2014 IEEE International Conference on Robotics & Automation (ICRA)
Hong Kong Convention and Exhibition Center
May 31 - June 7, 2014. Hong Kong, China

A Natural Language Planner Interface for Mobile Manipulators

Thomas M. Howard[1], Stefanie Tellex[2] and Nicholas Roy[1]

***Abstract*— Natural language interfaces for robot control aspire to find the best sequence of actions that reflect the behavior intended by the instruction. This is difficult because of the diversity of language, variety of environments, and heterogeneity of tasks. Previous work has demonstrated that probabilistic graphical models constructed from the parse structure of natural language can be used to identify motions that most closely resemble verb phrases. Such approaches however quickly succumb to computational bottlenecks imposed by construction and search the space of possible actions. Planning constraints, which define goal regions and separate the admissible and inadmissible states in an environment model, provide an interesting alternative to represent the meaning of verb phrases. In this paper we present a new model called the Distributed Correspondence Graph (DCG) to infer the most likely set of planning constraints from natural language instructions. A trajectory planner then uses these planning constraints to find a sequence of actions that resemble the instruction. Separating the problem of identifying the action encoded by the language into individual steps of planning constraint inference and motion planning enables us to avoid computational costs associated with generation and evaluation of many trajectories. We present experimental results from comparative experiments that demonstrate improvements in efficiency in natural language understanding without loss of accuracy.**

I. INTRODUCTION

Advances in human-robot interaction have improved our ability to communicate with robots. Though progress has been made, contemporary approaches are often tailored for specific platforms and domains. Robots still require detailed, low-level guidance in the form of commands in a restrictive language to perform non-trivial tasks. Although more recent work based on probabilistic graphical models has pushed towards connecting natural language to robot actions [1], grounding language in this manner is quite difficult in practice as even coarse approximations of the state-action space can be computationally prohibitive for inferring how a robot should respond to an instruction.

Rather than inferring a sequence of actions directly from the instruction, a more natural approach may be to infer a representation of the task that permits the robot to independently compute the desirable sequence of actions. It is then the responsibility of the individual platforms to determine how best to articulate its degrees of freedom to satisfy the objectives of the inferred task in the current environment model. Planning constraints are a logical choice of representation, because they partition the state-space into admissible, inadmissible, and goal regions. If we can infer how the shape of the admissible and goal regions varies over the course of the tasks, we can apply trajectory planning algorithms to solve for the sequence of actions that navigate through this space. An example of our approach is illustrated in Figure 1 where a trajectory and the admissible and goals regions are shown for the command "move near the red box and the blue crate". We could try to identify the best state-action sequence directly from the instructions, but the number of state-action sequences to be considered quickly becomes intractable with the complexity of the robot and the utterance. If instead we identify the admissible region of states and the goal region, we can use one of a variety of motion planning algorithms to generate the desired motion.

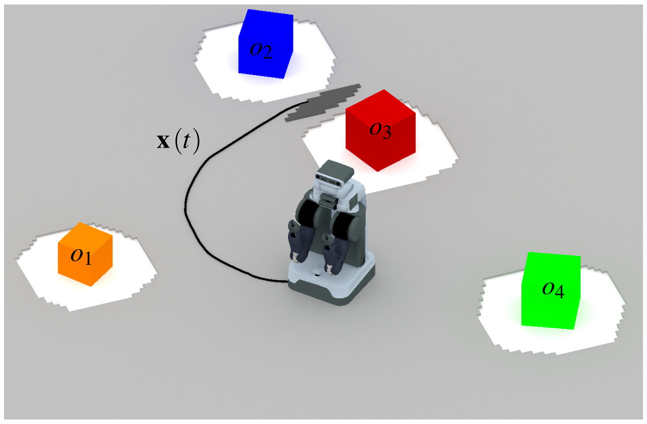


Fig. 1. An illustration of the robot trajectory $\mathbf{x}(t)$ generated from planning constraints that were inferred from the natural language instruction "move near the red box and the blue crate" using the Distributed Correspondence Graph (DCG) model. The dark gray, light gray, and white regions represent the goal states, admissible states, and inadmissible states respectively. The variables $o_1 \ldots o_4$ identify the four objects in the environment model.

We present a model, called the Distributed Correspondence Graph (DCG), that can be used to efficiently infer the most likely set of planning constraints given the natural language instruction and environment. The main advantages of this model in the context of planning constraint inference are improvements in efficiency and generality. The DCG model assumes conditional independence of the primitives that compose a phrase grounding to reduce the computational complexity of probabilistic inference. Since the space of trajectories is no longer required at inference time, we do not need to ground linguistic constituents in the infinitely large, continuous space of robot actions. Generality is improved because the shape of the admissible regions is a function of the constraints, the robot, and the environment model.

