Imitation Learning for Natural Language Direction Following through Unknown Environments

Felix Duvallet
Robotics Institute
Carnegie Mellon University
felixd@cmu.edu

Thomas Kollar
Computer Science
Carnegie Mellon University
tkollar@cmu.edu

Anthony Stentz
Robotics Institute
Carnegie Mellon University
tony@cmu.edu

***Abstract*—The use of spoken instructions in human-robot teams holds the promise of enabling untrained users to effectively control complex robotic systems in a natural and intuitive way. Providing robots with the capability to understand natural language directions would enable effortless coordination in human robot teams that operate in non-specialized unknown environments. However, natural language direction following through unknown environments requires understanding the meaning of language, using a partial semantic world model to generate actions in the world, and reasoning about the environment and landmarks that have not yet been detected.**

We address the problem of robots following natural language directions through complex unknown environments. By exploiting the structure of spatial language, we can frame direction following as a problem of sequential decision making under uncertainty. We learn a policy which predicts a sequence of actions that follow the directions by exploring the environment and discovering landmarks, backtracking when necessary, and explicitly declaring when it has reached the destination. We use imitation learning to train the policy, using demonstrations of people following directions. By training explicitly in unknown environments, we can generalize to situations that have not been encountered previously.

"Walk around to the other side of the staircase and go down the hall past the mailboxes. The room is on the left after the mailboxes."

(a) Natural Language Command.

(b) Environment.

Fig. 1. Our goal is to take a command expressed in natural language (e.g., 1a) and generate a sequence of actions through a partially known environment (such as the one shown in 1b).

I. Introduction

As robots move out of the lab and into daily life, people and robots are increasingly working together in shared spaces, with common tasks and goals. These robots are beginning to perform complex tasks, but the ability of untrained people to program these robots has been limited. To address this issue, new human-robot interaction modalities must be developed to enable lay users to command robots. Natural language holds the promise of enabling people to compose and convey complex behaviors in an intuitive and flexible way, without requiring programming knowledge, specialized interfaces, or extensive training.

This paper addresses the problem of enabling robots to understand task-constrained natural language directions, which requires leveraging the structure of language, mapping from commands onto actions, recognizing diverse objects, and reasoning about a potentially unknown goal. For example, a first responder may want to command a rescue robot using the language shown in Figure 1. Such commands are challenging because they include understanding diverse landmarks (e.g., "the staircase" or "mailboxes") and both actions and spatial relationships (e.g., "around," "go down," "past").

Although previous work [1], [2] has addressed many of the above issues, they assumed a fully known semantic map of the environment. This paper moves beyond these to address partially known environments, which are more challenging for three primary reasons that we will illustrate using Figure 2. Firstly, the robot must take actions or reason about landmarks that may not have realizations at the start of execution (e.g., the elevators are not visible in Figure 2c). Secondly, the robot must recover from failures after taking a wrong action (the robot must recover from taking the wrong "right turn" in Figure 2d). Lastly, the robot must decide explicitly when to stop following the current direction (as shown in Figure 2e). These challenges all require reasoning about uncertainty and the parts of the environment we have not observed.

This paper addresses the problem of learning the meaning for words and phrases in natural language from human demonstrations of correct behavior (e.g., grounded language acquisition). Specifically, we frame understanding natural language route directions as a sequential decision making under uncertainty problem, where the aim is to learn a policy

that maps from the natural language command and a partial map of the environment onto a sequence of actions that best obey the command.

We will describe in Section III our approach to tractably learn and execute the policy. First, we break down the natural language command into linguistic clauses called Spatial Description Clauses (SDCs) [1], where each SDC corresponds to an action that the robot should execute and a landmark that the robot should see. After describing our model for partially known environments, we will introduce our feature representation and the details of our policy. We then describe how we can learn the policy using imitation learning in Section IV, given demonstrations of people following directions. By training the policy on a partial semantic map, our approach is able to explore partially known environments in an informed manner, and learns to backtrack if it makes an error.

To evaluate our approach, we have collected a dataset of natural language commands. The task is for the robot to navigate through one floor of an indoor building using only partial map information. Our approach is able to follow 79% of directions from a held out set of directions. We show in Section V that successfully following natural language directions through unknown environments requires components that model verbs and spatial relations, identify (approximately) what the entire action should look like, and reason about landmarks that are likely to be similar to the referring expressions used in natural language.

