects around. Each command has on average 11 words and the total vocabulary was 266 words. Commands were given in environments that contained anywhere from two to six objects, with an average of four objects in each environment. We annotated each command in the dataset with a parse, as well as associated groundings. For each linguistic constituent in the parsed command, we annotated the associated grounding in the environment.

We report two measures of performance. First, we report the percentage of commands for which the robot executed the command correctly, as determined by our annotations. This metric corresponds to how well the system follows commands but does not assesses how accurately it grounded different parts of a command. Second, we report the fraction of concrete noun phrases in the corpus that the system grounded to the correct object in the external world. That is, for each noun phrases in the ground truth parses, we report whether the system correctly inferred a grounding for each concrete noun phrase in the command. This metric corresponds to how well the system learned to recognize the mapping between noun phrases in the command and corresponding objects in the external world, even without direct supervision during training.

For comparison, we present several baselines. The random baseline shows performance achieved when choosing groundings using the random model parameters which were used to initialize the gradient descent algorithm. Second, we present the performance of a fully supervised model trained using an aligned corpus of positive and negative examples. Each linguistic constituent in the supervised dataset is paired with an annotated grounding, together with an indicator variable that encodes whether the annotation matches the words



(a) t=0 (b) t=20 (c) t=30

Pick up pallet with refridgerator [*sic*] and place on truck to the left.

A distance away you should see a rectangular box. Approach it slowly and load it up onto your forklift. Slowly proceed to back out and then make a sharp turn and approach the truck. Raise your forklift and drop the rectangular box on the back of the truck.

Go to the pallet with the refrigerator on it and pick it up. Move the pallet to the truck trailer. Place the pallet on the trailer.

Pick up the pallet with the refrigerator and place it on the trailer.

(d) Commands

Fig. 4: Frames from a video in our dataset, paired with natural language commands.

in the language, as in the original G^3 framework [Tellex et al., 2011]. This model is trained using the same features and parameters as the model trained from unaligned data. Finally, we present performance of a model trained using randomly assigned groundings. This baseline assesses the importance of performing gradient updates for each linguistic constituent in an example in a unified way rather than separately.

We also present results on a system trained using fully automatic parses, as well as trained using ground-truth parses. However, we use the annotated parses and groundings for assessing system performance to avoid having to annotate groundings for linguistic constituents generated from incorrect parses. Using the same parse structure also aids in comparing algorithms, since all approaches use the same parse structure when presenting evaluation results.

Table 1a shows performance using these two metrics on a training set. Note that the unaligned training algorithm significantly improves performance on the training set compared to starting performance. This improvement is not surprising for predicting actions, because the algorithm receives direct supervision for the action. However, the algorithm is also able to correctly infer groundings for many noun phrases in the training set, despite not having access to this information during training. The random groundings baseline also achieves good performance on the test set at predicting the top-level action, but does not correctly predict as many noun phrases.

Next, we report performance on a test set to assess generalization to novel commands given in different environments, in Table 1b. The trained system

	Actions	% Correct Concrete Noun Phrases
Random Model Parameters	35%	39%
Random Groundings	68%	46%
Unaligned	65%	65%
Unaligned (automatic parsing)	49%	61%
Fully Supervised	55%	79%

(a) Training (all) — 214 actions, 183 noun phrases

	Actions	% Correct Concrete Noun Phrases
Random Model Parameters	27%	37%
Random Groundings	56%	38%
Unaligned	57%	66%
Unaligned (automatic parsing)	48%	54%
Fully Supervised	56%	79%

(b) Testing (all) — 93 actions, 71 noun phrases

	% Correct Concrete Noun Phrases
Random Model Parameters	49%
Random Groundings	28%
Unaligned	67%
Unaligned (automatic parsing)	75%
Fully Supervised	84%

(c) Testing (Noun phrases only)

Table 1: Results on the training set and test set.

is able to achieve high performance at both inferring correct actions as well as correct object groundings using the parameters learned from the training set. It achieves this performance despite never observing groundings for any linguistic constituent during training other than the top-level verb phrase. Unlike the random groundings method, it also achieves high performance at predicting groundings noun phrase groundings.

The random groundings method achieves high performance at predicting the top level action but performs worse at predicting objects. The performance it does achieve at predicting object groundings is due to its ability to pick top-level actions associated with the natural language commands. It predicts top-level actions by learning a model for the vector associated with the verb phrase and its immediate arguments. For example, for an action like "pick up," the random groundings method will be biased towards actions that involve picking up an object that is bound to the linguistic argument in the noun phrases. This bias from the action causes it to correctly infer some object groundings, but the learned model parameters for noun phrases are unable to infer correct groundings in isolation, without an associated verb phrase. To test this hypothesis, we inferred groundings for noun phrases in the test set

only, without using information from the top level action. Results appear in Table 1c. Here the random groundings method performs very poorly, at chance levels, compared to the unaligned method. This result demonstrates that the unaligned method is not only learning models for predicting actions but is also able to learn word meanings for noun phrases. These learned meanings can be used to make predictions in a test set. We hypothesize that models for understanding each part of the language will lead to higher performance in more complex domains. In additions, we have demonstrated how they support information-theoretic dialog: by estimating the entropy over the groundings for each constituent, the robot can intelligently pick questions whose answers improves its accuracy at choosing actions in response to a natural language command [Tellex et al., 2012]. This approach would not work without models for grounding each constituent.

