Grounding Spatial Relations for Outdoor Robot Navigation

Abdeslam Boularias, Felix Duvallet, Jean Oh and Anthony Stentz[1]

***Abstract*— We propose a language-driven navigation approach for commanding mobile robots in outdoor environments. We consider unknown environments that contain previously unseen objects. The proposed approach aims at making interactions in human-robot teams natural. Robots receive from human teammates commands in natural language, such as "Navigate around the building to the car left of the fire hydrant and near the tree". A robot needs first to classify its surrounding objects into categories, using images obtained from its sensors. The result of this classification is a map of the environment, where each object is given a list of semantic labels, such as "tree" and "car", with varying degrees of confidence. Then, the robot needs to ground the nouns in the command. Grounding, the main focus of this paper, is mapping each noun in the command into a physical object in the environment. We use a probabilistic model for interpreting the spatial relations, such as "left of" and "near". The model is learned from examples provided by humans. For each noun in the command, a distribution on the objects in the environment is computed by combining spatial constraints with a prior given as the semantic classifier's confidence values. The robot needs also to ground the navigation mode specified in the command, such as "navigate quickly" and "navigate covertly", as a cost map. The cost map is also learned from examples, using Inverse Optimal Control (IOC). The cost map and the grounded goal are used to generate a path for the robot. This approach is evaluated on a robot in a real-world environment. Our experiments clearly show that the proposed approach is efficient for commanding outdoor robots.**

I. INTRODUCTION

We consider the problem of commanding mobile robots in unknown, semi-structured, outdoor environments using natural language. This problem arises in human-robot teams, where natural language is a favored communication means. Therefore, robots need to understand the environment from the standpoint of their human teammates, and to translate instructions received in natural language into plans.

For example, to execute the command "Navigate around the building to the car that is left of the fire hydrant and near the tree", the robot needs to find out which objects in the environment are meant by "building" and "car", and to plan a path accordingly. To accomplish this goal, the robot needs to ground all the nouns in the command ("building", "car", "fire hydrant" and "tree") into specific objects in the environment, and to interpret the spatial relations ("left of" and "near") and the navigation mode ("around").

Grounding nouns as physical objects is a high-level skill that requires a combination of different cognitive capabilities. The first one is parsing command sentences, and

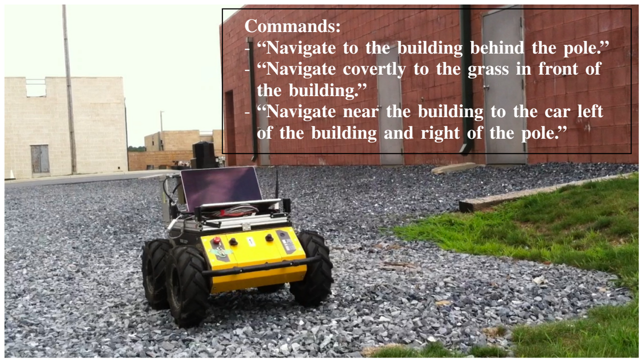

Commands:
- "Navigate to the building behind the pole."
- "Navigate covertly to the grass in front of the building."
- "Navigate near the building to the car left of the building and right of the pole."

Fig. 1: Clearpath™ Husky robot in an unknown environment

transforming them into structures that can be given to the grounding algorithm. Parsing and part-of-speech tagging is a fundamental problem in computational linguistics, and it is still a highly challenging one. Since this problem is not the main focus of this work, we assume that command sentences are generated using a restrained language called *Tactical Behavior Specification* (TBS). TBS is a language for specifying action commands and spatial constraints associated with the action. The grammar of the TBS language in the Backus-Naur Form (BNF) is given in Figure 2.

The robot must also be able to recognize the objects in the environment and to label them. We use the semantic perception method proposed in [1] which has been proven effective in outdoor environments [2]. The semantic perception module receives scene images from a 2D camera and classifies each pixel into categories with different confidence values. Pixels that belong to the same object are clustered together by using a 3D LADAR image of the same scene.

Grounding is performed by combining the labels obtained from semantic perception with the spatial relations obtained from parsing the command. There are three main sources of uncertainty that make grounding a challenging task. First, objects are often misclassified because of occlusions and noise in the sensory input. Classification errors also occur when the environment contains objects that are significantly different from the ones used for training the classifier. Second, the commands can be ambiguous, i.e. multiple objects satisfy the constraints in a given command. Third, spatial relations are often subjectively interpreted. People have different views on what "left of a building" is, for example. Some people would define it relatively to their current position, others would look for the main entrance of a building to define its left and right sides. Moreover, humans disagree on the correct term to use for describing the location of an object, if it is both on left and in front of a building for instance. Concepts such as

The authors are with the Robotics Institute of Carnegie Mellon University, Pittsburgh, PA 15213 USA. {abdeslam, fduvalle, jeanoh, tony}@andrew.cmu.edu

near and far are also very subjective. Therefore, one should not rely on fixed deterministic rules for grounding spatial relations, especially given the uncertainty in the labels of the objects. To illustrate this point, consider the following simple example. The robot is commanded to navigate to "the car behind the tree". The robot detects a tree and two objects. One of them is almost certainly a car, but is not exactly behind the tree and rather far from it. The other one is perfectly behind the tree, but the robot is not certain about its label. Which object should the robot choose as a goal?