II. Related Work

There exist several approaches to natural language direction following. MARCO was a system which used a hand coded language parser along with engineered procedures which executed actions through a simulated environment [3]. Chen and Mooney extended this work by learning a language parser which extracted semantic structure from the language and mapped it to actions in a fully known environment [4]. Similarly, Matuszek and colleagues learned a language parser that could follow directions through an unknown indoor environment [5]. These two approaches learn a mapping from natural language to an unambiguous grammar which can be directly mapped to actions in the world. In our work the semantic structure must still be grounded in the environment, as we are learning the meaning of commands directly.

The approaches presented by Kollar, Tellex, and colleagues [1], [6] learn groundings for language, but assume a complete semantically labeled map is available. Although we share their semantic descriptions (to represent the structure of spatial language) and some of their spatial and linguistic features (to characterize relationships between paths and landmarks), we frame direction following using a fundamentally different approach. Our system is explicitly designed to operate in unknown environments. We represent actions instead of complete paths, which requires reasoning about uncertainty. By training in unknown environments, we learn a policy which can recover from mistakes by backtracking.

We draw from work in Imitation Learning [7], specifically maximum margin classifiers that predict a sequence of actions [8]. We use DAGGER, an imitation learning framework which minimizes the error on the distribution of states induced by the learned policy (instead of the distribution of states visited by the expert) [9]. Our learning approach is also similar to search-based structured prediction [10] and Learning to Search [11], which learn classifiers and planners (respectively) parameterized by cost functions.

Reinforcement learning has been applied to train a policy to follow natural language directions for a GUI interaction task [12], as well as navigation in the MAP-TASK corpus [13]. The policy in these approaches also map states to actions, but our work learns the policy using imitation from expert demonstrations of correct behavior.

Our work is related to exploration strategies used for Simultaneous Localization and Mapping [14], as we must trade off the cost of executing actions with the cost of gathering information. However, instead of trying to reduce the uncertainty in the map or the robot's pose, we are trying to maximize the probability of correctly following the directions while exploring the environment only as much as necessary.

III. Approach

Direction following is treated as a sequential decision making process, where a policy π predicts a sequence of actions that take the robot to a destination. Our approach uses a class of policies that are linear the features ϕ of the state and action pair, and considers all available actions ($a \in A_s$) from state s, returning the one which minimizes a weighted sum of features:

$$\pi(s) = \underset{a \in A_s}{\operatorname{argmin}} \, w^T \phi(s, a) \tag{1}$$

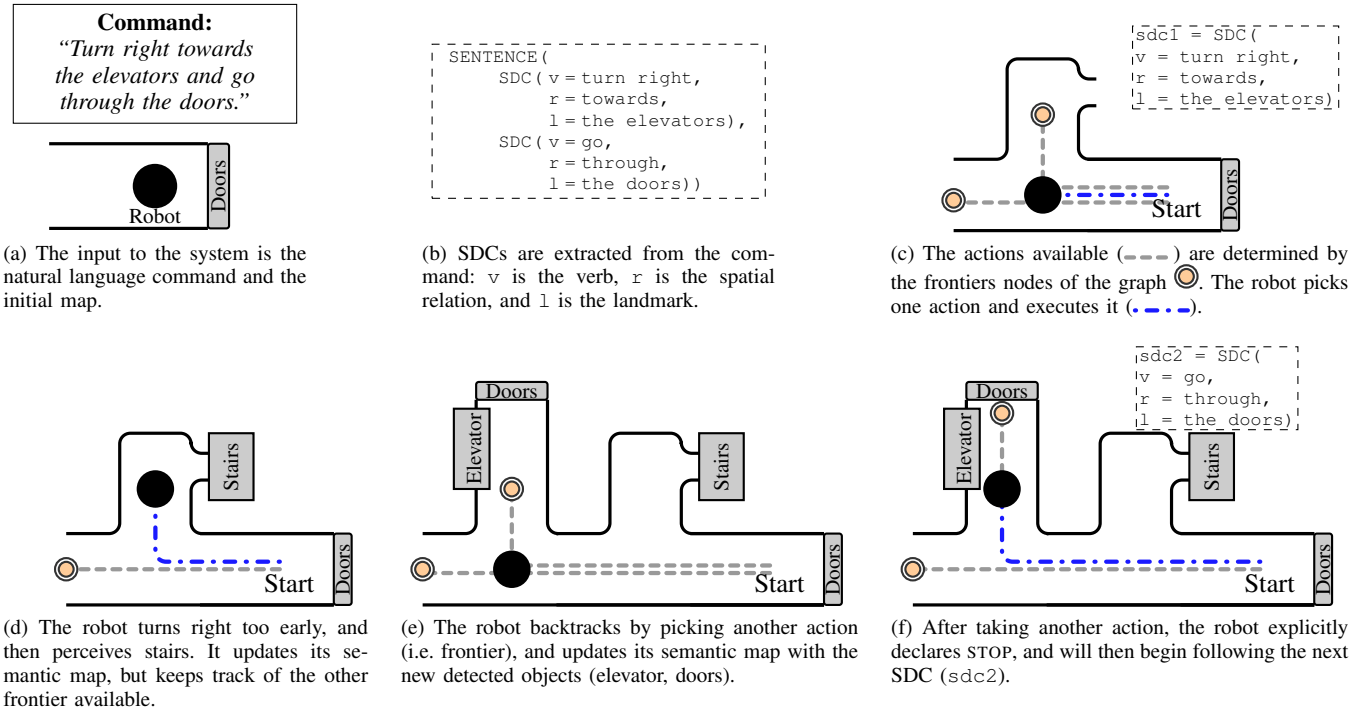
By decomposing the state s into the language Z and the semantic map M, the policy (specified by weights w) can be re-written as follows:

