Grounding Verbs of Motion in Natural Language Commands to Robots

Thomas Kollar, Stefanie Tellex, Deb Roy, and Nicholas Roy

Abstract To be useful teammates to human partners, robots must be able to follow spoken instructions given in natural language. An important class of instructions involve interacting with people, such as “Follow the person to the kitchen” or “Meet the person at the elevators.” These instructions require that the robot fluidly react to changes in the environment, not simply follow a pre-computed plan. We present an algorithm for understanding natural language commands with three components. First, we create a cost function that scores the language according to how well it matches a candidate plan in the environment, defined as the log-likelihood of the plan given the command. Components of the cost function include novel models for the meanings of motion verbs such as “follow,” “meet,” and “avoid,” as well as spatial relations such as “to” and landmark phrases such as “the kitchen.” Second, an inference method uses this cost function to perform forward search, finding a plan that matches the natural language command. Third, a high-level controller repeatedly calls the inference method at each timestep to compute a new plan in response to changes in the environment such as the movement of the human partner or other people in the scene. When a command consists of more than a single task, the controller switches to the next task when an earlier one is satisfied. We evaluate our approach on a set of example tasks that require the ability to follow both simple and complex natural language commands.

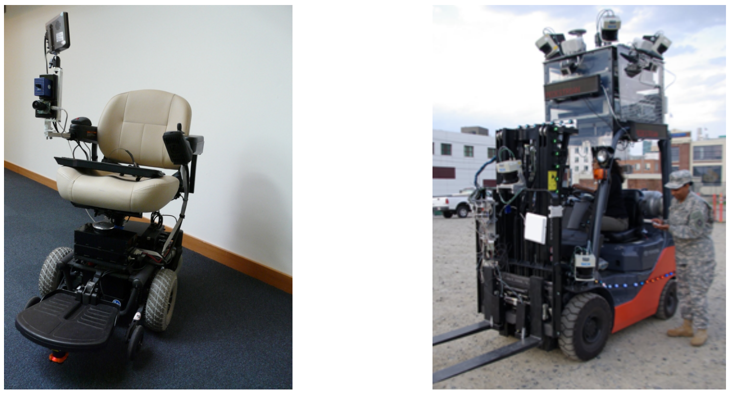
1 Introduction

In order to achieve higher levels of autonomy, robots need to be able to interact naturally with people in unstructured environments. One of the most natural ways to instruct robots is to use spoken natural language, enabling robots such as those

Thomas Kollar and Nicholas Roy
Computer Science and Artificial Intelligence Lab., Massachusetts Institute of Technology, 77 Massachusetts Ave. Cambridge, MA 02139. e-mail: tkollar@csail.mit.edu, nickroy@csail.mit.edu

Stefanie Tellex and Deb Roy
MIT Media Lab. 75 Amherst St. Cambridge, MA, 02139 e-mail: stefie10@media.mit.edu, dkroy@media.mit.edu

The first two authors contributed equally to this paper



(a) Robotic Wheelchair (b) Robotic Forklift

Fig. 1: Two systems that rely on natural language interaction: an autonomous wheelchair and a robotic forklift.

shown in Figure 1 to be commanded easily. For example, a patient in a robotic wheelchair might say, “Follow the nurse to the exit,” or the operator of a robotic forklift might say, “Meet me in receiving at 3pm.”

To explore the types of natural language commands that people would like to use, we collected an open-ended corpus by asking people to imagine commands they might give to a robot in different scenarios. Sample commands from this corpus included “Meet the two people at the elevators and bring them to 32-331,” and “Follow Tom when you see him.” Many of the commands in the corpus included verbs relating to people’s motion (or *verbs of motion*); these verbs are the focus of this paper.

Understanding verbs of motion requires reasoning about the path of both the person and the robot, converting the spoken words in the instructions into low level actions. Previous approaches [Wei et al., 2009, Kollar et al., 2010] converted a natural language instruction into a fixed sequence of actions in the environment and sent this plan to the robot for execution. However, when following a pre-computed plan, the robot lacks the ability to respond to a dynamic environment: if the person takes a different route to the exit, the robot is unable to follow.

We present an approach to solving this problem by defining a cost function over actions the robot can take and incrementally satisfying parts of the command. The system involves three components: a cost function, an inference algorithm, and a controller. Similar to Kollar et al. [2010], our cost function is the log-likelihood of the actions given the language. We expand this cost function in order to handle plans that involve the motion of other people in the environment. The inference algorithm quickly maximizes over plans, allowing us to connect the meaning of each word or phrase in the sentence to the environment in a domain-independent way. Finally, a controller enables us to re-plan at each timestep and also to determine when to move on to the next component of the language.