To trade off these uncertainties, we use a Bayesian model for grounding. Our approach is based on using the confidence values of the perception as a prior distribution on the true category of each object. A posterior joint distribution on the objects is computed based on how well each object satisfies the spatial constraints. A key component of this model is a function that maps two objects and a spatial relation into a probability. This function is learned from annotated examples. We also learn a function that maps a navigation mode into a cost map for path planning, using Inverse Optimal Control (IOC) [3]. Learning by imitation enables the robot to interpret commands according to the subjective definitions of its human user. Moreover, navigation modes as "navigate covertly" do not have clear definitions that can be used for handcrafting a path cost function.

Finally, we compute a joint distribution on goal objects and on landmark objects used for specifying a navigation mode, so that results with small path costs have high probabilities. For path planning, we use the cost map based planner, PMAP with Field D* [4]–[7]. The plan of the grounding result with the highest probability is executed by the robot.

This paper describes the grounding module of our language-driven navigation system. We start by reviewing related work in Section II. We then present the TBS grammar in Section III, the IOC approach for learning navigation modes in Section IV, and the probabilistic approach for grounding nouns in Section V. Section VI presents experiments in simulation and using a real robot. The final section concludes this paper with a summary of our findings.

II. RELATED WORK

The challenge of building human-robot interfaces using natural language generated a large body of work [8]–[19]. A full review of the related works is beyond the scope of this paper, so we highlight here some relevant examples. Symbol grounding was first formulated in [8] as the problem of mapping words (symbols) into manifestations in the physical world. A framework for following verbal route instructions was proposed in [9]. The proposed grounding algorithm translates a high-level action, such as "turn to face the building", into a sequence of implicit low-level actions using fixed rules. The same problem was addressed in [10]. We solve a similar problem in this paper by learning cost functions for different navigation modes. An algorithm for generating and resolving referring spatial expressions was presented in [11]. The proposed algorithm was based on a knowledge base and rule inference, in contrast to our probabilistic approach. First-order dynamic logic was used in [12] for grounding goal and action utterances. Golland et al. [13] described a game-theoretic approach for dialogue management, wherein spatial relations were learned from annotated examples. Using virtual examples, the authors showed that learned relation models are generally beneficial. Our grounding approach is closely related to [13]. However, we take into account label uncertainty and path cost in grounding, contrary to [13]. The Generalized Grounding Graphs (G^3) [14] is a generic framework that casts symbol grounding as a learning and inference problem in a Conditional Random Field. G^3 is an efficient graph factorization technique for grounding all the words in a given sentence. Our approach is tailored for grounding only navigation modes and spatial relations, in BNF, while taking into consideration perception errors. Note, however, that we use the same type of spatial relation clauses as the ones presented in [15] and used in [14]. A human-robot dialogue system based on the G^3 model was presented in [16]. The navigation system described in [17], also based on G^3, incorporates odometry and path constraints in grounding, which is conceptually comparable to our use of perception confidence and path costs in grounding. Matuszek et al. [18] showed that a joint model of language and perception for grounding can be learned simultaneously, although the considered spatial relations were simple. Guadarrama et al. [19] presented a system for human-robot interaction that also learns both models for spatial prepositions and for object recognition. However, simple relations were considered and perception uncertainty was not taken into account.

Finally, note that the grounding approach presented in this paper was briefly explained in [20], where other components of the complete robotic system were presented.

III. TACTICAL BEHAVIOR SPECIFICATION GRAMMAR

The Tactical Behavior Specification (TBS) language is defined to instruct a robot to perform tactical behavior including navigation, searching for an object or observation. The language is specifically focused on describing desired behavior using spatial relationships with objects in an environment. In this paper, we focus on the navigate action, where the main components of a command are a goal and a navigation mode. An object (or a symbol) referenced in a command can be associated with a spatial constraint relative to another object. For instance, in a command "Navigate covertly to a fire hydrant behind the building," a goal is to reach a fire hydrant, "behind the building" is a goal constraint, and "covertly" is the navigation mode. Often, the navigation mode also refers to an object. For instance, the navigation mode in "Navigate around the car to a fire hydrant behind the building" refers to an object named "car", which can also have its own spatial constraints that are independent from the constraints of the goal named "fire hydrant".