$$\pi(s) = \underset{a \in A_s}{\operatorname{argmin}} \, w^T \phi(Z, M, a) \tag{2}$$

In Section III-A we describe our model for the natural language command Z, which is split into a sequence of semantic clauses. Section III-B describes the partially known semantic map M, which contains semantically labeled objects and a graph which specifies the actions available at each state. Section III-C describes the features ϕ used for training the policy, and Section III-D brings these together to describe how the robot follows natural language directions through unknown environments using the policy.

A. Modeling Spatial Language

In order to understand natural language direction Z, we leverage the properties of spatial language. Firstly, directions are sequential: each clause in the directions refers to one step, ordered from the start to the destination. Secondly, spatial language decomposes into several components: verbs, spatial relations, and landmarks. Landmarks are aspects of the environment that are visible from the path. Verbs prescribe



(a) The input to the system is the natural language command and the initial map.

(b) SDCs are extracted from the command: `v` is the verb, `r` is the spatial relation, and `l` is the landmark.

(c) The actions available (- - -) are determined by the frontiers nodes of the graph ◎. The robot picks one action and executes it (-·-·-).

(d) The robot turns right too early, and then perceives stairs. It updates its semantic map, but keeps track of the other frontier available.

(e) The robot backtracks by picking another action (i.e. frontier), and updates its semantic map with the new detected objects (elevator, doors).

(f) After taking another action, the robot explicitly declares STOP, and will then begin following the next SDC (`sdc2`).

Fig. 2. Illustration of using a policy to follow natural language directions. The command (2a) is parsed into a sequence of Spatial Description Clauses (2b), which can be understood by the robot. As the robot moves in the environment, new landmarks are perceived and the semantic map is updated (2c-2d). Actions are paths to frontier nodes (which lie at the edge of explored space), and may represent a backtracking action (e.g., 2e). Furthermore, at each decision point a separate STOP action (not shown) is available, which transitions to the next SDC if it exists (2f) or declares the goal has been been reached otherwise.

what to do and where to go (e.g., "turn right"). Spatial relations such as "past" and "towards" describe the relative geometry between the path and the landmark.

We formalized this structure by modeling each sentence as a sequence of structured clauses called Spatial Description Clauses (SDCs) [1]:

$$Z \rightarrow [\mathrm{sdc}_0, \ldots, \mathrm{sdc}_N] \tag{3}$$

Each SDC consists of a *figure* (the subject of the sentence), a *verb* (an action to take), a *landmark* (an object in the environment), and a *spatial relation* (a geometric relation between the landmark and the figure). Any of these fields can be unlexicalized and therefore only specified implicitly. Figure 2b shows the sequential SDC representation for an example sentence. Because we model directions sequentially, we can treat direction following as a sequence of SDC-following procedures, which decomposes into features of the *current* SDC, the semantic map, and the action.

There are some sentences that do not fit into this representation (for example nested clauses). However, SDCs do capture the important semantics of the language, and the sequential representation is easy to extract and efficient for use in direction following.

B. Modeling Partially Known Environments

To represent the partially known environment, our approach uses a combined topological/metric representation which is built online as the robot moves through the world. A graph $G = (V, E)$, contains vertices $v \in V$ (representing viewpoints) that are connected by edges $e \in E$ which represent allowable robot travel segments. The graph is created incrementally by computing frontier nodes, which lie between explored and unexplored space. As the robot moves to a frontier node, it can see into unexplored space, which "pushes forward" the frontiers, and adds new information to the map used for following directions [15].