We evaluate our approach on a corpus of directives that was constructed around combinations of five motion verbs: “go,” “meet,” “follow,” “bring,” and “avoid.” For each verb, we collected an evaluation corpus that consisted of approximately ten primitive natural language directives such as, “Follow the person to the kitchen.” and compound commands involving several phrases such as, “Follow the person to the kitchen. Then move towards the bathroom. Next go down the hall to the lounge.” For each directive, we created a set of initial conditions in a simulator, situating a virtual robot and person in a map of a real office environment. A human controlled a simulated robot to obey the command, which was then used to train and evaluate models of the meanings of verbs. We evaluated the performance of our model on a held-out test set of the corpus. The system successfully followed most of the commands in our corpus, showing our approach captured much of the compositional structure of these natural language commands.

2 Related Work

There is been a rich body of work building language understanding systems for robots. Previous work focused on natural language route instructions [Kollar et al., 2010, Matuszek et al., 2010, MacMahon et al., 2006, Shimizu and Haas, 2009], using a natural language command to infer a path through an unchanging environment. Vogel and Jurafsky [2010] used reinforcement learning to automatically learn a model for understanding route instructions. In contrast, our work focuses on situations in which the environment changes over time, and the robot must adapt. Hsiao et al. [2008] and Skubic et al. [2004] presented frameworks for robotic language understanding in which the actions themselves are responsive to changes in the environment. Our approach creates an entirely new plan in response to these changes, meaning that the system’s repertoire of actions can be simpler.

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]. Some systems accept input and corrections from the teacher in the form of language to correct the plan [Nicolescu and Mataric, 2003]. 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 an action from the natural language commands, rather than from watching the teacher perform an action. The system is trained off-line, and the task of the robot is to respond on-line to the natural language command.

The problem of predicting the cost function for a task in a domain has received attention in the inverse optimal control and inverse reinforcement learning domains. Abbeel and Ng [2004] and Peters and Kober [2009] have shown that learning from demonstrations of trajectories could be improved by inferring the cost function, rather than recovering the policy directly from the demonstrated trajectories. Silver et al. [2009] showed that an autonomous vehicle could recover a global cost function from local examples of driving. While these classes of algorithms are similar in spirit to ours, the assumption in previous work was that the cost functions must be stationary over the environment and do not depend on an external signal, which in our case is the instruction from the human teammate.

Partially Observable Markov Decision Processes (POMDPs) are a planning framework that have been used for robot control [Kaelbling et al., 1998]. In order to use a POMDP to control a robot to follow an instruction, the natural language command specifies a reward function. However, in the standard formulation, the reward function depends only on the current state and action. This is insufficient to capture verbs like “bring” or “meet” which require access to the history of states. Instead, we extend the approaches of Toussaint and Storkey [2006] and Attias [2003], which develop graphical models analogous to MDPs and POMDPs. By defining the cost function as a log-likelihood, we are free to reason using the entire history of states and factor the distribution into independent components in order to learn the model with less training data.

3 Technical Approach

To convert from natural language expressions into robot behavior, we take as input a natural language command and output the sequence of actions that minimizes the cost function C that corresponds to the command. If C is a cost function, s_i are states of the world, L is the natural language command and Φ is a correspondence variable that determines the mapping between the language and states in the world, then the problem at each timestep is to find a sequence of states and Φ that minimizes the cost function:

$$\operatorname*{argmin}_{s_1 \dots s_T, \Phi} C(s_1 \dots s_T, |\Phi, L) \; . \tag{1}$$

We define a cost function as the log-likelihood of the actions given the language and the correspondence variable:

$$C(s_1 \dots s_T | \Phi, L) \triangleq -\log(p(s_1 \dots s_T | \Phi, L)) \tag{2}$$

Conditioning on L introduces a problem of data sparsity; natural language instructions can have almost arbitrary structure and almost arbitrary vocabulary. However, we have shown previously [Kollar et al., 2010] that we can use a shallow semantic structure called the *spatial description clause* (SDC) to parse the instructions L into a set of separable instruction clauses $L = \{SDC_1 \dots SDC_K\}$. SDCs enable us to factor the distribution and learn each factor separately. Each SDC corresponds to a single task that the robot must take in following the text of the instructions, and consists of a figure f_k, verb v_k, spatial relation sr_k, and landmark l_k, any of which can be unspecified. Both the figure and landmark may refer to the path of an object or person, the spatial relation describes the general extent of the task, and the verb specifies how the task should take place. For example, for the command “Meet the person,” the figure is implicitly “(you)”, the verb is “Meet,” the spatial relation is implicit, and the landmark is “the person.” A parse tree for the command, “Meet the person at the kitchen.” showing a sequence of two SDCs is shown in Figure 2.