The full specification in the Backus-Naur Form (BNF) is included in Figure 2.

```
<tbs> ::= <trigger><action>[<direct-obj>][<mode>]
          <goal>[<goal-constraint>]
<trigger> ::= NONE
<action> ::= navigate | search | observe
<direct-obj> ::= <named-obj>
<named-obj> ::= "Robot" | "Building" | "Wall" |
                 "Door" | "Grass" | "Asphalt" |
                 "Concrete" | "Person" |
                 "TrafficBarrel" | "Car" |
                 "GasPump" | "FireHydrant"
<mode> ::= <simple-mode> { <path-constraint> }
<simple-mode> ::= "quickly" | "covertly"
<path-constraint> ::= <constraint-list>
<goal-constraint> ::= <constraint-list>
<constraint-list> ::= <constraint> | <constraint>
                      { <operator> <constraint> }
<constraint>::=[not]<spatial-relation><landmark>
                | [not] "(" <constraint-list> ")"
<spatial-relation> ::= left | right | behind |
                      front | around | near | away
<landmark> ::= <named-object>
<operator> ::= and | or
<goal> ::= { to | <spatial-relation> }<named-obj>
```

Fig. 2: Tactical Behavior Specification (TBS) language in BNF. In this work, we consider only navigation actions.

IV. Navigation Mode Grounding

Once the landmark objects have been grounded and the metric position of the goal has been determined, the robot must plan a path from its current position to the given goal location that obeys the path constraints imposed in the command. Path constraints describe a navigation mode. For example, the user may specify that the robot should stay "left of the building", or navigate "covertly". We return to the object grounding question, "which building should the robot stay left of?", in Section V.

Path constraints are subjective and explicitly writing down a cost function that encapsulates them would be time consuming due to the many trade-offs inherently present in the planning problem. For example, the robot must trade off path length with distance from the building. A covert navigation behavior may look different to different people. We instead use Imitation Learning to learn *how* to navigate between a start and end position using examples of desired behavior.

We treat understanding spatial language as learning a mapping from terms (such as "left of", "around", or "covertly") to a cost function c which can be used to generate a matrix of costs known as a cost map. A planner can then optimize to produce the minimum cost path under this cost function.

Specifically, given a term σ (such as "left of") in the command specifying the navigation mode, the robot solves the planning problem of finding the minimum cost path ξ^* under cost function c_σ:

$$\xi^* = \operatorname*{argmin}_{\xi \in \Xi} c_\sigma(\xi) = \operatorname*{argmin}_{\xi \in \Xi} w_\sigma^T \phi(\xi) \qquad (1)$$

where the set of valid paths is Ξ, and we assume that the cost function c_σ takes the form of a linear sum of features ϕ under weights w_σ. The features describe the shape of the path, the geometry of the landmark, and the relationship between the

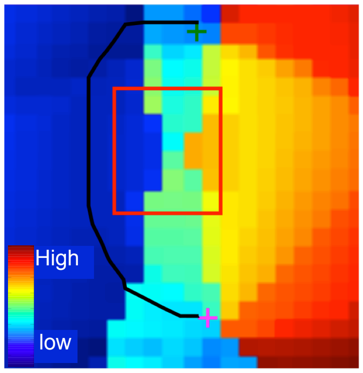
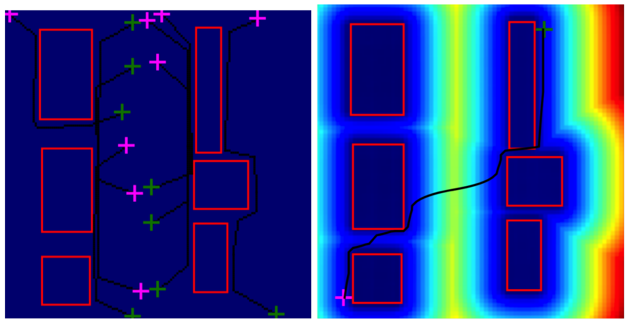


Fig. 3: Learned cost function for the navigation mode "left of". The landmark is the rectangle in the center (e.g., a building), and the path (optimized by minimizing the traversal costs) correctly stays on the left side of the landmark.



(a) Demonstrated paths. (b) Learned cost function and an optimal path generated accordingly.

Fig. 4: Demonstrated paths and resulting learned cost function (along with a validation example) for "covert" navigation. Paths are in black, the red rectangles are buildings. Each path starts with a pink cross and ends with a green one.

two [14]. We use imitation learning to learn the weights w_σ from a set of demonstrated paths $\{\hat{\xi}_i\}_1^N$.

To learn the weights w_σ, we minimize the difference between the cost of the expert's demonstrated path $\hat{\xi}$ and the minimum cost path under the current cost function:

$$\ell\left(w_\sigma, \hat{\xi}\right) = w_\sigma^T \phi(\hat{\xi}) - \min_{\xi \in \Xi} w_\sigma^T \phi(\xi) + \frac{\lambda}{2}\|w_\sigma\|^2 \qquad (2)$$

under our regularization parameter λ. The first term in Equation 2 is the cost of the demonstrated path under the current cost function, and the second term is the cost of the optimal path (again, under the current cost function). Note that we are omitting the loss-augmentation term for clarity. Ignoring regularization, we achieve zero loss when the cost function produces the expert's path. This loss is optimized using the sub-gradient technique [3], [21].