[1]T.M. Howard and N. Roy are with the Computer Science and Artificial Intelligence Laboratory, Massachusetts Institute of Technology, Cambridge, MA 02139, USA

[2]S. Tellex is with the Computer Science Department at Brown University, Providence, RI, 02912, USA

II. NATURAL LANGUAGE UNDERSTANDING OF ROBOT INSTRUCTIONS

The overall goal of natural language interfaces for robot control is to find the trajectory ($\mathbf{x}(t) = \left[\mathbf{x}(t_i) \ldots \mathbf{x}\left(t_f\right)\right]$) that best performs the activities described by an instruction (Λ) in the context of the world model (Υ):

$$\underset{\mathbf{x}(t) \in \Re^K}{\arg\max}\, p\left(\mathbf{x}(t) \mid \Lambda, \Upsilon\right) \tag{1}$$

In practice, Equation 1 is difficult to compute because of the diversity of language, environments, and trajectories. The Generalized Grounding Graph (G^3) model [1] is a contemporary technique for generating robot actions from natural language instructions. This model is a factor graph that is trained from a corpus of labeled examples to ground language with objects, locations, and paths. Grounding is the process of assigning physical meanings to natural language. For example, it is common to associate nouns with objects, prepositions with locations, and verbs with actions. The structure of the factor graph follows the parse tree of the instruction by connecting phrases ($\Lambda = [\lambda_1 \ldots \lambda_n]$) to groundings ($\Gamma = [\gamma_1 \ldots \gamma_n]$) and correspondence variables ($\Phi = [\phi_1 \ldots \phi_n]$). In this model the random variables that represent phrases and correspondence variables are known and the groundings are unknown. To find the values of the grounding variables that most closely resemble the command, we search for the assignments of Γ that maximize Equation 2.

$$\underset{\gamma_1 \cdots \gamma_n}{\arg\max}\, p\left(\Phi \mid \gamma_1 \ldots \gamma_n, \Lambda, \Upsilon\right) \tag{2}$$

The G^3 model assumes that the groundings for linguistic constituents are conditionally independent. This permits the probability of the groundings and the language to be factored across individual phrases. This is shown in Equation 3, where Γ_{c_i} is the array of the groundings from phrases that are children of the current phrase (λ_i).

$$\underset{\gamma_1 \cdots \gamma_n}{\arg\max} \prod_i p\left(\phi_i \mid \gamma_i, \lambda_i, \Gamma_{c_i}, \Upsilon\right) \tag{3}$$

An example parse tree and grounding graph for the command "move near the red box and the blue crate" is shown in Figures 2 and 3. In Figure 2 the instruction is broken down into phrases, parts of speech, and words. The sentence has three noun phrases ("and", "the red box", and "the blue crate"), one verb phrase ("move"), and one prepositional phrase ("near"). While it is straightforward to associates individual object with "the red box" and "the blue crate", the grounding for "and" requires an association with both of these objects. This means that the space of groundings must be the power set of objects in the world model to correctly ground noun phrases, leading to an exponential number of locations and paths that must be evaluated by both prepositional and verb phrases.

We observe two shortcomings with the solution proposed by the G^3 model. First, it is difficult to efficiently approximate the space of possible motions for non-trivial robot systems. Second, for environments where we must infer

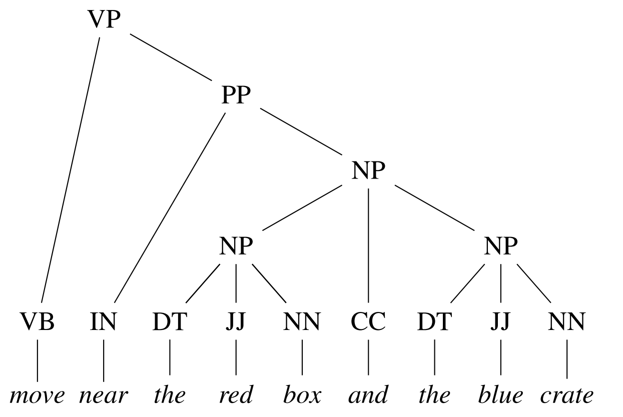


Fig. 2. A parse tree for the sentence "move near the red box and the blue crate". Part-of-speech tags in the parse tree are from the Penn Treebank [2].

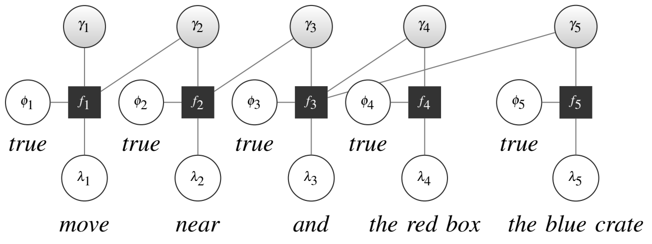


Fig. 3. The factor graph resulting from the parse tree in Figure 2, used by the G^3 algorithm to infer the groundings of the instructions. Each linguistic is grounded to a object, location, or action through a factor that incorporates the grounding of its children. Black boxes, white spheres, and gray sphere are factors, known random variables, and unknown random variables respectively.

sets of objects, the number of groundings that we must evaluate grows exponentially. This means that the algorithm can quickly become intractable in environments with even a relatively small number of objects. These observations provoke the question of whether paths and locations are the best candidates for grounding verbs and prepositions. In many scenarios there are numerous samples from the continuum of paths and locations that may equally correspond to phrase and child groundings. A far more efficient method would search across sets of equally probable paths or locations instead of individual samples. If we instead search for the boundaries and preferences of robot motion we could use natural language instructions to formulate robot planning problems, rather than try to infer solutions directly from the continuum of robot trajectories.