The system automatically extracts a sequence of SDCs from the natural language command. Since each SDC generally corresponds to individual tasks that the robot should execute, we can assume that each SDC is independent given the entire state sequence, thus enabling us to factor the distribution as

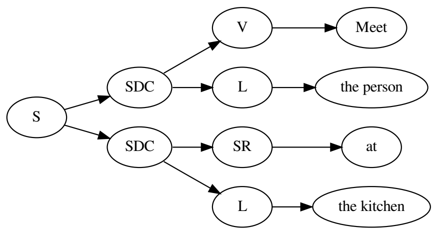


Fig. 2: SDCs for the sentence "Meet the person at the kitchen." Here, *S* is the entire sentence, *SDC* is a spatial description clause, *F* is the figure, *V* is the verb, *SR* is the spatial relation, and *L* is a landmark.

$$p(s_1 \ldots s_T|\Phi,L) \propto \prod_{k=1}^{K} p(\mathrm{SDC}_k|s_{\phi_k})p(s_1 \ldots s_T), \tag{3}$$

where the probability of SDC_k is

$$p(\mathrm{SDC}_k|s_{\phi_k}) = p(f_k|s_{\phi_k}) \cdot p(sr_k|s_{\phi_k}) \cdot p(v_k|s_{\phi_k}) \cdot p(l_k|s_{\phi_k}) \tag{4}$$

ϕ_k assigns parts of the states from $s_{1:T}$ to the corresponding fields of SDC_k. $s_1 \ldots s_T$ describes a state sequence of the robot as it follows the instructions; $p(v_k|s_{\phi_k})$ is the probability of a verb in the *k*th SDC given a piece of the trajectory; $p(sr_k|s_{\phi_k})$ is the probability of a particular spatial relation and landmark given a piece of the trajectory occurring in a specific location of the environment; $p(f_k|s_{\phi_k})$ and $p(l_k|s_{\phi_k})$ are the probabilities of the respective noun phrases given an object trajectory sequence picked out by s_{ϕ_k}. The prior $p(s_1, \ldots s_T)$ enforces the sequential nature of the robot and person paths as well as the stationarity of the objects. We define the prior so that, for example, state sequences that reverse direction abruptly are lower probability.

Finally, we must define the state space and provide groundings for each term in the factorization. In order to understand verbs like "follow," the state must include both the robot position as well as the person's position. In order to understand landmarks like "the kitchen," the state must also include the location of objects in the environment over time. Using this cost function, the robot can evaluate how well candidate paths match the natural language command. The key components of this distribution model are summarized in Table 1.

4 Model Learning

To implement our model, we must ground each of the components of Equation 4. The following sections describe how we model the figure, verb, spatial relation, and landmark from the SDC.

States	The state s_i includes the current location of the robot and the person along with the location of detected objects, each of which additionally has a class name. A sequence of robot positions is defined as r and a sequence of person poses is defined as p.
Observations	The observations are the observed natural language in the form of a sequence of SDCs, $L = \{SDC_1 \ldots SDC_K\}$.
Correspondence	The correspondence variable Φ encodes the bindings between the state space and the language. In particular, it encodes the physical objects that bind to the landmark phrase of each SDC, as well as the sequence of states that binds to an entire SDC.

Table 1: Components of our model.

4.1 Verbs

Verbs are one of the most complex and important parts of natural language directives. They define events that take place between one or more entities in the world, imposing a rich internal structure. Syntactically, they form the core of the sentence, relating the rest of the phrases together to form a coherent event.

For SDCs containing a transitive verb of motion such as "follow," we can decompose the motion trajectories of the person following and the person being followed:

$$p(\text{follow} = \text{True}|r = \text{robot poses}, p = \text{person poses}) \tag{5}$$

$$= p(\text{follow} = \text{True}|f_1(r,p)\ldots f_N(r,p)) \tag{6}$$

The figure, in this case a sequence of robot poses r, and the landmark, in this case a sequence of person poses p, are given to the model as a sequence of locations tagged with times and are converted into a set of features $f_i(r,p)$. Given a training corpus and this set of features, a classifier is learned for each verb that describes the probability that this verb exists (follow = True) given the features of the two paths. This distribution needs to be conditioned on potentially the entire state sequence in order to capture the internal structure of a verb. For example, "meet" is not just the robot being near the person at a particular state. Rather, the robot must start out far away from the person, and then end up close to them. The history of states is needed to make this determination.

In order to compute features $f_i(r,p)$, the figure and the landmark paths must be converted to a set of features. In order to convert these paths to a set of features, we take a sequence of windows and compute low-level features with respect to these windows. Each window describes a part of both paths corresponding to 1 second of time. Each of the low-level features $g_k(r,p,t)$ is Boolean and is computed with respect to the robot positions r and person positions p paths for a one-second window with center at time t. This results in a vector of feature values, one value for each time $0 < t < T$. The low-level Boolean features can be seen in Table 2. Additional features are generated that involve the conjunction of two of the low-level features (e.g., $g_k(r,p,t) = g_i(r,p,t) \wedge g_j(r,p,t)$), where i, j are from Table 2 and $\wedge$ is the logical *and* operation.

Low-Level Features	
$g_1(r,p,t)$ = MovingTowards(landmark, figure, time)	$g_2(r,p,t)$ = MovingTowards(figure, landmark, time)
$g_3(r,p,t)$ = IsVisible(figure, landmark, time)	$g_4(r,p,t)$ = IsClose(figure, landmark, time)
$g_5(r,p,t)$ = IsMoving(landmark, time)	$g_6(r,p,t)$ = IsMoving(figure, time)

Table 2: The set of low-level features used to learn the probability of a verb. MovingTowards(x,y,t) is true when x is moving towards the current location of y at time t. IsClose(x,y,t) is true when the distance between x and y is less than a set threshold at time t. IsMoving(x,t) is true when x is moving at time t. IsVisible(x,y,t) is true when x is visible from y at time t.