Figure 3 shows the learned cost function for the relation "left of", and Figure 4 shows the training examples and learned cost function for "covert" navigation, along with the minimum cost path for a given start and end. Note that we learned to avoid being in the center area between the rows of buildings, and the planner took a sharper path across.

V. Object Grounding with Spatial Constraints

A. Problem

The object grounding algorithm receives as inputs a text command, a set $\mathcal{O} = \{o_1, o_2 \ldots, o_n\}$ of perceived objects in the environment, and the position (x, y) of the robot at the time when the command was given. Each object o in set O is represented as a two-dimensional polygon, defined by the convex envelope of the object's points. Each object o is given a probability distribution P_o over labels $l \in \mathcal{L}$, obtained from the semantic perception module. For example, label l is "car" and $P_o(l)$ is the probability that object o is a car. A command contains one or more symbols from the label set $\mathcal{L}$. The symbols of particular interest for planning a path are the landmark-objects in goal and path constraints, denoted by ψ_g and ψ_p, respectively. For example, $\psi_g =$ "car" and $\psi_p =$ "building" in the command "Navigate near the building to the car that is behind the fire hydrant". The object grounding algorithm returns a joint probability distribution P on each pair of objects $(o_i, o_j) \in \mathcal{O} \times \mathcal{O}$. $P(o_i, o_j)$ is the probability that objects o_i and o_j are what the commander intended by symbols ψ_g and ψ_p, respectively. To compute $P(o_i, o_j)$, one needs to compute the probability of each object given all the symbols in the command, such as "behind" and "fire hydrant", in addition to symbols ψ_g and ψ_p. But only these two last symbols are used for planning a path. The other symbols only help grounding ψ_g and ψ_p. There are several ways for planning the robot's path based on the resulting distribution P. In this work, we pick the pair (o_i, o_j) that has the highest probability, and use it to generate a path according to the grounded navigation mode (Section IV). We show how P is computed in the rest of this section.

B. Model

Label prior distribution P_o is directly obtained from the object recognition method proposed in [1]. A uniform distribution on all labels in $\mathcal{L}$ is used for objects that do not belong to any known class. This happens when the robot encounters a new type of objects for the first time. Also, the symbols in a received command that are not in $\mathcal{L}$ are automatically added to $\mathcal{L}$. P_o is adjusted such that the probabilities of the new labels are nonzero. For instance, the user refers to a "fire hydrant" in our previous example. That means that one of the objects in the environment has to be grounded as a 'fire hydrant' even if this term (or label) was never used before.

We use a log-linear model to represent $P_{\mathcal{R},o_i}(o_j)$, the probability that object $o_j \in \mathcal{O}$ is the one that satisfies spatial relation $\mathcal{R}$ with object $o_i \in \mathcal{O}$,

$$P_{\mathcal{R},o_i}(o_j) = \frac{\exp\left(w_{\mathcal{R}}^T \phi(x, y, o_i, o_j)\right)}{\sum_{o_k \in \mathcal{O}} \exp\left(w_{\mathcal{R}}^T \phi(x, y, o_i, o_k)\right)}, \tag{3}$$

wherein $\phi(x, y, o_i, o_j)$ is a vector of spatial features of the objects o_i and o_j from the robot's perspective at current position (x, y), and $w_{\mathcal{R}}$ is a vector of weights specific to relation $\mathcal{R}$. We dropped the robot's coordinates (x, y) and objects set $\mathcal{O}$ from the notation $P_{\mathcal{R},o_i}$ because they are constant during the grounding process. The spatial features used here are the distance between center(o_i) and center(o_j), the centers of objects o_i and o_j, in addition to the sine and cosine of the angle between (x, y)-center(o_i) axis and center(o_i)-center(o_j) axis. These features are adequate for learning spatial relations between relatively small objects. For large objects, such as buildings, the spatial relations depend on the overall shape and orientation of the object. Therefore, we use Principal Component Analysis (PCA) to find the primary and secondary axis of o_i when o_i is most likely a building (according to the perception module), and replace the (x, y)-center(o_i) axis by the nearest axis to it among the primary and secondary axis. We also define the distance between a building o_i and the center of another object o_j as the smallest of the distances between o_j and each vertex of o_i. These geometric features were sufficient for learning weights $w_{\mathcal{R}}$ of all the spatial relations used in our experiments. The same general approach can be used for learning other relations by using additional features.