As the robot navigates in the environment, it also builds a set of semantically labeled objects $\mathcal{O}$, which consists of all landmarks that have been previously detected. These objects $o \in \mathcal{O}$ can be used for navigation after they have been classified by a perception system, which semantically labels them with a name (in addition to their geometry). Taken together, this information (environment graph and known objects) forms our semantic map M which gets updated during navigation as the robot gains more information about the environment:

$$M = \{V, E, \mathcal{O}\} \tag{4}$$

Both the semantic map M and the command Z are contained in the state s. For any state s, the available actions A_s are paths in the graph paired with a landmark from the known object set. We only consider paths which terminate at a current frontier node to restrict the action space and bias towards unknown parts of the map. The object may be null, for example when no landmark is specified or if no suitable landmark is detected. Intuitively, an action represents one step along the direction's path. However, since each action reveals new parts of the environment, they may be exploratory (for

example traveling down a hallway to an intersection), or may represent a backtracking action (by going to a different part of the environment). A separate STOP action moves to the next SDC in the sequence if it exists, or declares the goal has been reached otherwise.

The sequential decomposition (the sequence of SDCs) and the fact that each SDC is self contained (e.g., it is independent of the previous and next SDC) creates the following approximation, where the policy is indexed by each SDC. This decomposition paired with the action model described above yields the following policy which will be used during direction following through unknown environments:

$$\pi_i(s) = \underset{a \in A_s}{\operatorname{argmin}} \, w^T \phi(\mathrm{sdc}_i, M, a) \tag{5}$$
$$A_s = \{path \in G \cup \mathrm{STOP}\} \times \mathcal{O}$$

Here, the action set A_s includes paths in the graph to frontier nodes, in addition to the STOP action.

C. Feature Representation

For a given action $a \in A_s$, the features are computed for a path from the start of the current SDC to the frontier node that is being evaluated, without taking previous (potentially backtracking) actions into account. This is to prevent backtracking actions from obscuring the meaningful features of the action we wish to evaluate. For example in Figure 2, the robot first goes into a hallway, backtracks, and then finally takes the correct right turn. In Figure 2e we are interested in features of the final right turn path without backtracking, since features computed over the entire history (including backtracking) would lose their meaning with respect to the direction being followed.

The feature vector $\phi(s, a)$ contains several types of features. Geometric features describe the shape of the path, the geometry of the landmark, and the relationship between the two [6], [16]. Linguistic features express a similarity metric between the landmark field in the SDC and the object in the world, and utilize WordNet (a lexical database of English words) and a database of tagged images (extracted from Flickr) to generalize across various landmark names [17]. We also use the Cartesian product of the geometric and linguistic features to represent paths and objects that occur together (for example, turning right *and* seeing the elevator in Figure 2e) instead of just matching the action *or* the landmark.

As we explicitly represent STOP actions, it is important to be able to compare the features of a completed path with its expected shape. We thus compute the distance (in the space of geometric features) of any STOP action with average canonical paths. The features for various canonical paths (right turns, left turns, ...) are computed by averaging features of multiple paths in the corpus. These features only apply when we stop, and ensure that the complete path (as opposed to one individual action) is representative of the direction as a whole before declaring that the goal has been reached.

1: **procedure** FOLLOW DIR($\pi, Z, v_{\mathrm{start}}$)
2: $V = \{v_{\mathrm{start}}\}, E = \emptyset$ ▷ Unknown environment
3: $\mathcal{O} = \emptyset$ ▷ No known objects
4: $M \leftarrow \{V, E, \mathcal{O}\}$ ▷ Semantic map
5: $Z \rightarrow [\mathrm{sdc}_0, \ldots, \mathrm{sdc}_N]$ ▷ Extract SDCs from Z
6: $\mathrm{cur\_sdc} \leftarrow 0$ ▷ Begin with the first SDC
7: $v_{\mathrm{cur}} \leftarrow v_{\mathrm{start}}$
8: **repeat**
▷ Update semantic map (graph and objects):
9: $M \leftarrow$ PERCEIVE_WORLD(M)
10: $a \leftarrow \pi_{\mathrm{cur\_sdc}}(s)$ ▷ Picks action from A_s
11: $v_{\mathrm{cur}} \leftarrow$ MOVE(a)
12: **if** $a ==$ STOP **then**
13: $\mathrm{cur\_sdc} \leftarrow \mathrm{cur\_sdc} + 1$ ▷ Follow next SDC
14: **end if**
15: **until** $\mathrm{cur\_sdc} > N$
16: **return** v_{cur}
17: **end procedure**

Fig. 3. Algorithm for following the directions represented by a sequence of SDCs. We begin with a graph consisting only of our current location (v_{start}), and no known objects. We then perceive the world in line 9, updating the graph and set of landmarks. We pick an action for the current SDC, and move in the world. If the action is a stop action (line 12), we move to the next SDC. Once all SDCs have been followed to completion (line 15), we have finished following the complete directions.