III. PLANNING CONSTRAINT INFERENCE

We begin the description of a natural language planner interface with a formal description of the problem that we will solve. We formulate the robot planning problem generally as constrained optimization:

$$\text{minimize}: \quad \Phi\left(\mathbf{x}\left(t_f\right)\right) + \int_{t_i}^{t_f} \mathsf{L}\left(\mathbf{x}(t), \mathbf{u}(t), t\right) dt \tag{4}$$

$$\text{subject to}: \quad \dot{\mathbf{x}}(t) = \mathbf{f}_{\mathbf{PMM}}\left(\mathbf{x}(t), \mathbf{u}(t), t\right) \tag{5}$$

$$\mathbf{x}(t_i) = \mathbf{x_I} \tag{6}$$

$$\mathbf{c}\left(\mathbf{x}(t), \mathbf{u}(t), t\right) = \mathbf{0} \tag{7}$$

In this description of the robot planning problem, $\mathbf{x}(t)$ represents the state of the robot as a function of time. Equation 4 describes a cost function that outlines preferences for the behavior of the robot (e.g., minimum time, minimum energy). Equation 5 characterizes the state-space response to actions ($\mathbf{u}(t)$) in the form of a predictive motion model ($\mathbf{f_{PMM}}(\mathbf{x}(t),\mathbf{u}(t),t)$). Equation 6 defines the initial conditions of the world model. Equation 7 describes a set of constraints $\mathbf{c}(\mathbf{x}(t),\mathbf{u}(t),t)$ to define the admissible states of the environment. We define a constraint as a function over the robot state, input, and time. The constraint function implicitly specifies a subregion of the state space and a window of time; the function returns 0 if the configuration is inside the region during the time window and some non-zero value if the configuration is outside the region. The function also returns 0 for all configurations outside the time window.

$$\mathbf{c}(\mathbf{x}(t),\mathbf{u}(t),t) = \begin{bmatrix} c_1(\mathbf{x}(t),\mathbf{u}(t),t) \\ c_2(\mathbf{x}(t),\mathbf{u}(t),t) \\ \vdots \\ c_n(\mathbf{x}(t),\mathbf{u}(t),t) \end{bmatrix} \quad t \in (t_i, t_f] \tag{8}$$

$$c_i(\mathbf{x}(t),\mathbf{u}(t),t) = \begin{cases} g_i(\mathbf{x}(t),\mathbf{u}(t),t) & \text{if } t \in (t_j,t_k] \\ 0 & \text{otherwise} \end{cases} \tag{9}$$

The sequence of time windows that are defined by the constraints correspond to a sequence of admissible regions of the configuration space. Individual constraints are deemed active when t is within t_j and t_k and inactive when not. Constraint parameters t_j and t_k may vary for individual constraints, but the minimum t_j must equal t_i and the maximum t_k must equal t_f. Constraints in the constraint set may overlap and jointly filter states as the intersection of n time-varying configuration spaces. Inactive constraints do not influence the shape of the admissible region.

Our approach to natural language understanding of robot instructions replaces search in the continuum of actions with search for the set of constraints that most likely represents the utterance in the context of the perceived environment. We address the issue of the size of the space of groundings with two observations. First is that the space of planning constraints that can be understood by our planner is finite. The number of constraints is bounded by the quantity of objects in our environment and the number of relationships between objects that the planner can consider. Second is that the expression of many, if not all of the constituents of a grounding can be evaluated independently from each other. If we assume that verbs are grounded to constraint sets, constraints in a constraint set are conditionally independent, and that the constraints are known, we can factorize the model defined in Equation 3 across individual constituents. In this model we search for both the unknown correspondence variables ($\phi_{ij} \in \Phi$) and unknown groundings ($\gamma_i \in \Gamma$) to maximize the product of individual factors:

$$\underset{\gamma_i \in \Gamma, \phi_{ij} \in \Phi}{\arg\max} \prod p(\phi_i \mid \gamma_i, \lambda_i, \Gamma_{c_i}, \Upsilon) \prod p(\phi_{ij} \mid \gamma_{ij}, \lambda_i, \Gamma_{c_{ij}}, \Upsilon) \tag{10}$$

The variables γ_i and γ_{ij} are random variables from the set Γ that represent the groundings of i^{th} phrase and the j^{th} constituent of the grounding of the i^{th} phrase respectively. Likewise, the variables ϕ_i and ϕ_{ij} are random variables from the set Φ that represent the correspondence of the i^{th} grounding and the j^{th} constituent of the i^{th} grounding respectively. If we further assume that all phrases are conditionally independent and all groundings are composed of conditionally independent elements (e.g. constraint sets, object sets), we could further factorize the model defined in Equation 3 to form a model where only the correspondence variables are not known:

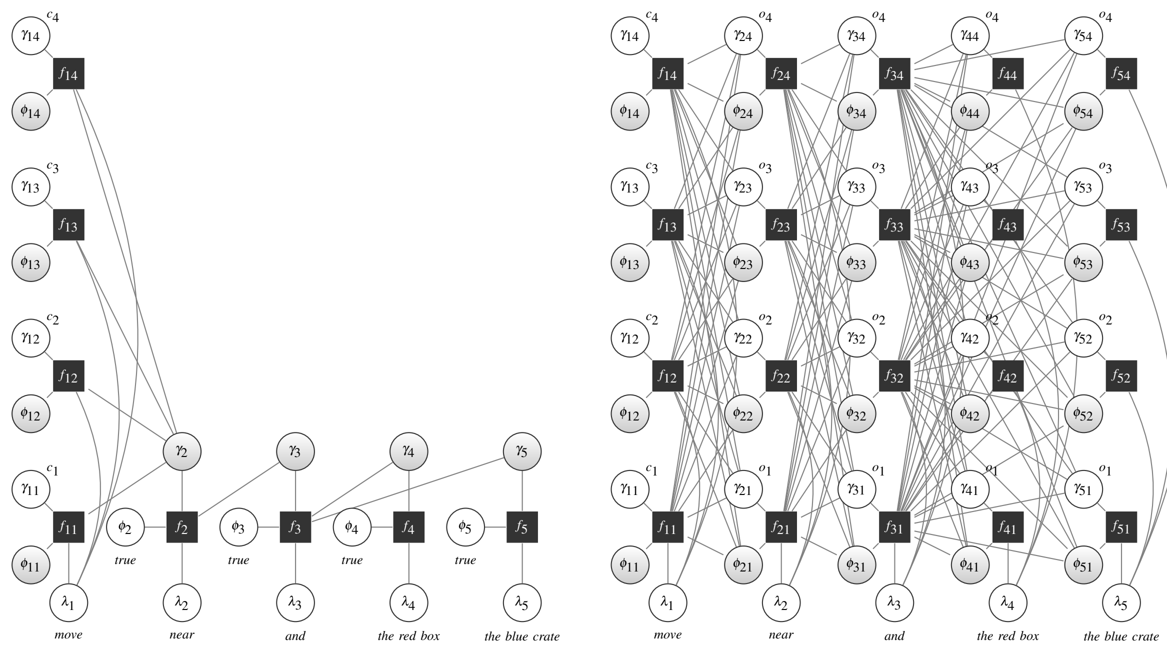
$$\underset{\phi_{ij} \in \Phi}{\arg\max} \prod p(\phi_{ij} | \gamma_{ij}, \lambda_i, \Gamma_{c_{ij}}, \Upsilon) \tag{11}$$