The fully supervised method performs surprisingly poorly on this dataset. We believe this poor performance is due to lack of sufficient annotated training data. Although we hypothesize that the fully-supervised approach would outperform our method given enough data, creating enough annotations is an expensive task. To generate the supervised training data, we sampled actions from the decision problem defined in Section 3 and annotated each factor in the associated grounding graph with the correct value for ϕ, depending on whether or not the sampled grounding was correct ($\phi = True$) or incorrect ($\phi = False$). This approach yielded good models for noun phrase groundings, as evidenced by the supervised method's high performance in Table 1c. However, because there are many more actions, the supervised learner did not yield good performance at predicting the top-level action. Given the very many possible negative instances, it was very difficult to provide enough labeled training instances to learn well. This is further, if somewhat anecdotal, evidence to support the unsupervised learning approach. In contrast, the supervised method outperforms the unaligned method when focusing on a verb which appears frequently in the corpus in a simple syntactic environment: "pick up." (83% for the supervised method, versus 78% for the unaligned method).

Finally, to emphasize that the system is learning grounded, functional word meanings, we present a visualization of the probability distribution learned for the word "to" in Figure 5. Each pixel z is colored according to $p(\phi = True|\lambda = \text{“to”}, \gamma_z)$, where the path γ_z starts at the left side of the image and ends at the location of pixel z. Red is high probability and blue is low probability. Note that the color scale is normalized within each image, but not normalized across images. Artifacts are due to a limited amount of training data and the discretization of continuous feature values (for example, the normalized distance between the end of the trajectory and the tire pallet.) In both cases, there is a high-probability peak near the landmark object. These heat maps emphasize that the algorithm has learned distributions corresponding to the meanings of words such as "to" and "pick up." It has not simply learned a mapping between a word such as "to" and a symbol such as TO. Instead it has a learned a distribution that can be called with *any* candidate grounding

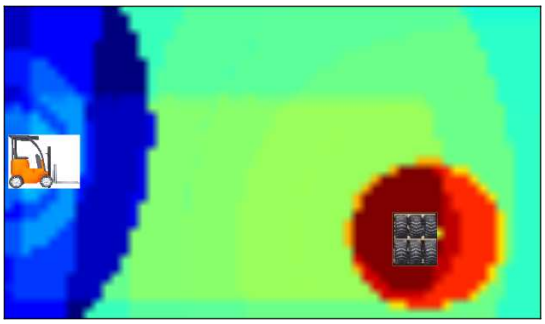


Fig. 5: Heat map showing the learned distribution for "to the tire pallet." probability.

and returns the probability that the words in the language match a path in the world.

Failures occurred in the system for several reasons. Sometimes, the command requested an action that is not in the action space of the robot. For example, the command "Move slightly forward" requires a very small motion on the part of the robot, while "move forward in an 'S' shape" requires complex motion planning. These actions are not in the search space of possible actions, so it is not possible for the robot to behave correctly. Other commands used less frequent language. For example, one person instructed the robot to "loiter around momentarily;" the word "loiter" appeared only in the test set and not in the training set. Finally, some commands contained ambiguous language, such as "Pick it up," where the system inferred that it should pick up an object, but acted upon the wrong object.

5 Related Work

Beginning with SHRDLU [Winograd, 1971], many systems have exploited the compositional structure of language to statically generate a plan corresponding to a natural language command [Hsiao et al., 2008, MacMahon et al., 2006, Skubic et al., 2004, Dzifcak et al., 2009]. Our work moves beyond this framework by defining a probabilistic graphical model according to the structure of the natural language command, inducing a distribution over plans and groundings.

Semantic parsing is the problem of mapping between sentences in natural language and formal semantic representations. Early work in semantic parsing uses supervised data consisting of sentences paired with logical form [Thompson and Mooney, 2003, Zettlemoyer and Collins, 2005, Wong and Mooney, 2007, Piantadosi et al., 2008, Kwiatkowski et al., 2010]; some of these approaches have been applied to robotics [Matuszek et al., 2012b, Chen and Mooney, 2011]. Newer approaches learn symbolic word meanings with less supervision; Poon and Domingos [2009] presents an approach to unsupervised

semantic parsing, while other approaches use an external reward signal as in this paper [Branavan et al., 2009, Vogel and Jurafsky, 2010, Liang et al., 2011, Clarke et al., 2010]. The current work, in contrast, learns grounded meaning representations in terms of perceptual features rather than symbols. Previous models that learned grounded word meanings [Tellex et al., 2011, Kollar et al., 2010] required detailed alignment annotations between constituents in the language and objects, places, paths, or events in the external world; the current work relaxes these annotation requirements. Matuszek et al. [2012a] describe a model for jointly learning word meanings and perceptual features for referring expressions, rather than both noun and verb phrases as in this work.