Finally, we take the mean of the of each of these Boolean vectors in order to compute the features used in the learning:

$$f_i(r,p) = \frac{1}{T}\sum_{t=1}^{T} g_i(r,p,t). \tag{7}$$

A corpus of labeled examples for each verb, together with these features $f_i(r,p)$, were used to train a naive Bayes classifier. Performance curves for the verbs trained on a corpus of labeled examples is shown in Figure 3, showing that standard machine learning techniques suffices to learn good verb models. In addition, since the features used to learn the verbs are scale-invariant (Equation 7) and we have shown that the verb classifier generalizes to new paths, we should expect that our algorithm is able to generalize to new environments. Verbs can be composed with any of the spatial relations and landmarks, according to Equation 4, which allows for the composition of novel commands that the robot has never seen before.

4.2 Figure, Spatial Relations and Landmarks

To ground spatial relations and landmarks, we use the models described in Kollar et al. [2010]. We review the work briefly here for completeness.

Spatial relations are words such as "to," "past," and "across," that describe the motion of an object in relation to a landmark. The second term in Equation 4, $p(sr_k|s_{\phi_k})$ corresponds to a model for the meaning of spatial prepositions. The system models this distribution via supervised classifier trained on labeled examples of each preposition. Each distribution takes as input the geometry of a path and a landmark object, computes a set of features and outputs a probability that the situation can be described using that spatial relation. In our previous work we showed the feasibility of using these models of spatial relations to follow route directions [Kollar et al., 2010].

Both the figure and landmark (the first and last parts of Equation 4) correspond to the probability of seeing a textual label such as "kitchen" given a particular object in the environment and the labels of other visible objects in the environment: $p(l_i|s_{\phi_k})$. We estimate the probability of a phrase given observed objects in the environment using co-occurrence statistics learned from tags for over a million images downloaded from the Flickr website [Kollar et al., 2010, Kollar and Roy, 2009]. Using this corpus, the robot can infer that if it detects a refrigerator at a particular location, it is likely to be in a kitchen, enabling the robot to estimate the location of novel landmark words in the text.

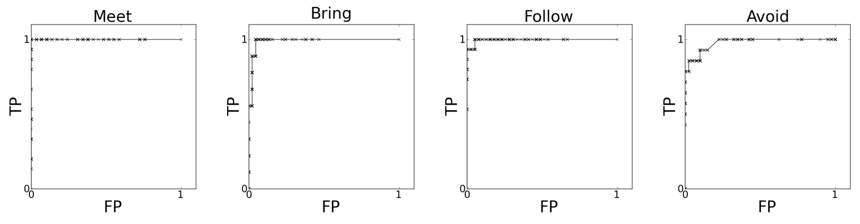


Fig. 3: ROC curves showing the performance of distributions for various verbs when they are treated as classifiers. TP is true positive rate, and FP is false positive rate.

By combining spatial relations and landmarks that were not in the training dataset, we can compose the two in novel ways. Since the spatial relation/landmark component relies only on the local context of the robot and scale-invariant features describing the relationship of the robot path to the geometry of the landmark, we can expect the approach to generalize to new situations in new environments.

5 Inference

Given a natural language command, the goal of inference is to compute the best state sequence and assignment of the language onto the states (or sequence of states) according to the cost function at the current time.

In order to perform this inference, we use forward search at each timestep to find the sequence of actions that minimizes the learned cost function. In general, the robot will have executed a sequence of states up to the current time, and will thus have a state history H as well as the future state sequence. The goal of inference is then to compute the minimal cost state sequence and to infer the correspondence variable Φ:

$$\underset{s_1 \ldots s_T, \Phi}{\operatorname{argmin}} C(H; s_1 \ldots s_T | \Phi, L) \tag{8}$$

Searching deeply enough to find a low cost plan is key to understanding verbs of motion. A key optimization that makes this possible is our choice of action representation. Rather than using low-level actions in a grid-map, we represent actions as motions between locations in a topological road map that is automatically generated from a gridmap of the environment. The topological roadmap corresponding to a SLAM-generated gridmap [Grisetti et al., 2007] of an office environment is shown in Figure 4. Junction and end points in the road map are shown as circles, and correspond to locations where paths diverge or dead-end. We refer to an edge in this graph as a *pathlet*. By considering moves in the topological map rather than in a grid map, the system can search farther into the future for a plan that minimizes the cost. Candidate plans are generated using breadth-first search, starting from the current state.