D. Policy Execution through Unknown Environments

Given a policy $\pi(s)$ and a natural language command Z, we can follow directions by making a sequence of decisions (actions in the environment). This algorithm in shown in Figure 3. We initialize our semantic map M and decompose the directions into a sequence of SDCs. Then, beginning with the first SDC, we iterate between observing the world (using the robot's onboard perception to update M) and applying the action specified by the policy in Equation 5. When a STOP action is returned we transition to the next SDC, or declare that we have reached our destination if we are following the last SDC.

IV. Imitation Learning Formulation

The policy is trained using imitation learning, by treating action prediction as multi-class classification problem where we wish to correctly predict the expert's action (out of all possible actions). Given traces of people following directions, we learn a policy on single-SDC segments. Unique to our approach is the fact that we train *explicitly* in unknown environments, thus learning a policy which directly takes uncertainty into consideration. At a high level, we learn a policy by iteratively applying the current policy and applying corrections based on the expert demonstrations.

We assume that the expert's policy π^* minimizes the unknown immediate cost $C(s, a^*)$ of performing action a^* from state s. Since we do not directly observe the true costs of the expert's policy, we must instead minimize a surrogate loss function which will penalize disagreements between the expert's observed action $\pi^*(s)$ and our action $\pi(s)$. Since our policy picks a different action $a \neq a^*$ only if

procedure TRAIN_POLICY(Z_{train}, N)
  Initialize π to any policy in Π parameterized by w
  for t = 1 to N **do**
    Sample a trajectory $\{s, \pi(s)\}$ by applying current policy to all directions.
    Compute expert's actions at states visited by the current policy: $a^* = \pi^*(s)$
    Minimize $\ell(s, a^*, w)$ over all examples by computing $\partial\ell/\partial w$.
    Update policy according to Equation 11.
  end for
  return π
end procedure

Fig. 4. Dataset Aggregation for direction following. Each iteration applies the policy to the all training directions Z_{train} using FOLLOW_DIR (Figure 3). We then compute the expert's demonstrated actions for all states visited by the policy, and compute a gradient using Equation 9.

its cost $w^T\phi(s,a)$ is lower than the cost of the expert's action (see Equation 1), we treat this as a multi-class prediction problem, where disagreements are penalized using the multi-class hinge loss [18]:

$$\ell(s,a^*,w) = \max\left(0, 1 + w^T\phi(s,a^*) - \min_{a\neq a^*}\left[w^T\phi(s,a)\right]\right) \quad (6)$$

The loss in Equation 6 is zero when the cost of the expert's action is lower than the cost of all other actions with a margin of one. If the cost of the expert's action is more than the cost of another action (again by a margin), the loss is positive and follows the well-known multiclass SVM loss. This loss can be rewritten as:

$$\ell(s,a^*,w) = w^T\phi(s,a^*) - \min_{a}\left[w^T\phi(s,a) - l_{sa}\right] \quad (7)$$

where $l_{sa} = 0$ if $a = a^*$ and 1 otherwise. This ensures that the expert's action is better than all other actions by a margin [8]. Adding a regularization term λ to Equation 7 yields our complete optimization loss:

$$\ell(s,a^*,w) = \frac{\lambda}{2}\|w\|^2 + w^T\phi(s,a^*) - \min_{a\in A_s}\left[w^T\phi(s,a) - l_{sa}\right] \quad (8)$$

Although this loss function is convex, it is not differentiable. However, we can optimize it efficiently by taking the subgradient of Equation 8 and computing action predictions for the loss-augmented policy [8]:

$$\frac{\partial\ell}{\partial w} = \lambda w + \phi(s,a^*) - \phi(s,a'), \quad (9)$$

for the best loss-augmented action a' at state s:

$$a' = \operatorname*{argmin}_{a\in A_s}\left[w^T\phi(s,a) - l_{sa}\right]. \quad (10)$$

Note that a' is simply the solution to our policy using a loss-augmented cost. This leads to the update rule for w:

$$w_{t+1} \leftarrow w_t - \alpha\frac{\partial\ell}{\partial w} \quad (11)$$

with a learning rate $\alpha \propto 1/t^\gamma$. Intuitively, this update decreases the cost associated with features of the expert's action a^*, and increases the cost associated with features of the predicted action a'. If the expert's action matches the policy's, the gradient will be zero and the weights will not be updated (as we desire).