We use log-linear models with binary features as factors to approximate the probability of a correspondence variable given the groundings and language. Since the function expressed by a factor is guaranteed to be convex, we apply the Limited-memory Broyden-Fletcher-Goldfarb-Shanno (L-BFGS) algorithm [3] to optimize the binary feature weights μ from a corpus of labeled training examples.

$$\underset{\phi_{ij} \in \Phi}{\arg\max} \prod \psi(\phi_{ij}, \gamma_{ij}, \lambda_i, \Gamma_{c_{ij}}, \Upsilon) \tag{12}$$

$$\psi(\phi_{ij}, c_{ij}, \lambda_i, \Gamma_{c_{ij}}, \Upsilon) = \frac{e^{\sum_l \mu_l f_l \left(\phi_{ij}, \gamma_{ij}, \lambda_i, \Gamma_{c_{ij}}, \Upsilon\right)}}{\sum_q e^{\sum_l \mu_l f_l \left(\phi_q, \gamma_{ij}, \lambda_i, \Gamma_{c_{ij}}, \Upsilon\right)}} \tag{13}$$

For the model described in Equation 11 all of the random variables that represent the groundings are known. What is not known is how a constraint corresponds to the current phrase in the context of the child groundings. If we were to use the binary correspondence variable from the G^3 model it would allow us to evaluate whether it is more likely that a constituent should or should not be expressed in the grounding. A constraint however defines two disjoint regions and it is desirable to evaluate whether a region or its complement should be expressed. We therefore introduce a ternary correspondence variable for constraint groundings that allows us to evaluate whether a constraint should be active, inverted, or inactive. This reduces the likelihood of inferring constraints whose intersection is the empty set because a constraint and its complement could not both be actively expressed. It also facilitates inference by reducing the number of factor evaluations from four (a pair of factors with two evaluations each) to three (a single factor with three evaluations) for a single relationship. We call the model described in Equation 11 the Distributed Correspondence Graph (DCG) because it distributes conditionally independent elements of phrase groundings across multiple factors and we search for the correspondences for known constituents. We call the model in Equation 10 the Hybrid G^3-DCG model because it searches across both unknown groundings and unknown correspondence variables to ground the natural language expression. The Hybrid G^3-DCG and DCG models for the parse tree from Figure 2 are illustrated in Figure 4. In this example, there are four objects in the environment, four possible regions in the region set, and four possible constraints in the constraint set. In this example, we



(a) Hybrid G^3-DCG Model (b) DCG Model

Fig. 4. An illustration of the Hybrid G^3-DCG and DCG models that result from the parse tree in Figure 2. Black boxes, white spheres, and gray sphere are factors, known random variables, and unknown random variables respectively. The values of the known grounding constituents $c_1 \ldots c_4$ and $o_1 \ldots o_4$ represent four constraints from the constraint set and four objects from the environment model.

assume that there are four objects in the environment and four relationships that can be expressed in two ways, for a total of eight possible constraints in the constraint set.

On first inspection it appears that we have increased the difficulty of the learning problem by introducing more factors in the graphical model. While it is true that the graphical models in Figure 4 have more unknown random variables than the one shown in Figure 3, the space of values for the unknown random variables has been reduced significantly. The number of candidate regions that a state-action must fill increases exponentially with the number of equivalent constraints. This leads to exponential growth in the number of sums of weighted feature evaluations to maximize the likelihood of a truthful grounding, compared to linear growth of the sums of weighted feature evaluations required to maximize the likelihood of a correspondence between known phrases and groundings with an equivalent number of constraints. The minimum number of factor evaluations required to maximize the product of the factors that represent each phrase in the G^3, Hybrid G^3-DCG, and DCG models from Figures 3 and 4 are shown in Table I[1]. For the noun and prepositional phrases the DCG model requires only half of the factor evaluations as the G^3 and Hybrid G^3-DCG models since inclusion or exclusion from the object set that represents the grounding is evaluated independently. The difference is even more pronounced for the verb phrase in these examples because the DCG and Hybrid G^3-DCG models take advantage of four ternary correspondence variables to supplant search over the power set of eight constraints (four pairs of complementary constraints) in the G^3 model.