Pseudocode for this approach can be seen in Algorithm 1. Initially, the algorithm computes the set of paths for the a sequence of robot poses r_i and a sequence of a person's poses p_i corresponding to breadth-first search, extending pathlets until some depth or number of paths is achieved. The algorithm then iterates through these paths and, depending on type of the landmark field (either an object or a person),

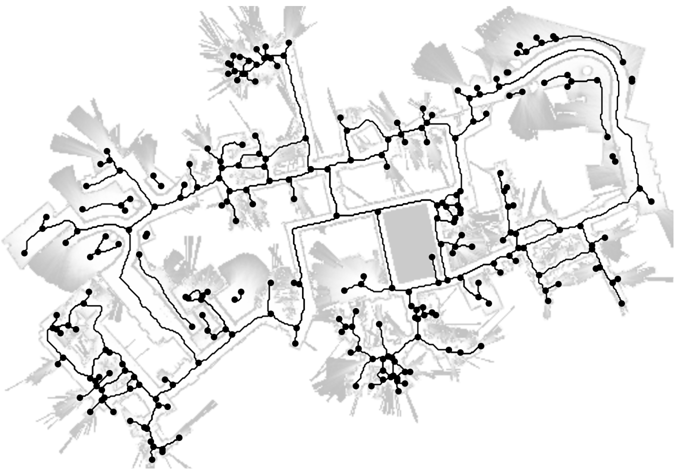


Fig. 4: The automatically generated skeleton of the map, with junction points and end points labeled. We refer to an edge in this graph as a *pathlet*.

Algorithm 1 Inference($SDC_1 \dots SDC_K$, H)

```
min_cost ← $MAX\_INT$
min_state_seq ← []
$p_1 \dots p_N, r_1 \dots r_N$ ← generate_breadthfirst_paths($H$[end])
for $p_i, r_i \in \{(p_1, r_1) \dots (p_N, r_N)\}$ do
    cost ← 0; objs = []
    for $SDC_j \in \{SDC_1, \dots SDC_K\}$ do
        if $l_j$ is an object then
            cost = cost + $\min_{o_k, \text{pathlet on } r_j} -\log p(SDC_j | o_k, \text{pathlet})$
            objs ← [objs, $o_{\min}$]
        end if
        if $l_j$ is a person then
            cost = cost − $\log p(SDC_j | H, p_i, r_i)$
            objs ← [objs, []]
        end if
    end for
    if cost − $\log p(r_i, p_i, \text{objs})$ < min_cost then
        min_cost = cost − $\log p(r_i, p_i, \text{objs})$
        min_state_seq = [$p_i, r_i$, objs]
    end if
end for
return min_state_seq, min_cost
```

grounds the probability of an SDC in different ways. In the case of an object we take the set of pathlets on the robot path and maximize over candidate objects o_k and pathlets. Thus, if *some* part of the path looks like “past the bathroom,” then the score will be high. Although, we could look at larger parts of the path, this would have been computationally intractable. The approximation via pathlets enables us to cache the probabilities along pathlet edges for each planning step.

In case the SDC refers to a person, we take the entire history H and planned path of the person p_i and robot r_i and compute the probability of an SDC given this pair of paths. Partitioning the path in a computationally efficient way was out of scope for this work. Finally, we add in the prior over the state sequence $p(r_i, p_i, \text{objs})$ and store the minimum cost and corresponding state sequence.

This inference algorithm enables the system to efficiently find a plan corresponding to the natural language command.

6 Controller

In order to respond to the actions of people in the environment, we must use a controller to determine a current plan as the state of the world changes. Since elements of the state space are not under our control, a plan made at the beginning of time will not remain valid if the person does something unanticipated. In the case of "Follow the person," we may have very little idea exactly where the person will go. Because the system needs to respond in real-time, the inference will only have time to search a limited distance in the future.

The result of this is that not every SDC will have a corresponding path in either the path history or in the planned future path, especially for complex commands such as "Go to the restroom, and then follow the person to the kitchen." To solve this problem, the system sequentially applies the SDCs to a sequence of states. At the beginning, it considers state sequences using only the first SDC and executes the best plan at that time (e.g., $c = 1$). When the algorithm does not re-plan, it executes the action that gets the robot to the next state in the current best plan (e.g., by calling achieve_state on s_i). When the cost of the plan begins to monotonically increase over the set of the K most recent plans (e.g., an inflection point has been reached), the algorithm will move to the next SDC by incrementing c. This heuristic means our algorithm will move from an SDC such as "to the kitchen" to the next SDC when the best future plan looks less like "to the kitchen" than the history. In order to ensure that the entire plan is consistent with the language, the algorithm will consider both the current SDC and the previous SDCs when ranking a sequence of states. This process repeats until the robot has executed all of the SDCs, at which point the al-

Algorithm 2 Controller($SDC_1 \ldots SDC_N$)

```
$i \leftarrow 0; c \leftarrow 1; H \leftarrow [s_0]; \text{costs} \leftarrow []$
while all SDCs are not satisfied do
    if is_replan_timestep then
        $s_1 \ldots s_T, \text{cost} \leftarrow \operatorname{argmin}_{s_1 \ldots s_T, \Phi} C(H, s_1 \ldots s_T | \Phi, SDC_1 \ldots SDC_c)$
        $\text{costs} \leftarrow [\text{costs}, \text{cost}]; i \leftarrow 0;$
    end if
    if $\text{costs}[\text{end}] < \min \text{costs}[end - K : end - 1]$ then
        $c \leftarrow c + 1$
    end if
    achieve_state($s_i$)
    $H \leftarrow [H, s_i]$
    $i \leftarrow i + 1$
end while
```

gorithm terminates. Pseudocode is presented in Algorithm 2 and a worked example is shown in Figure 5.