While minimizing Equation 8 leads to a good action prediction from state s, it is important to note that optimizing this loss *only* over the states visited by the expert may not lead to good test-time accuracy, since the learned policy's predictions will affect future states and observations, which violates the i.i.d. assumption made by most imitation learning approaches. To give a concrete example in our setting, a small mistake (picking the wrong door to go through) leads to a completely different distribution of states (a different room/hallway with different landmarks), which the learning algorithm has not seen during training (and thus cannot decide how to recover).

To remedy this problem, we apply DAGGER (Dataset Aggregation) [9], which learns a policy by iterating between collecting data (using the current policy) and applying expert corrections to the decisions that were made (using the expert's demonstrated policy). Using this method, we learn a policy that does well on the distribution of states induced by the learned policy, instead of only the distribution of states visited by the expert. Figure 4 describes how we collect data to train our policy. This approach to learning a policy is simple, elegant, and requires no complex engineering of components or tuning of parameters.

V. EXPERIMENTS AND RESULTS

We have evaluated our approach on a map of one floor in an indoor building introduced by Kollar et al. [1], which contains an occupancy grid and semantically labeled objects. A corpus of 30 directions was created for this problem. Each direction consists of multiple SDCs (2.5 on average), and travels a large distance through the map (28.8 m on average). The directions include multiple verbs ("turn right," "go straight," ...), spatial relations ("towards," "past"), as well as a variety of landmarks. Some of the landmarks referenced in the direction are not directly contained in the environment map; for example a "couch" or "seat" in the directions may refer to a "sofa" in the environment.

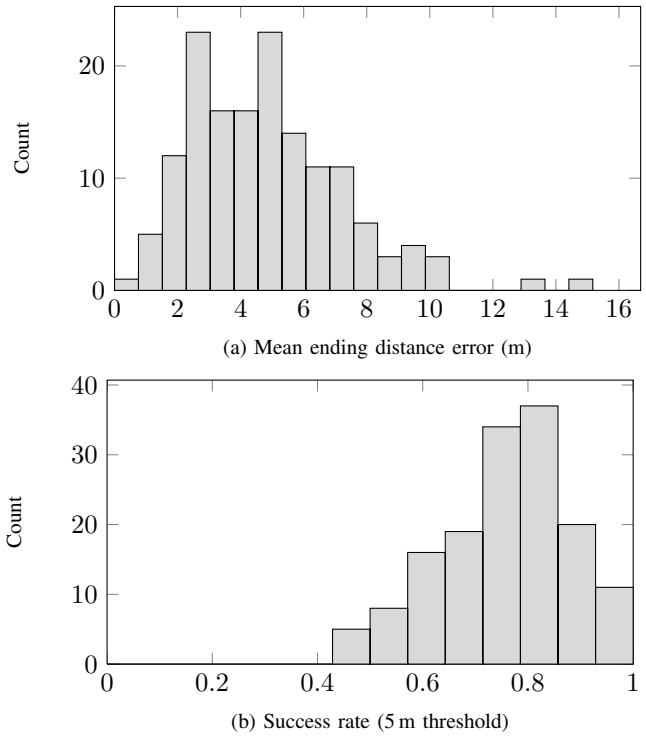
Our policy currently operates in simulation, so we have abstracted perception to track only objects which are visible (connected by line-of-sight and within some maximum distance threshold). Training in a simulated environment (while still observing all visibility constraints) enables us to learn a policy efficiently, as running multiple iterations in the real world would be time consuming. In the following experiments, we trained the policy using $N = 25$ iterations of DAGGER.

We first evaluated various configurations of our approach on a test set of directions, retraining for each configuration. We split the corpus so that directions starting to the left of a particular point were in the training set, and the rest of the directions were held out for a test set, which resulted in 16 training directions and 14 test directions. The results shown in Table I include the mean distance error (over

TABLE I
VALIDATION RESULTS ON HELD OUT SET OF DIRECTIONS.

Configuration	Mean error	Success rate*
Complete trained policy	2.39 m	79 %
Full semantic map	3.35 m	85 %
No semantic sim. feats.	4.44 m	64 %
No stop action feats.	7.15 m	50 %
No verbs or spatial rels.	16.56 m	36 %
Supervised learning†	7.38 m	71 %
Random destination‡	32.79 m	5 %

* Percentage of directions which finish within 5 m of the destination.
† Training only on states visited by expert demonstration.
‡ Expected results for a random vertex (assuming complete graph knowledge).