[1]We show the minimum number of factor evaluations because beam search could be applied with a non-unit beam width

TABLE I
MINIMUM NUMBER OF FACTOR EVALUATIONS FOR EACH PHRASE IN THE G^3, HYBRID G^3-DCG, AND DCG MODELS FOR "MOVE NEAR THE RED BOX AND THE BLUE CRATE"

	phrase				
	λ_1	λ_2	λ_3	λ_4	λ_5
G^3	256	16	16	16	16
Hybrid G^3-DCG	12	16	16	16	16
DCG	12	8	8	8	8

IV. IMPLEMENTATION

To apply planning constraint inference within the G^3 framework, we need the capacity to model an environment, convert natural language instructions into factor graphs, and generate trajectories from inferred planning constraints. We represent the environment as a collection of objects at a known time. Each object is represented a tree of rigid bodies connect via revolute, prismatic, or fixed joints. Rigid bodies in our model have geometric, visual, and inertial properties as well as a set of semantic tags. Environment dynamics are modeled using the Bullet physics engine to both initialize environment models and simulate the effect of actions in the trajectory planner. Sentences are converted into parse trees using either the Stanford Parser [4] or the Cocke-Kasami-Younger (CKY) algorithm [5] with a fixed grammar. We applied three different types of constraints (distance, orientation, and contact) at two time intervals between pairs of rigid bodies in the environment model. Humans assign natural language instructions to examples using four images that depict the initial and terminal states of a trajectory that satisfies the planning constraint set. Each labeled example is then annotated to provide examples of verb phrases grounded to constraint sets and noun phrases grounded to rigid bodies. This process is repeated many times in randomly organized environments to form a corpus. Trajectories are generated from constraint sets and a cost function that minimizes time of execution by searching a tree of actions sampled from a control library until the goal region is reached.

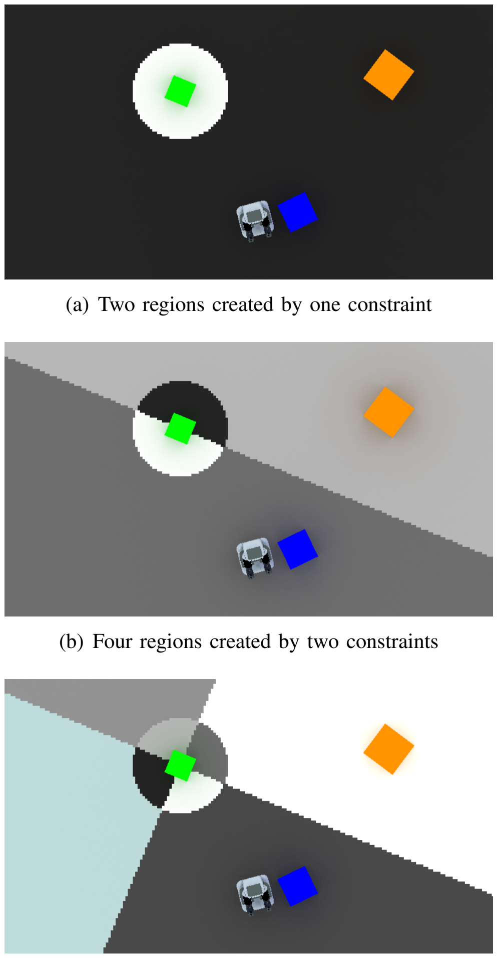
V. Experiments

We propose several experiments to evaluate the ability of our models to correctly and efficiently infer constraints from natural language. First, we compare the runtime of constraint inference in a Hybrid DCG-G^3 model against action inference in the G^3 model baseline. To fairly measure their comparative performance, the number of trajectories searched by the G^3 model will be equal to the number of non-overlapping regions of admissible states that can be expressed by the set of constraints in the Hybrid DCG-G^3 model. Next, we evaluate the accuracy of the G^3 and Hybrid DCG-G^3 models by training each from two corpora of labeled examples and measuring whether the correct action or constraint set is correctly inferred. We then evaluate the ability to infer constraints from sentences with conjunctions, transfer the learned model to platforms with similar physical characteristics, and generate trajectories from natural language in dynamic environments.

VI. Results

A. Runtime

Two factors that influence the runtime of natural language interfaces are the time that it takes to construct the model and the time that it takes to evaluate the model. In Section III we showed that the size of a state-action space grows exponentially with a linear increase in the space of planning constraints. To support this claim, we measure the time that it takes to compute the most likely correspondence variables for planning constraint sets of increasing size against the time that it takes to find the most probable trajectory from the space of trajectories that can be represented by the planning constraint set. Figure 5 shows these regions for one, two, and three constraints in an environment with one robot and three different colored boxes. The first constraint evaluates proximity between the robot base and the center of the green box. Expression and inversion of this constraint produces two distinct regions inside and outside of the decision boundary. A second constraint is added that partitions the space based on the relative position of the robot base to the x-axis of the coordinate frame of the green box, producing four distinct regions. A third constraint is added that further divides the space based on the relative position of the robot base to the y-axis of the coordinate frame of the green box, generating eight distinct sets of admissible states. These three


(a) Two regions created by one constraint

(b) Four regions created by two constraints

(c) Eight regions created by three constraints

Fig. 5. An illustration of the regions defined by the expression or inversion of one, two, and three constraints between the robot base and the green box.

constraints and eight regions could be used to understand prepositional phrases such as "near the green box", "in front of the green cube" or verb phrases like "approach the right of the smallest box" or "move away from the green item". Note that it is not guaranteed that there will be an admissible region for all combinations of constraints. The active expression of three proximity constraints between each of the three colored boxes and the robot base is one example of three constraints whose intersection would not yield any admissible states[2].