We found that in practice searching between 400 and 1000 paths gave good results in a reasonable amount of time. Without using a verb model, the model generates a plan in less than a second; when using verb models, the model generates a plan approximately every five seconds. Performance could easily be improved by parallelizing the system to run on multiple CPU cores.

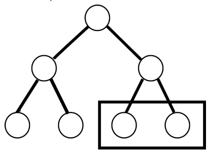
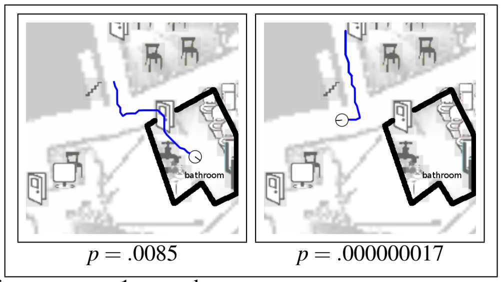
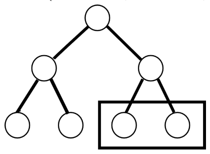
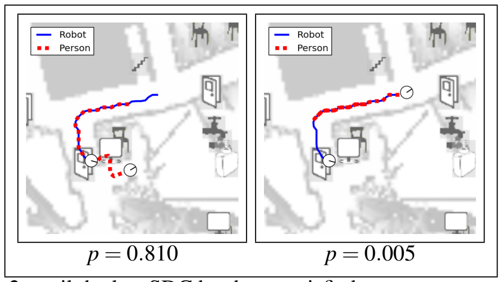
Input Language:	"Go to the restroom and then follow the person."
Parsed SDCs:	SDC(v="Go", sr="to", l="the restroom"), SDC(v="Follow, l="the person")
Step 1:	Evaluate first SDC; c=1 SDC(v="Go", sr="to", l="the restroom")   $p = .0085$ $p = .000000017$
Re-plan:	Goto step 1 and run inference every 1 second
Inflection Point	Robot arrives at the restroom; the probability of the first SDC goes down.
Step 2:	Evaluate both SDCs; c=2 SDC(v="Go", sr="to", l="the restroom"), SDC(v="Follow, l="the person")   $p = 0.810$ $p = 0.005$
Finish:	Re-plan, executing step 2, until the last SDC has been satisfied.

Fig. 5: Worked example of the system in action for the command "Go to the restroom, and then follow the person," showing high- and low- scoring examples. The language is first parsed into two SDCs and we start out with just the first SDC ($c = 1$ from Algorithm 2). While the plans look like the first SDC, the robot will keep executing the associated action. After the robot goes "to the bathroom," the system detects that all future paths are worse examples of this SDC and it adds the second SDC "follow the person". Until another inflection point happens, the robot recomputes its plan and executes it.

7 Evaluation

In order to evaluate our system, we created a corpus pertaining to spatial motion verbs. We focused on a set of five verbs: "bring," "meet," "avoid," "follow," and "go." We created about ten natural language commands that used each verb. In addition we created ten compound commands that involved at least two different verbs, such as “Follow the person to the kitchen. Then move toward the bathroom. Next go down the hall to the lounge.” For each command we created one or more scenarios. A scenario consists of an initial location for the robot and the simulated person (if any) in a map of a real office environment, as well as a behavior model for the person.

To collect the corpus for each scenario, an annotator controls the virtual robot to follow the command. For example, for “Follow the person to the kitchen,” a scenario would have the robot and the person starting in a particular location in the environment, and the person moving along a pre-recorded trajectory to the kitchen. A human annotator then completes the scenario by moving the robot in a way that obeys the natural language command. Thus the corpus consists of natural language commands, together with examples of a person’s behavior when following these commands in a simulated environment. We use this corpus for training generic models of the meanings of verbs, as well as evaluating our system.

To evaluate our system using held-out data from this corpus, we used our inference algorithm to control the robot’s activity, given the same information a person had when creating the corpus. When the behavior of the robot was judged to have followed the natural language command, then we counted it as correct. When any part of the robot’s behavior did not correspond to the command, then the behavior was judged to be incorrect. These results are shown in Table 3.

	Go	Follow	Avoid	Meet	Bring	**Overall**
Hand	90%	80%	78%	70%	29%	**69%**
Auto	-	-	-	-	-	**60%**
Compound	-	-	-	-	-	**60%**

Table 3: Percentage of time that our algorithm followed the language exactly for various command sets. *Hand* shows an experiment with 46 commands and annotated object detections. *Auto* shows results for 10 commands and a fully automatic object detector. *Compound* shows results for 10 compound commands involving two or more verbs with annotated object detections.