C. Inference

Algorithm 1 computes a joint distribution P on landmark-objects named as ψ_g and ψ_p in the goal and the path constraints of a TBS command (Figure 2). Each of the two objects can be subject to one or more spatial constraints, parsed as a binary tree and denoted by $\mathcal{T}_g$ and $\mathcal{T}_p$ respectively. We start by first computing a distribution on the objects in $\mathcal{O}$ for each label mentioned in the command. The object distribution, denoted by P_l for label l, is computed from the label distributions P_o (available from semantic perception) using Bayes' rule and a uniform prior. The next step consists in computing two distributions on goal and path landmarks, denoted as $P_{\mathcal{T}_g}$ and $P_{\mathcal{T}_p}$, from the spatial constraints in trees $\mathcal{T}_g$ and $\mathcal{T}_p$. The trees are traversed in a post-order depth-first search, which corresponds to reading the constraints in a reverse Polish notation. The logical operators ("and","or","not") are in the internal nodes of the tree, whereas the atomic spatial constraints ("behind building", "near car", etc.) are in the leaves. In the following, we show how object distributions $P_{\mathcal{T}_g}$ and $P_{\mathcal{T}_p}$ are recursively computed and updated (Algorithm 2). Since $P_{\mathcal{T}_g}$ and $P_{\mathcal{T}_p}$ are computed in the same way, we simply use P to denote both $P_{\mathcal{T}_g}$ and $P_{\mathcal{T}_p}$ in Algorithm 2.

At a leaf node with spatial relation $\mathcal{R}$ and symbol ψ used to indicate a reference object, P is given by

$$P(o_j) = \sum_{o_i \in \mathcal{O}} P_{\mathcal{R},o_i}(o_j) P_{\psi}(o_i). \tag{4}$$

The reference object symbol ψ is the name of the reference object in the spatial relation. For instance, the word "building" is the reference object symbol ψ in the relation "near the building". $P_{\mathcal{R},o_i}(o_j)$ is given by Equation 3. In Equation 4, every object $o_i \in \mathcal{O}$ is considered as a potential candidate for being the reference object intended by the user, with a prior probability $P_{\psi}(o_i)$ obtained from semantic perception.

At every internal node of the tree, two distributions P_{left} and P_{right} are calculated by recursively calling Algorithm 2.

The two distributions are combined according to the logical operator. We normalize the distributions by conditioning on the fact that one of the objects at least should satisfy all the constraints, using a uniform prior.

The last step of Algorithm 1 consists in combining $P_{\mathcal{T}_g}$ and $P_{\mathcal{T}_p}$, obtained from Algorithm 2, in a joint distribution P on pairs of goal and path (or navigation) landmark objects while taking into account the label priors given by the semantic perception. We also calculate the costs of paths that go to each candidate goal object using every potential navigation landmark. The path costs are computed as explained in Section IV. Pairs of goal and path landmarks that lead to costly paths are given a lower probability. The reweighting of P according to path costs is optional, we used it in the simulation experiments, but we removed it in the robotic experiments because of the real-time requirements.

D. Learning

Given a weight vector $w_{\mathcal{R}}$, probability $P_{\mathcal{R},o_i}(o_j)$ (Equation 3) indicates how likely a human user would choose o_j among all objects in a set $\mathcal{O}$ as the one that satisfies $\mathcal{R}(o_i,o_j)$. Because of perception uncertainties, estimating $P_{\mathcal{R},o_i}(o_j)$ for each object o_j is more important than simply finding the object that most satisfies relation $\mathcal{R}$ with o_j.

We used twenty examples for learning the spatial relations $\mathcal{R} \in$ {“left”, “right”, “front”, “behind”,“near”,“away”}. Each example i contains a set of objects in a simulated environment, a position (x_i, y_j) of the robot, a command with spatial constraints, in addition to the best answer o_i^* according to a human teacher. Weight vector $w_{\mathcal{R}}$ of each relation $\mathcal{R}$ is obtained by maximizing the log-likelihood of all the training examples using gradient descent, with the l_1 regularization for sparsifying the weights [22].

Algorithm 1 Object Grounding

```
Input: Binary trees $\mathcal{T}_g$ and $\mathcal{T}_p$ of goal and path BNF
       constraints in the command, set of objects $\mathcal{O}$,
       distribution $P_o$ on labels of each object, learned
       relation weights $\{w_{\mathcal{R}}\}$, robot position $(x, y)$
Output: Joint distribution $P$ on the landmark objects in
        the goal and the path constraints
foreach label $l$ occurring in the command do
    Calculate a posterior distribution $P_l$, defined as
    $P_l(o) = P(o|l)$, using Bayes Rule with $P_o$ and a
    uniform prior on objects $o \in \mathcal{O}$;
Calculate distribution $P_{\mathcal{T}_g}$ (resp. $P_{\mathcal{T}_p}$) on objects using
$\mathcal{T}_g$ (resp. $\mathcal{T}_p$), $\mathcal{O}$, $P_l$, $\{w_{\mathcal{R}}\}$, and $(x, y)$ in Algorithm 2;
foreach $(o_i, o_j) \in \mathcal{O} \times \mathcal{O}$ do
    Calculate $c_{i,j}$, the cost of the optimal path that
    goes from $(x, y)$ to goal object $o_i$, using $o_j$ as a
    navigation landmark;
foreach $(o_i, o_j) \in \mathcal{O} \times \mathcal{O}$ do
    $P(o_i, o_j) \propto (P_{\psi_g}(o_i) P_{\psi_p}(o_j) P_{\mathcal{T}_g}(o_i) P_{\mathcal{T}_p}(o_j)$
                 $\exp(-\alpha c_{i,j}))$
```