Figure 6 shows the runtime of planning constraint inference and trajectory inference for six planning constraint sets. The first three planning constraint sets are equivalent to those illustrated in Figure 5. The latter three constraints add proximity and half-plane constraints between the robot base and the blue box. Admissible terminal states of trajectories are found by randomly sampling in the space of robot and object positions and orientations. We see the expected linear and exponential increase in runtime for planning constraint inference and trajectory inference respectively. Inferring the

[2]If the trajectory planner is allowed to move boxes, an admissible solution can be found by relocating two of the boxes near the third.

correct expression of only six planning constraints takes approximately 88% less time than computing the most probable trajectory from a space of sixty-four trajectories, not taking into account the time that it takes to generate the state-action space. As more objects and relationships between objects are considered, the computational burden of evaluating the equivalent state-action space quickly becomes intractable.

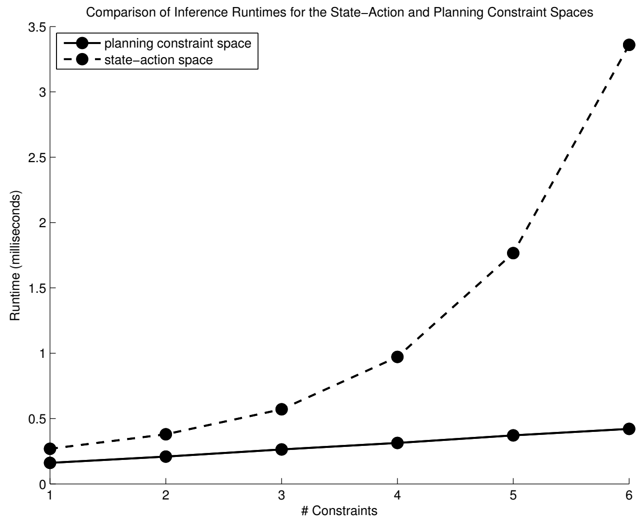
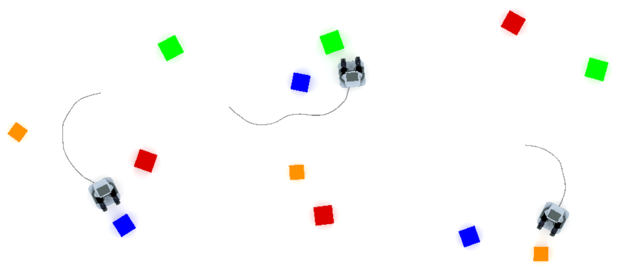


Fig. 6. Planning constraint inference and state-action inference runtime as a function of the number of constraints. In this comparison each n constraints required 2^n trajectories to form an equivalent state-action space.

The second factor that influences runtime is the time that it takes to represent the space of groundings. Effectively sampling the space of actions from the continuum of robot motion requires a forward model of the system dynamics. If deliberative or randomized sampling is used to generate the space of actions, this space could grow rapidly without control over their expressiveness or function. If instead constraints are used to form a variety of well-separated solutions, we incur the cost of numerous trajectory planning queries before inference time. The trajectory planner used to construct the examples in the corpora in Section VI-B required an average of 2.1 seconds to generate each motion. We also observed in the experiments described in Section VI-B that the Hybrid G^3-DCG model required only an average of 34.7 milliseconds to compute the most likely constraint set in the context of the language and the environment. Since the amount of time required to infer the most likely constraint set is relatively small and the minimum number of trajectories required to form a state-action space that is equivalent to a constraint set grows exponentially with the number of constraints, in practice it is far more efficient to infer the formulation of a planning problem and solve it once to find the trajectory that most likely corresponds to the natural language expression.

B. Accuracy

In the second experiment we assess the ability of planning constraint inference to determine the correct planning constraints from natural language instructions. We developed two corpora for this purpose. The first corpus consisted of 8 constraint sets and 16 environment models. Each environment consisted of four objects and one ground plane modeled as single rigid bodies and one robot that was modeled as a tree of 45 rigid bodies linked through prismatic and revolute joints for a total of 50 distinct objects. We generated a trajectory for each constraint set and environment for a total of 128 pairs of constraint sets and trajectories. We examined images of each trajectory and assigned 3 natural language instructions that describe the robot behavior. This initial dataset hand designed with purposefully varying verbs, adjectives, and nouns to create a diverse set of commands. The instructions in the first corpus contained 68 unique parts of speech, including 13 adjectives, 13 nouns, and 29 verbs. The first corpus was randomly divided into a training set of 96 examples and a test set of 288 examples. Three examples illustrating the environment model, robot trajectory, and assigned instructions are shown in Figure 7.



(a) "go to the blue box" (b) "move towards the green object" (c) "travel to the orange object"

Fig. 7. Images of labeled trajectories generated by constraint and environment sampling that form the training and test sets for constraint inference evaluation.

Each noun phrase in the G^3 and Hybrid G^3-DCG models uses a single factor to search the 50 objects for the one that is best described by the linguistic constituents. The state-action space for the G^3 model consisted of trajectories that satisfied each of the eight constraints sets in the current environment. The constraint set for the Hybrid G^3-DCG model consisted of 570 constraints that could be either active, inverted, or ignored. Performance of the two approaches on the training and test sets are shown in Table II. Each approach was given the command and environment model and asked to produce either the most likely trajectory or the most probable constraint set. A trial was deemed successful if it produced the inferred trajectory or constraint set matching the example associated with the command and environment model.