Most of the analysis used a map populated by hand with the labels of certain classes of objects in order to eliminate a dependency on noisy object detection. To include the effects of potentially noisy object recognition, we also present results for creating this semantic map automatically. For the automatically generated detections, we used the wheelchair robot depicted in Figure 1(a), equipped with a camera and LIDAR to drive around the environment. Then we used an object detector [Felzenszwalb et al., 2008] to detect six types of objects to seed the map. Figure 6 shows the object detector automatically detecting a microwave in the environment.

We also quantitatively compared the robot’s behavior to the behavior of a human annotator who controlled the robot. To do this, we had two different human annotators control the robot for the same scenario. We also had our system control the robot. Each example in the corpus is associated with two values: the difference between our system and a human annotator, and the difference between the two human annotators. We plot these values as a scatter plot in Figure 7.

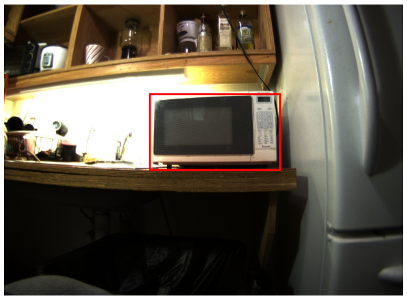


Fig. 6: An object detection of a microwave from the environment.

For all verbs, the human-human edit distance correlates with the human-robot edit distance. For verbs such as "go" and "follow," there is less variation in correct paths and all edit distances are small. For "avoid," there are many possible paths, so the edit distances are larger for both the human and robot, and there is therefore less of a correlation between edit distance and example correctness.

The verb "bring" performs much worse than the others in our test set. This disparity occurs because "bring" events are longer and contain significant internal structure. Our algorithm cannot search deeply enough to find a complete "bring" event in the time it has, meaning that it cannot choose the optimal plan. Optimizing and parallelizing our system to enable it to search deeper in the plan space could alleviate this problem.

8 Experimental Insights / Future Work

Some parts of the language were not currently handled by our decomposition into SDCs, such as events where the robot must recognize a time or a situation that is happening and conditionally execute the corresponding actions such as, "At noon, come to my office and lead the visitor to the kitchen." In addition, there are actions that must be repeated until another event occurs, such as "Make sure no one touches my project while I'm at group meeting." Extending the system handle language like this requires the ability to recognize events and react accordingly.

Table 3 shows that our inference process performs significantly worse for verbs with a complex internal event representation than for other verbs. With the verb "bring," this loss in performance happens because the cost function must reflect whether the robot has first approached and collected the object or person to be brought to another location. After the collection, the cost function must move the robot to a destination. As a result, the cost function is non-Markov. This problem might be overcome by adding one bit of memory but inferring from a set of natural language instructions that extra bits of memory are needed, and for what tasks, is an open problem.

9 Contributions

The contribution of this work is a planning framework that enables robots to understand verbs of motion. The system follows natural language commands by finding a plan that minimizes a cost function which relates the language to the plan. We

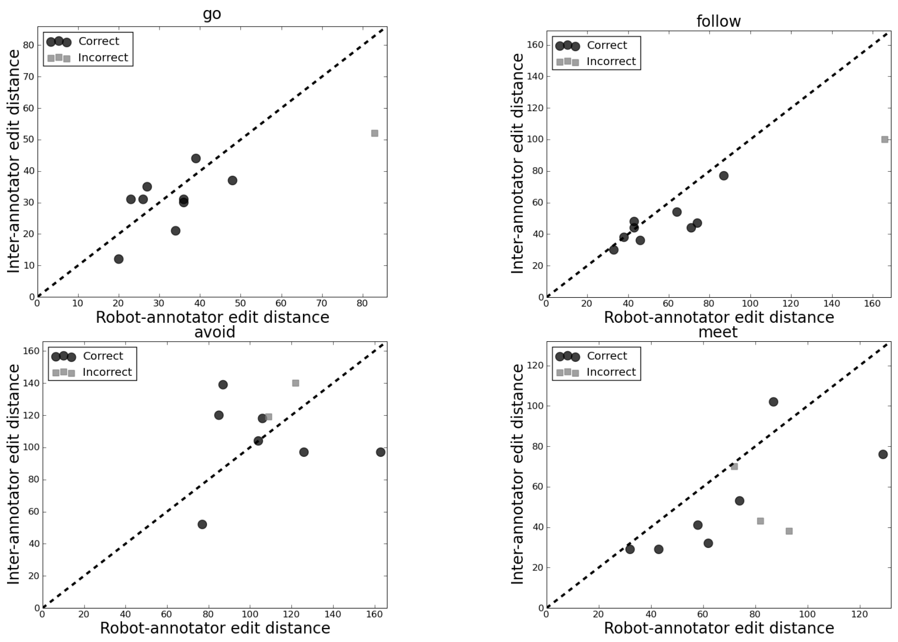


Fig. 7: Robot-annotator and inter-annotator edit distance for four of the modeled verbs. On the vertical axis is the agreement between the test robot annotations and those given by another human annotator. On the horizontal axis is the agreement between the path taken by our algorithm and a human. Both are measured by mapping each path onto a skeleton of the map, taking each location in the map as if they were a character and computing the edit distance between two paths (strings). In this graph, points above the line indicate that the robot matches the ground-truth path better (according to edit-distance) than another human performing in the same scenario.