Algorithm 2 Recursive Grounding of Spatial Relations

```
Input: Binary tree $\mathcal{T}$ of BNF constraints describing
       spatial relations, set of objects $\mathcal{O}$, distribution
       $P_l$ on objects for each label, learned relation
       weights $\{w_{\mathcal{R}}\}$, robot position $(x, y)$
Output: Probability distribution $P$ on objects that sat-
        isfy constraints tree $\mathcal{T}$ with priors $P_l$
if $\mathcal{T}$ is a leaf node then
    foreach $o_i \in \mathcal{O}$ do
        Calculate $P(o_i)$ with Equation 4;
else
    Calculate $P_{\text{left}}$ (resp. $P_{\text{right}}$) recursively by calling
    Algorithm 2 with the left (resp. right) branch of $\mathcal{T}$;
    foreach $o \in \mathcal{O}$ do
        if $root\_node(\mathcal{T})$ = "and" then
            $P(o) \propto P_{\text{left}}(o) P_{\text{right}}(o)$;
        if $root\_node(\mathcal{T})$ = "or" then
            $P(o) \propto P_{\text{left}}(o) + P_{\text{right}}(o) - P_{\text{left}}(o) P_{\text{right}}(o)$;
if $not(\mathcal{T})$ then
    foreach $o \in \mathcal{O}$ do
        $P(o) \propto 1 - P(o)$;
```

VI. EXPERIMENTS

We performed experiments in simulated and real-world environments. We report here a summary of the results.

A. Simulation experiments

In the first simulation experiment, we created a simple world model containing 18 objects (Figure 5). The robot's environment is discretized as a 20×20 grid for path planning. Each object is given one label with a high probability (between 0.8 and 0.95), except one unknown object which is given a uniform distribution on the labels.

Table I shows all the TBS commands that were sent to the robot in this experiment, and the grounded goals. Detailed commands and answers are reported here because the interpretation of these results is subjective. However, we can verify that all these answers are valid. Note also how object 12 is correctly interpreted as the fire hydrant intended in the commands, although it is labeled as unknown with a uniform prior on the labels. Other objects, such as the traffic cone nearby, could well satisfy the spatial constraints in this case, but they have a low probability for label “fire hydrant”. Equation 4 shows how label priors were taken into account.

The second series of experiments is a study involving three uninformed human subjects. We created a world model with eleven objects: a building, two cars, six traffic cones and two unknown objects. We used five simple commands and five complex commands. Each command contains a navigation mode (“quickly” or “covertly”) with a spatial constraint of the path, in addition to a spatial constraint

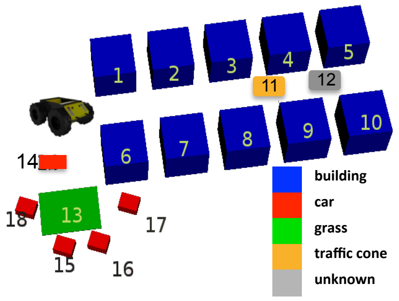


Fig. 5: A world model used with commands in Table I.

Command	Grounded Goal
“Navigate to Building right of Traffic Cone”	object 9
“Navigate to Building left of Traffic Cone”	object 4
“Navigate to Building right of Fire Hydrant”	object 10
“Navigate to Building left of Fire Hydrant”	object 5
“Navigate to Building near of Fire Hydrant”	object 5
“Navigate to Building away of Fire Hydrant”	object 6
“Navigate to Car behind of Grass”	object 15
“Navigate to Car front of Grass”	object 14
“Navigate to Car left of Grass”	object 17
“Navigate to Car right of Grass”	object 18
“Navigate to Car behind of Grass and right of Grass”	object 15
“Navigate to Car behind of Grass and left of Grass”	object 16
“Navigate to Car behind of Grass or left of Grass”	object 15
“Navigate to Car behind of Grass or left of Grass and near of Building”	object 17
“Navigate to Car not near of Building”	object 18

TABLE I: TBS examples and grounded goals in the world model of Figure 5, using Algorithm 1 with learned weights.