TABLE II

SUMMARY OF RESULTS FROM THE EVALUATION OF CORPUS 1

	accuracy (%)	
	Hybrid G^3-DCG (constraint)	G^3 (trajectory)
training	100.00	94.79
test	88.54	75.69

The second corpus consisted of examples labeled by experimental subjects to ensure a more realistic evaluation. Labels that contain more than one verb phrase were excluded from the experiments for the purposes of simplifying the annotation process. A total of 77 examples were randomly divided into a training and test sets of 34 and 43 examples each. The instructions in these examples contained 51 unique parts of speech, including 7 adjectives, 11 nouns, and 14 verbs.

Performance of the G^3 and Hybrid G^3-DCG models on the second corpus is shown in Table III. We observe that performance of both approaches is slightly inferior to that of the models trained by the first corpus. This is an expected result as the number of training examples is quite small in relation to the variety of language in the second corpus. We observe in both Table II and Table III that the two approaches share similar performance in both the training and test sets.

TABLE III
SUMMARY OF RESULTS FROM THE EVALUATION OF CORPUS 2

	accuracy (%)	
	Hybrid G^3-DCG (constraint)	G^3 (trajectory)
training	90.91	81.81
test	76.74	65.11

C. Conjunctions

The experiments from Section VI-B demonstrate that the Hybrid G^3-DCG model enables robots to search the space of planning constraints without reducing the likelihood of generating the desired trajectory. The Hybrid G^3-DCG model does however suffer from the same exponential growth in runtime when the space of groundings for objects transitions from individual objects to sets of objects, as it must for the expression illustrated by the models in Figures 3 and and 4. To show that the DCG model is capable of correctly inferring sets of objects and constraints, we augmented the first corpus from Section VI-B with examples that included more complex expressions that include conjunctions and the expression of multiple proximity and/or orientation constraints. In this example we assumed that there were 48 constraints between the parts of the robot and objects in the environment in the constraint set and and 50 objects in the object set. An example of the trajectory and the configuration space for the constraint set that was inferred by the DCG model for the natural language expression "move near the red box and the blue crate" is illustrated in Figure 1. The model correctly inferred the active expression of distance constraints between the robot base the two named objects and the inversion of the contact constraints between the robot body and the four boxes to avoid collisions in approximately 25 milliseconds.

D. Transferability

The ability to infer constraints from natural language instructions enables a number of interesting capabilities. First and foremost is the opportunity for transferability across platforms with similar physical characteristics. Since our inference model has geometric, kinematic, and semantic features and treats all robots simply as collections of rigid bodies, it is possible to train a model for one platform and apply it to another. To illustrate our point, we applied the inference model trained from the first corpus in Section VI-B to a simulated YouBot robot in a similar environment. We were able to correctly infer the active expression of a proximity constraint between the base of the YouBot robot and the green object for the command "go towards the green box", even though the model factors were trained with examples from a different platform and the command did not appear in the training examples. The phrases "go towards" and "the green box" did appear separately in the corpus but paired with different verb and noun phrases, illustrating one of the benefits of applying grounding graphs to natural language understanding. The generated trajectory, illustrated in Figure 8, used a platform-specific predictive motion model and control library that enables it to drive both forwards and backwards to the goal region.

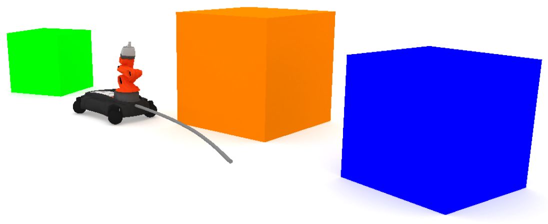


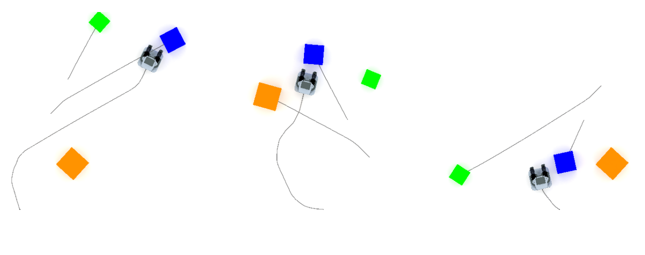
Fig. 8. A trajectory generated for a simulated YouBot robot subject to the command "go towards the green box" using a constraint inference model trained with examples from a different platform, such as the ones illustrated in Figure 7.

E. Dynamic Environments

Another feature of constraint inference is the ability to seamlessly handle dynamic environments. Since we only infer a set of desired relationships between rigid bodies, it is not necessary to explicitly sample paths or states in the time dimension. Figure 9 illustrates trajectories generated from the constraint set inferred from the command "approach the blue box". In each example, one of the three objects is assigned a non-zero velocity, sometimes disturbing other objects in the environment. Although paths vary significantly, the desired relationship between the robot and the blue box at the terminal state remains consistent across scenarios.

VII. RELATED WORK

Algorithms for inferring actions from language permeate the artificial intelligence and human-robot interaction literature. Most research centers on training probabilistic models from a corpus of examples to evaluate the correspondence between controllers, actions, or paths. Kress-Gazit et al [6] applies Linear Temporal Logic (LTL) to map structured language to robot trajectories. Vogel and Jurafsky [7] and Branavan et al [8] applies reinforcement learning to learn an optimal policy from language. Chen and Mooney [9]



(a) $|\mathbf{v}_{\mathbf{box}_{\mathbf{blue}}}(t_0)| > 0$ (b) $|\mathbf{v}_{\mathbf{box}_{\mathbf{orange}}}(t_0)| > 0$ (c) $|\mathbf{v}_{\mathbf{box}_{\mathbf{green}}}(t_0)| > 0$