There has been a variety of work in transferring action policies between a human and a robot. In imitation learning, the goal is to create a system that can watch a teacher perform an action, and then reproduce that action [Kruger et al., 2007, Chernova and Veloso, 2009, Schaal et al., 2003, Ekvall and Kragic, 2008]. Rybski et al. [2007] developed an imitation learning system that learns from a combination of imitation of the human teacher, as well as natural language input. Our work differs in that the system must infer word meanings from the demonstrations, and then use those learned meanings later to infer new actions for different commands in different contexts

6 Conclusion

In this paper we described an approach for learning perceptually grounded word meanings from an unaligned parallel corpus of language paired with robot actions. The training algorithm jointly infers policies that correspond to natural language commands as well as alignments between noun phrases in the command and groundings in the external world. In addition, our approach learns grounded word meanings or distributions corresponding to words in the language. We presented an evaluation on a small corpus, demonstrating that the system is able to infer meanings for concrete noun phrases despite having no direct supervision for these values.

Our approach points the way towards a framework that can learn a large vocabulary of general grounded word meanings, enabling systems that flexibly respond to a wide variety of natural language commands given by untrained users. There are many directions for improvement. We plan to train our system using a large dataset of language paired with robot actions in more complex environments, and on more than one robotic platform. Identifying a set of perceptual features that generalizes across these different contexts remains an open problem. As long as perceptual features are available, the system is able to learn meanings for adjectives and prepositions. For example, the training corpus contains phrases such as “red and black pallet,” and the system learns corresponding feature weights for the words “red” and “black.” However as the number of features increases, the learning problem becomes more challenging, and if features are unavailable, the system will fail to learn word meanings. For

example, phrases such as "left" and "right" are challenging because features need to take into account the frame-of-reference being used by the speaker. Integrating our approach with a richer symbolic semantics could enable it to handle more complex linguistic phenomena, including quantifiers, negation, and determiners.

7 Acknowledgments

We would like to thank the anonymous reviewers for their insightful comments, which significantly shaped the paper. We would also like to thank Thomas Howard for his helpful comments on a draft of this paper. This work was sponsored by the Robotics Consortium of the U.S Army Research Laboratory under the Collaborative Technology Alliance Program, Cooperative Agreement W911NF-10-2-0016, and by the Office of Naval Research under MURIs N00014-07-1-0749 and MURI N00014-11-1-0688, and the DARPA BOLT program under contract HR0011-11-2-0008.

References

S. R. K. Branavan, H. Chen, L. S. Zettlemoyer, and R. Barzilay. Reinforcement learning for mapping instructions to actions. In *Proceedings of ACL*, page 82–90, 2009.

D. L. Chen and R. J. Mooney. Learning to interpret natural language navigation instructions from observations. In *Proc. AAAI*, 2011.

S. Chernova and M. Veloso. Interactive policy learning through confidence-based autonomy. *JAIR*, 34(1):1–25, 2009.

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

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

J. Dzifcak, M. Scheutz, C. Baral, and P. Schermerhorn. What to do and how to do it: Translating natural language directives into temporal and dynamic logic representation for goal management and action execution. In *Proc. IEEE Int'l Conf. on Robotics and Automation (ICRA)*, pages 4163–4168, 2009.

S. Ekvall and D. Kragic. Robot learning from demonstration: a task-level planning approach. *International Journal of Advanced Robotic Systems*, 5 (3), 2008.

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

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

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

V. Kruger, D. Kragic, A. Ude, and C. Geib. The meaning of action: A review on action recognition and mapping. *Advanced Robotics*, 21(13), 2007.

T. Kwiatkowski, L. Zettlemoyer, S. Goldwater, and M. Steedman. Inducing probabilistic ccg grammars from logical form with higher-order unification. In *Proceedings of the 2010 conference on empirical methods in natural language processing*, pages 1223–1233. Association for Computational Linguistics, 2010.

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

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

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

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

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

N. Mavridis and D. Roy. Grounded situation models for robots: Where words and percepts meet. In *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 4690–4697. IEEE, Oct. 2006. ISBN 1-4244-0258-1.

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

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

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

P. Rybski, K. Yoon, J. Stolarz, and M. Veloso. Interactive robot task training through dialog and demonstration. In *Proceedings of HRI*, page 56. ACM, 2007.