of the goal. Complex commands contain additional goal constraints. Participants were separately asked to point to the goal they would choose for executing each command. The best answer, chosen by a majority vote, is compared to the robot’s answer. Table II shows that the robot’s answer matches with the best answer in 80% of the commands. A robot’s answer is counted as valid if it matches the answer of at least one participant. All the grounded goals were valid in this study. We also report the consensus rate which is the percentage of commands where all the three participants agreed on one answer. The low rates of consensus clearly show the advantage of customized human-robot interfaces that can learn from users. For instance, one participant interpreted “front of a building” as the side where the cars were located. Similarly, we asked each participant to classify the robot’s path as conform to the navigation mode and constraints or as non-conform. The mode was classified as conform by the majority of the participants in only 60% of the commands. We noticed that the participants had all different definitions of what it means to navigate covertly.

B. Robot experiments

We performed extensive experiments using the robotic platform shown in Figure 6. The robot’s environment contained mainly buildings, cars, traffic cones, fire hydrants, and a gas pump. We evaluated the performance of the learned grounding model in five different scenes. Figure 7 shows one the scenes, as perceived by the robot. In each scene, we

	Simple Commands	Complex Commands
Best goal	80%	80%
Valid goal	100%	100%
Consensus	40%	20%
Best navigation mode	60%	60%
Valid navigation mode	100%	100%
Consensus	60%	60%

TABLE II: Comparing the learned grounding model to human subjects. Notice the low consensus among the subjects on the best answers, which are chosen by a vote of majority.

used five simple commands and five complex ones. The total number of test scenarios is then 50. In each test scenario, we select a goal and a navigation mode, send a command to the robot, and rate the planned path as a success if it matches the selected goal and mode, and as a failure otherwise. Table III shows the results of these experiments. Overall, we notice that complex commands help finding the right goals because they are less ambiguous than simple commands.

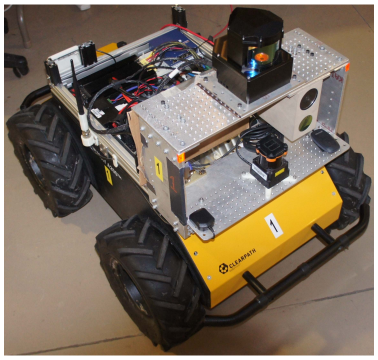


Fig. 6: The robotic platform used in our experiments: a Clearpath^{TM}Husky robot equipped with the General Dynamics XR 3D LADAR sensor and Adonis camera.

VII. CONCLUSION

Communicating with robots in natural language is a highly challenging problem. To correctly understand the different commands given to them, robots need to reason about their environments like humans. Spatial navigation and relations are one type of subjective linguistic concepts that robots can learn from human users. Our approach to solving this problem uses inverse optimal control for learning navigation modes, and a Bayesian model for trading off perception uncertainties with spatial constraints. Empirical evaluations show that the human-robot interface built using the proposed approach is an efficient tool for commanding mobile robots. In a future work, we plan to train the robot to ground other spatial relations and actions such as “search” and “observe”. This work can also be improved by considering more complex contextual features. For instance, the frontal façade of a building can be detected from the objects surrounding it.

(a) Simple Commands

	Correct Goals	Correct Navigation Modes
Scene 1	5/5	5/5
Scene 2	4/5	4/4
Scene 3	4/5	4/4
Scene 4	3/5	3/3
Scene 5	5/5	5/5
Average	**84±17 %**	**100±0 %**

(b) Complex Commands

	Correct Goals	Correct Navigation Modes
Scene 1	4/5	4/4
Scene 2	5/5	5/5
Scene 3	4/5	3/4
Scene 4	4/5	3/4
Scene 5	5/5	5/5
Average	**88±11 %**	**90±14 %**

TABLE III: Results of experiments using the robot.

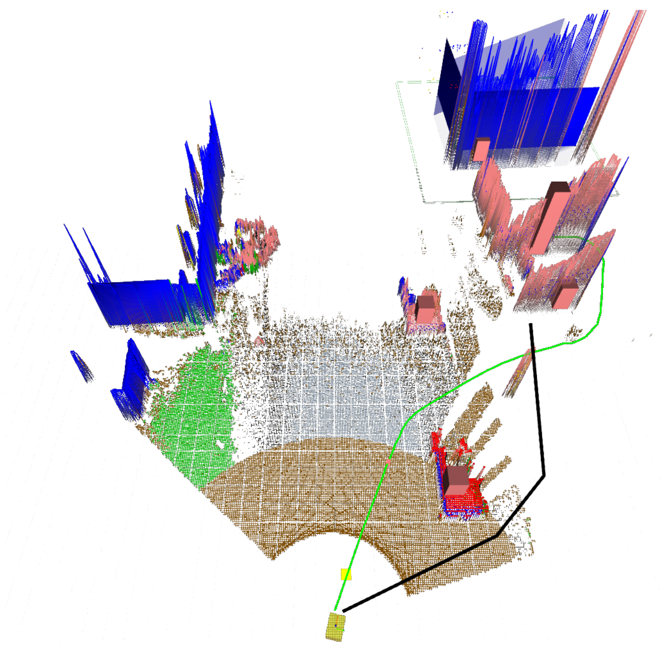


Fig. 7: Example of the experiments with the robot. Blue objects are classified as buildings and pink objects are classified as cars. Using the approach described in this paper, the green path is planned for the command "Navigate to car in front of a building and **behind** a car", while the black path is planned for the command "Navigate to car in front of a building and in **front** of a car".