Fig. 9. Trajectories generated from a constraint set inferred from the command "approach the blue box" in three different dynamic environments. In each the initial conditions of objects vary, resulting in significant differences in the generated path.

searches semantic maps for action sequences that maximize the metric of words and graphs. Matuszek, Fox, and Koscher [10] map utterances to a path description language to follow directions in a labeled map. Zender et al [11] builds a navigation roadmap of admissible states from sensor data and a natural language dialog system. Branavan et al [12] learns precondition relationships between objects in an environment from text to generate high-level plans. Duvallet et al [13] learns a policy that predicts the sequence of actions for direction following in unknown environments. The approach described in this paper differs from these works by using natural language instructions and environmental models to infer the most probable planning constraint set that defines the time-varying admissible regions of a task. The output from our inference algorithm is not a controller, action, path, policy, or set of precondition relationships, but rather is a description of the problem that can be solved by trajectory planning algorithms.

Silver, Bagnell, and Stentz [14] learn cost functions to indicate preference for terrain and maneuvers for mobile robot navigation. Cost functions do not explicitly restrict the configuration space and mix quantities to optimize (e.g. time, energy) with conditions that you want to satisfy (e.g. no collisions, no tip-over). This work is therefore complimentary to those that apply constrained optimization formulations of the trajectory planning problem.

VIII. CONCLUSIONS

Constraint inference improves the efficiency and generality of human-robot interaction by inferring the admissible regions of the state-space from a natural language instruction. Our approach preserves many of the benefits of traditional grounding graphs (e.g., infinite recursion of noun phrases) by only replacing factors used to infer single actions with many factors that infer the expression of constraints. This technique is compatible with recent advances in trajectory planning since we only use machine learning to form problems, rather than attempt to solve them.

One limitation of this approach is the requirement that all constraints intersect. A more general approach would permit additional set operators, such as unions or differences, enabling robots to perform commands with several acceptable outcomes. Efficient techniques for searching this larger space of constraints are an open area for investigation. Another weakness is the dependency on accurate sentence parsing. Since a sentence can have several different parses that are grammatically correct, a better approach could infer the desired parse structure from the environment or encode the uncertainty of the parse structure directly in the probabilistic graphical model. One last limitation is the reliance on aligned training data. We are interested in applying the techniques described in [15] to train model factors for planning constraint inference from an unaligned corpus of examples.

IX. ACKNOWLEDGMENT

This work was partially supported by the Robotics Consortium of the U.S Army Research Laboratory under the Collaborative Technology Alliance Program, Cooperative Agreement W911NF-10-2-0016, and by the Office of Naval Research under MURI N00014-09-1-1052.

REFERENCES

[1] S. Tellex, T. Kollar, S. Dickerson, M. Walter, A. G. Banerjee, S. Teller, and N. Roy, "Understanding natural language commands for robotic navigation and mobile manipulation," in *Proc. of the National Conf. on Artificial Intelligence (AAAI)*, San Francisco, CA, 2011.

[2] M. Marcus, B. Santorini, and M. Marcinkiewicz, "Building a large annotated corpus of English: the Penn Treebank," *Computational Linguistics*, vol. 19, no. 2, pp. 313–330, 1993.

[3] D. Lui and J. Nocedal, "On the limited memory BFGS method for large scale optimization," *Mathematical Programming*, vol. 45, pp. 503–528, 1989.

[4] D. Klein and C. Manning, "Accurate unlexicalized parsing," in *Proc. of the 41st Meeting of the Association for Computational Linguistics*, 2003, pp. 423–430.

[5] D. Younger, "Recognition and parsing of context-free languages in time n^3," *Information and Control*, vol. 10, no. 2, pp. 189–208.

[6] H. Kress-Gazit, G. Fainekos, and G. Pappas, "Translating structured english to robot controllers," *Advanced Robotics*, vol. 22, pp. 1343–1359, 2008.

[7] A. Vogel and D. Jurafsky, "Learning to following navigational directions," in *In Proc. of the Association for Computational Linguististics*, 2010, pp. 806–814.

[8] S. Branavan, H. Chen, L. Zettlemoyer, and R. Barzilay, "Reinforcement learning for mapping instructions to actions," in *Proc. of the Joint Conf. on Natural Language Processing*, Singapore, August 2009, pp. 82–90.

[9] D. Chen and R. Mooney, "Learning to interpret natural language navigation instructions from observations," in *Proc. of the 25th AAAI Conf. on Artificial Intelligence*, August 2011, pp. 859–865.

[10] C. Matuszek, D. Fox, and K. Koscher, "Following directions using statistical machine translation," in *In Proc. of the ACM/IEEE Int. Conf. on Human-Robot Interaction*, 2010, pp. 251–258.

[11] H. Zender, O. Mozos, P. Jensfelt, G. Kruijff, and W. Burgard, "Conceptual spatial representations for indoor mobile robots," *Robotics and Autonomous Systems*, vol. 56, no. 6, pp. 493–502, June 2008.

[12] S. Branavan, N. Kushman, T. Lei, and R. Barzilay, "Learning high-level planning from text," in *Proceedings of the 50th Annual Meeting of the Ass. for Computational Linguistics*, vol. 1, 2012, pp. 126–135.

[13] F. Duvallet, T. Kollar, and A. Stentz, "Imitation learning for natural language direction following through unknown environments," in *Proc. of the 2013 Int. Conf. on Robotics and Automation*, May 2013.

[14] D. Silver, J. Bagnell, and A. Stentz, "Learning autonomous driving styles and maneuvers from expert demonstration," in *Int. Symposium on Experimental Robotics*, June 2012.

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