(a) Mean ending distance error (m)

(b) Success rate (5 m threshold)

Fig. 5. Histograms of results (mean ending distance error and success rate) for 150 cross-validation trials using the complete trained policy.

all test directions) as well as the percentage of directions which ended within 5 m of the intended destination (our success metric). Over the test directions, our trained policy correctly follows 79 % of the directions with an average ending distance from the true destination of 2.39 m.

Interestingly, providing the full semantic map (i.e. removing the visibility constraints) had mixed (but slight) impact on performance, increasing both distance error and success rate. We believe this is because there is a disconnect between having to reason about local actions (incremental steps on the current SDC) while being able to "use" objects that would otherwise be invisible. In some sense, the visibility constraints makes the policy's decision easier by only allowing a few landmarks to be considered at a time.

Removing linguistic features that relate semantic similarity of landmarks decreases performance, since the policy can not generalize across different landmark names. The stop action features are a significant component, as without them the policy cannot compare the final action with a canonical representation of the desired path, and stops too early. Performance decreases most drastically if we remove the verbs/spatial relations, as the policy can no longer differentiate between the various commanded actions. Overall, these results demonstrate that direction following requires an understanding of actions (verbs), semantic similarity (landmarks), and an explicit way to reason about stopping.

For comparison, we trained the policy using a traditional supervised learning approach, by learning a policy using only the states visited by the expert's demonstration. Although this policy reached the goal 71 % of the time, the mean error was much higher due to some very large errors in a few directions where un-recovered mistakes occurred. This demonstrates the ability of our approach to learn a policy which generalizes to new directions and can backtrack when the current action does not match the direction being followed.

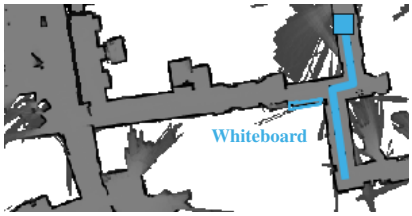
We also performed a cross validation experiment, where we randomly split the corpus into the same number of testing and training directions as above, then learned a policy and evaluated it on the test set. The results shown in Figure 5 present the overall average ending distance and success rate for 150 trials. One particular held out test direction is shown in Figure 6, showing how partial information is used when available. The command ("*turn right to the whiteboard, go past the water fountain, go to the couch.*") is followed as best as possible using landmarks as they are detected. The policy initially makes an error in the first SDC, but backtracks and recovers to follow the direction correctly.

VI. CONCLUSIONS AND FUTURE WORK

This work enables robots to autonomously follow natural language directions through unknown environments. By exploiting properties of spatial language, we are able to extract a sequence of semantic structures (Spatial Description Clauses) which encode the information necessary for direction following. We have framed direction following as a sequential decision making under uncertainty problem, which enables us to learn a policy that explores the environment, backtracks when necessary, and explicitly declares when the destination has been reached. We learn this policy using imitation learning, given demonstrations of people following directions. Our results demonstrate that natural language direction following through unknown environments requires understanding verbs, detecting similar landmarks in the environment, and reasoning explicitly about stopping.

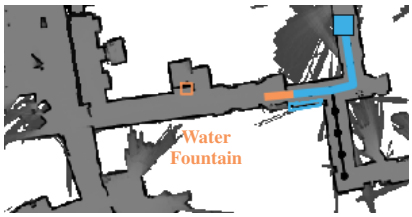
This work is one step towards intuitive command of complex robotic systems. Future work will include evaluating our approach on a larger corpus of directions, and

(a) Starting at the blue square, the policy first goes straight and explores to find a whiteboard.

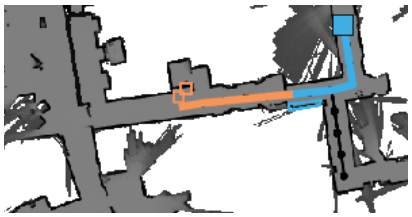


(b) Policy finds the whiteboard but makes a wrong turn.

(c) Policy backtracks to the correct hallway, calls the STOP action, and moves to the next SDC.



(d) Following the second SDC. A water fountain is perceived, and the policy goes towards it.



(e) A second water fountain is perceived. The policy considers the current SDC to be completed, calls STOP, and transitions to the third SDC.

(f) A sofa is found, and is associated with the "couch" from the SDC's landmark field. The policy once again calls STOP, and is now done following the directions.