ACKNOWLEDGMENT

This work was conducted through collaborative participation in the Robotics Consortium sponsored by the U.S Army Research Laboratory under the Collaborative Technology Alliance Program, Agreement W911NF-10-2-0016.

REFERENCES

[1] D. Munoz, *Inference Machines: Parsing Scenes via Iterated Predictions*. PhD thesis, The Robotics Institute, Carnegie Mellon University, June 2013.

[2] D. Munoz, J. A. Bagnell, and M. Hebert, "Stacked Hierarchical Labeling," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2010.

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

[4] A. Stentz, "Optimal and efficient path planning for partially-known environments," in *Proceedings of IEEE International Conference on Robotics and Automation (ICRA)*, pp. 3310–3317, 1994.

[5] D. Ferguson and A. Stentz, "Field D*: an interpolation-based path planner and replanner," in *Proceedings of the International Symposium on Robotics Research (ISRR)*, October 2005.

[6] J. P. Gonzalez, B. Nagy, and A. Stentz, "The geometric path planner for navigating unmanned vehicles in dynamic environments," in *Proceedings of the 1st Joint Emergency Preparedness and Response and Robotic and Remote Systems*, 2006.

[7] A. Stentz and B. Naggy, *PMAP User's Guide*. National Robotics Engineering Center, Carnegie Mellon University, 1.0 ed., Mar. 2007.

[8] S. Harnad, "The symbol grounding problem," *Physica D*, vol. 42, pp. 335–346, 1990.

[9] 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.

[10] 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.

[11] H. Zender, G.-J. M. Kruijff, and I. Kruijff-Korbayová, "Situated resolution and generation of spatial referring expressions for robotic assistants," in *Proceedings of the Twenty-First International Joint Conference on Artificial Intelligence (IJCAI)*, pp. 1604–1609, 2009.

[12] J. Dzifcak, M. Scheutz, C. Baral, and P. W. Schermerhorn, "What to do and how to do it: Translating natural language directives into temporal and dynamic logic representation for goal management and action execution," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pp. 4163–4168, 2009.

[13] D. Golland, P. Liang, and D. Klein, "A game-theoretic approach to generating spatial descriptions," in *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pp. 410–419, 2010.

[14] S. Tellex, T. Kollar, S. Dickerson, M. R. Walter, A. G. Banerjee, S. J. Teller, and N. Roy, "Understanding natural language commands for robotic navigation and mobile manipulation.," in *Proceedings of the 25th AAAI Conference on Artificial Intelligence*, 2011.

[15] T. Kollar, S. Tellex, D. Roy, and N. Roy, "Toward understanding natural language directions," in *Proceedings of the 5th ACM/IEEE International Conference on Human-robot Interaction (HRI)*, pp. 259–266, 2010.

[16] S. Tellex, P. Thaker, R. Deits, T. Kollar, and N. Roy, "Toward information theoretic human-robot dialog.," in *Robotics: Science and Systems IIX*, 2012.

[17] M. R. Walter, S. Hemachandra, B. Homberg, S. Tellex, and S. J. Teller, "Learning semantic maps from natural language descriptions," in *Robotics: Science and Systems IX*, 2013.

[18] C. Matuszek, N. Fitzgerald, L. Zettlemoyer, L. Bo, and D. Fox, "A joint model of language and perception for grounded attribute learning," in *Proceedings of the 29th International Conference on Machine Learning (ICML)*, pp. 1671–1678, 2012.

[19] S. Guadarrama, L. Riano, D. Golland, D. Gouhring, Y. Jia, D. Klein, P. Abbeel, and T. Darrell, "Grounding spatial relations for human-robot interaction," in *Proceedings of the 26th IEEE International Conference on Intelligent Robots and Systems (IROS)*, pp. 1640–1647, 2013.

[20] J. Oh, A. Suppe, F. Duvallet, A. Boularias, J. Vinokurov, L. Navarro-Serment, O. Romero, R. Dean, C. Lebiere, M. Hebert, and A. Stentz, "Toward Mobile Robots Reasoning Like Humans," in *Proceedings of the 29th AAAI Conference on Artificial Intelligence*, 2015.

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

[22] C. M. Bishop, *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Secaucus, NJ, USA: Springer-Verlag New York, Inc., 2006.