S. Schaal, A. Ijspeert, and A. Billard. Computational approaches to motor learning by imitation. *Phil. Trans. R. Soc. Lond. B*, (358), 2003.

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

Systems, Man, and Cybernetics, Part C: Applications and Reviews, 34(2): 154–167, 2004. ISSN 1094-6977.

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 *Proc. AAAI*, 2011.

S. Tellex, P. Thaker, R. Deits, T. Kollar, and N. Roy. Toward information theoretic human-robot dialog. In *Proceedings of Robotics: Science and Systems*, Sydney, Australia, July 2012.

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

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

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

Y. Wong and R. Mooney. Learning synchronous grammars for semantic parsing with lambda calculus. In *Association for Computational Linguistics*, volume 45, page 960, 2007.

L. S. Zettlemoyer and M. Collins. Learning to map sentences to logical form: Structured classification with probabilistic categorial grammars. In *UAI*, pages 658–666, 2005.

8 Appendix

We take the derivative of the expected reward for a single example with respect to Θ. (We drop the d superscripts for brevity.)

$$\frac{\partial}{\partial \theta_k} E_{p(\Gamma|\Phi,\Lambda,M,\Theta)} r(\Gamma, g_a) = \sum_{\Gamma} r(\Gamma, g_a) \frac{\partial}{\partial \theta_k} p(\Gamma|\Phi, \Lambda, M, \Theta) \tag{18}$$

Focusing on the inner term, we expand it with Bayes' rule:

$$\frac{\partial}{\partial \theta_k} p(\Gamma|\Phi, \Lambda, M, \Theta) = \frac{\partial}{\partial \theta_k} \frac{p(\Phi|\Gamma, \Lambda, M, \Theta) p(\Gamma|\Lambda, M, \Theta)}{p(\Phi|\Lambda, M, \Theta)} \tag{19}$$

We assume the priors do not depend on Θ, as well as a uniform prior over groundings and correspondence variables:

$$\frac{\partial}{\partial \theta_k} p(\Gamma|\Phi, \Lambda, M, \Theta) = \frac{p(\Gamma|M)}{p(\Phi|\Lambda)} \frac{\partial}{\partial \theta_k} p(\Phi|\Gamma, \Lambda, M, \Theta) \tag{20}$$

Previously, the G^3 framework required alignment annotations or observed values for the grounding variables to update model parameters. Here, we update model parameters based on the top-level reward and propagate that signal

down to factors for each linguistic constituent. Although updates for any individual sample will be noisy, over many updates, the algorithm will converge to a local optimum.

Next, we take the partial derivative of the likelihood of Φ. For brevity, we compress Γ, Λ, and M in the variable X. First, we assume each factor is independent:

$$\frac{\partial}{\partial \theta_k} p(\Phi|X,\Theta) = \frac{\partial}{\partial \theta_k} \prod_i p(\phi_i|X,\Theta) \tag{21}$$

Next, we expand the right side of Equation 21, taking the derivative using the product rule:

$$\frac{\partial}{\partial \theta_k} p(\Phi|X,\Theta) = \prod_i p(\phi_i|X,\Theta) \times \left(\sum_j \frac{\frac{\partial}{\partial \theta_k} p(\phi_j|X,\Theta)}{p(\phi_j|X,\Theta)} \right) \tag{22}$$

Finally, we assume each factor takes a log-linear form with feature functions f_k and parameters θ_k, as in the G^3 framework, Equation 10. Taking the derivative of a log-linear distribution yields:

$$\frac{\partial}{\partial \theta_k} p(\phi|X,\Theta) = p(\phi|X,\Theta) \times \left(f_k(\phi, X) - E_{p(\phi'|X,\Theta)} \left[f_k(\phi', X) \right] \right) \tag{23}$$

Substituting back into Equation 20:

$$\frac{\partial}{\partial \theta_k} p(\Gamma|\Phi,\Lambda,M,\Theta) = \frac{p(\Gamma|M)}{p(\Phi|\Lambda)} \prod_i \left(p(\phi_i|X,\Theta) \times \sum_j f_k(\phi_j, X) - E_{p(\phi'_j|X,\Theta)} \left[f_k(\phi'_j, X) \right] \right) \tag{24}$$

We substitute back into the overall expression for the partial derivative of the expectation:

$$\frac{\partial}{\partial \theta_k} E_{p(\Gamma|\Phi,\Lambda,M,\Theta)} r(\Gamma, g_a) = E_{p(\Gamma|\Lambda,\Phi,M,\Theta)} r(\Gamma, g_a) \times \left(\sum_j f_k(\phi_j, \Gamma, \Lambda, M) - E_{p(\phi'|\Gamma,\Lambda,M,\Theta)} \left[f_k(\phi', \Gamma, \Lambda, M) \right] \right) \tag{25}$$