Fig. 6. Sequence of decisions for following the direction "*turn right to the whiteboard, go past the water fountain, go to the couch.*" The actions and landmarks associated with each SDC are color-coded. The policy explores (looking for landmarks), backtracks if it makes an error (as in 6c), and transitions between SDCs using the STOP action. Not all decisions are shown.

applying it to a robot operating autonomously in an indoor environment. Reasoning in belief space and transitioning probabilistically between actions would improve the direction following policy, by enabling us to backtrack *between* SDCs and reason about uncertainty over the entire direction which would prevent small mistakes from compounding.

To resolve more complex forms of ambiguity and missing information (e.g., incorrect directions, missing landmarks, incomplete perception, etc…), we will evaluate using *dialogue* to "close the loop" on direction following. By asking questions, robots can begin to learn recovery strategies for dealing with these (and other) types of errors.

ACKNOWLEDGMENTS

The authors gratefully acknowledge Alexander Grubb and Stefanie Tellex for their contributions to our early direction following work and providing access to the spatial features library. We have appreciated valuable discussions with Stéphane Ross and Drew Bagnell, as well as the comments from the anonymous reviewers. This work was supported in part by ONR under MURI grant "Reasoning in Reduced Information Spaces" (no. N00014-09-1-1052) and by the National Science Foundation under a Graduate Research Fellowship and grant IIS-1218932.

REFERENCES

[1] T. Kollar, S. Tellex, D. Roy, and N. Roy, "Toward Understanding Natural Language Directions," in *International Conference on Human-Robot Interaction*, 2010.

[2] T. Kollar, "Learning to Understand Spatial Language for Robotic Navigation and Mobile Manipulation," Ph.D. dissertation, Massachusetts Institute of Technology, 2011.

[3] M. MacMahon, B. Stankiewicz, and B. Kuipers, "Walk the Talk: Connecting Language, Knowledge, and Action in Route Instructions," in *National Conference on Artificial Intelligence*, 2006.

[4] D. L. Chen and R. J. Mooney, "Learning to Interpret Natural Language Navigation Instructions from Observations," in *AAAI*, 2011.

[5] C. Matuszek, E. Herbst, L. Zettlemoyer, and D. Fox, "Learning to Parse Natural Language Commands to a Robot Control System," in *International Symposium on Experimental Robotics*, 2012.

[6] S. Tellex, T. Kollar, S. Dickerson, M. R. Walter, A. G. Banerjee, S. Teller, and N. Roy, "Understanding Natural Language Commands for Robotic Navigation and Mobile Manipulation," in *National Conference on Articial Intelligence*, 2011.

[7] B. D. Argall, S. Chernova, M. Veloso, and B. Browning, "A survey of robot learning from demonstration," *Robotics and Autonomous Systems*, 2009.

[8] N. D. Ratliff, J. A. Bagnell, and M. A. Zinkevich, "Maximum Margin Planning," in *International Conference on Machine Learning*, 2006.

[9] S. Ross, G. J. Gordon, and J. A. Bagnell, "A Reduction of Imitation Learning and Structured Prediction to No-Regret Online Learning," in *International Conference on Artificial Intelligence and Statistics*, 2011.

[10] H. Daumé, J. Langford, and D. Marcu, "Search-based structured prediction," *Machine Learning*, 2009.

[11] N. D. Ratliff, D. Silver, and J. A. Bagnell, "Learning to Search: Functional Gradient Techniques for Imitation Learning," *Autonomous Robots*, 2009.

[12] S. R. K. Branavan, H. Chen, L. Zettlemoyer, and R. Barzilay, "Reinforcement Learning for Mapping Instructions to Actions," *ACL-IJCNLP*, 2009.

[13] A. Vogel and D. Jurafsky, "Learning to Follow Navigational Directions," *ACL*, 2010.

[14] C. Stachniss, G. Grisetti, and W. Burgard, "Information Gain-based Exploration Using Rao-Blackwellized Particle Filters," in *Robotics: Science and Systems*, 2005.

[15] B. Yamauchi, "Frontier-Based Exploration Using Multiple Robots," in *International Conference on Autonomous Agents*, 1998.

[16] S. Tellex, "Natural Language and Spatial Reasoning," Ph.D. dissertation, Massachusetts Institute of Technology, 2010.

[17] T. Kollar and N. Roy, "Utilizing object-object and object-scene context when planning to find things," in *International Conference on Robotics and Automation*, 2009.

[18] K. Crammer and Y. Singer, "On the Algorithmic Implementation of Multiclass Kernel-based Vector Machines," *Journal of Machine Learning Research*, 2002.