provide models for the meanings of spatial motion verbs such as “follow,” “avoid,” “meet,” “bring,” and “go.” The system composes these models with other parts of the language such as landmarks and spatial relations using spatial description clauses, a shallow semantic structure that corresponds to independence assumptions in the language that can be used in the model. The system re-plans at each timestep, enabling it to respond dynamically to changes in the environment. It handles compound commands by looking for an inflection point in the cost function to decide when to move from one clause to the next. Our framework enables robots to understand verbs of motion in natural language commands and demonstrates that novel behavior can be generated in a closed-loop manner for a large number of natural language commands.

Acknowledgments

We are grateful for the support of the Office of Naval Research, which supported Thomas Kollar and Stefanie Tellex under MURI N00014-07-1-0749.

References

Yuan Wei, Emma Brunskill, Thomas Kollar, and Nick Roy. Where to go: Interpreting natural directions using global inference. In *ICRA*, 2009.

Thomas Kollar, Stefanie Tellex, Deb Roy, and Nicholas Roy. Toward understanding natural language directions. In *Proceedings of HRI*, 2010.

C. Matuszek, D. Fox, and K. Koscher. Following directions using statistical machine translation. In *Proceedings of HRI*, 2010.

Matt MacMahon, Brian Stankiewicz, and Benjamin Kuipers. Walk the talk: Connecting language, knowledge, and action in route instructions. *Proceedings of the National Conference on Artificial Intelligence*, pages 1475—1482, 2006.

Nobuyuki Shimizu and Andrew Haas. Learning to follow navigational route instructions. In *IJCAI'09: Proceedings of the 21st international jont conference on Artifical intelligence*, pages 1488–1493, San Francisco, CA, USA, 2009. Morgan Kaufmann Publishers Inc.

Adam Vogel and Dan Jurafsky. Learning to follow navigational directions. In *ACL '10: Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 806–814, Morristown, NJ, USA, 2010. Association for Computational Linguistics.

Kai-yuh Hsiao, Stefanie Tellex, Soroush Vosoughi, Rony Kubat, and Deb Roy. Object schemas for grounding language in a responsive robot. *Connect. Sci*, 20(4):253–276, 2008.

M. Skubic, D. Perzanowski, S. Blisard, A. Schultz, W. Adams, M. Bugajska, and D. Brock. Spatial language for human-robot dialogs. *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*, 34(2):154–167, 2004. ISSN 1094-6977. doi: 10.1109/TSMCC.2004.826273.

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.

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

Stefan Schaal, Auke Ijspeert, and Aude Billard. Computational approaches to motor learning by imitation. *Philosophical Transactions of the Royal Society B: Biological Sciences*, 358(1431): 537–547, March 2003. ISSN 0962-8436. PMID: 12689379 PMCID: 1693137.

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

M. Nicolescu and M. Mataric. Natural methods for robot task learning: instructive demonstrations, generalization and practice. In *Proc. AAMAS*, 2003.

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

P. Abbeel and A. Ng. Apprenticeship learning via inverse reinforcement learning. In *Proc. ICML*, 2004.

J. Peters and J. Kober. Using reward-weighted imitation for robot reinforcement learning. In *Proc. Inter. Symp. on Approximate Dynamic Programming and Reinforcement Learning*, 2009.

David Silver, J. Andrew Bagnell, and Anthony Stentz. Perceptual interpretation for autonomous navigation through dynamic imitation learning. In *Proc. ISRR*, 2009.

Leslie P. Kaelbling, Michael L. Littman, and Anthony R. Cassandra. Planning and acting in partially observable stochastic domains. *Artificial Intelligence*, 101(1-2), 1998.

M. Toussaint and A. Storkey. Probabilistic inference for solving discrete and continuous state Markov Decision Processes. In *Proceedings of the 23rd international conference on Machine learning*, page 952. ACM, 2006.

H. Attias. Planning by probabilistic inference. In *Proceedings of the 9th International Workshop on Artificial Intelligence and Statistics*, 2003.

Thomas Kollar and Nick Roy. Utilizing object-object and object-scene context when planning to find things. In *IEEE International Conference on Robotics and Automation*, 2009.

G. Grisetti, C. Stachniss, and W. Burgard. Improved techniques for grid mapping with Rao-Blackwellized particle filters. *IEEE Transactions on Robotics*, 23(1):34–46, 2007.

P. Felzenszwalb, D. Mcallester, and D. Ramanan. A discriminatively trained, multiscale, deformable part model. In *CVPR-2008*, June 2008.