Learning Qualitative Spatial Relations for Robotic Navigation

Abdeslam Boularias
Department of Computer Science
Rutgers University

Felix Duvallet
École Polytechnique
Fédérale de Lausanne (EPFL)

Jean Oh and Anthony Stentz*
Robotics Institute
Carnegie Mellon University

Abstract
We consider the problem of robots following natural language commands through previously unknown outdoor environments. A robot receives commands in natural language, such as “Navigate around the building to the car left of the fire hydrant and near the tree”. The 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, i.e. mapping each noun in the command into a physical object in the environment. The robot needs also to ground a specified navigation mode, such as “navigate quickly” and “navigate covertly”, as a cost map. In this work, we show how to ground nouns and navigation modes by learning from examples provided by humans.

1 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”).

The robot must first recognize the objects in the environment and to label them. We use the semantic perception method proposed in [Munoz, 2013] which has been proven effective in outdoor environments [Munoz et al., 2010]. The semantic perception module receives scene images from a 2D camera and classifies each object into categories with different confidence values.

Grounding is performed by combining the labels obtained from semantic perception with the spatial relations obtained from parsing the command. There are three types 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.

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) [Ratliff et al., 2006]. 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* [Stentz, 1994; Ferguson and Stentz, 2005; Gonzalez et al., 2006; Stentz and Naggy, 2007]. The plan of the grounding result with the highest probability is executed by the robot. For more technical details on this work, we refer the reader to the longer version of this paper [Boularias et al., 2015] and the intelligence architecture paper [Oh et al., 2015] that describes how symbol grounding is used in semantic navigation.

2 Related Work

The challenge of building human-robot interfaces using natural language generated a large body of work [Harnad, 1990; MacMahon et al., 2006; Matuszek et al., 2012b; Zender et al., 2009; Dzifčak et al., 2009; Golland et al., 2010; Tellex et al., 2011; Kollar et al., 2010; Tellex et al., 2012; Walter et al., 2013; Matuszek et al., 2012a; Guadarrama et al., 2013]. 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 [Harnad, 1990] as the problem of mapping words (symbols) into manifestations in the physical world. The Generalized Grounding Graphs (G³) [Tellex et al., 2011] is a generic framework that casts symbol grounding as a learning and inference problem in a Conditional Random Field. The same type of spatial relation clauses presented in [Kollar et al., 2010] and used in [Tellex et al., 2011] are used in the current work. The navigation system described in [Walter et al., 2013], also based on G³, incorporates odometry and path constraints in grounding, which is conceptually comparable to our use of perception confidence and path costs in grounding. Guadarrama et al. [Guadarrama et al., 2013] 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.

3 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”.

4 Navigation Mode Grounding

Once the landmarks have been grounded and the 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 5.

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. 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 cσ:

\[ ξ^* = \arg\min_{ξ \in Ξ} c_σ(ξ) = \arg\min_{ξ \in Ξ} w^T_σ φ(ξ) \]

where the set of valid paths is Ξ, and we assume that the cost function cσ takes the form of a linear sum of features φ under weights wσ. The features describe the shape of the path, the geometry of the landmark, and the relationship between the two [Tellex et al., 2011]. We use imitation learning to learn the weights wσ from a set of demonstrated paths \( \{ξ_i\}_{i=1}^N \).

To learn the weights wσ, we minimize the difference between the cost of the expert’s demonstrated path ξ and the minimum cost path under the current cost function:

\[ ℓ(w_σ; ξ) = w^T_σ φ(ξ) - \min_{ξ' \in Ξ} w^T_σ φ(ξ') + \frac{λ}{2} \|w_σ\|^2 \]

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 [Ratliff et al., 2006].

Figure 2 shows the learned cost function for the relation “left of”, and Figure 3 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.
5 Object Grounding With Spatial Constraints

5.1 Problem

The object grounding algorithm receives as inputs a text command, a set \( O = \{o_1, o_2, \ldots, o_N\} \) 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 \(\psi_g\) and \(\psi_p\), respectively. For example, \(\psi_g = \text{“car”}\) and \(\psi_p = \text{“building”}\) 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 \((o_i, o_j) \in O \times O\). \(P(o_i, o_j)\) is the probability that objects \(o_i\) and \(o_j\) are the commander intended by symbols \(\psi_g\) and \(\psi_p\), 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 \(\psi_g\) and \(\psi_p\). But only these two last symbols are used for planning a path. The other symbols only help grounding \(\psi_g\) and \(\psi_p\). 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 4). We show how \(P\) is computed in the rest of this section.

5.2 Model

Label prior distribution \(P_o\) is directly obtained from the object recognition method proposed in [Munoz, 2013]. 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 \(P_{R,o_i}(o_j)\), the probability that object \(o_j \in O\) is the one that satisfies spatial relation \(R\) with object \(o_i \in O\):

\[
P_{R,o_i}(o_j) = \frac{\exp\left(w^T_R \phi(x, y, o_i, o_j)\right)}{\sum_{o_j \in O} \exp\left(w^T_R \phi(x, y, o_i, o_j)\right)},
\]

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 \(w_R\) is a vector of weights specific to relation \(R\). We dropped the robot’s coordinates \((x, y)\) and objects set \(O\) from the notation \(P_{R,o_i}\) 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 \(o_i\) and each vertex of \(o_j\). These geometric features were sufficient for learning weights \(w_R\) of all the spatial relations used in our experiments. The same general approach can be used for learning other relations by using additional features.

5.3 Inference

We show how to compute a joint distribution \(P\) on landmarks named as \(\psi_g\) and \(\psi_p\) in the goal and the path constraints of a command. Each of the two objects can be subject to one or more spatial constraints, parsed as a binary tree and denoted by \(T_g\) and \(T_p\) respectively. We start by first computing
a distribution on the objects in \( O \) 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 \( P_{T_l} \) and \( P_{T_o} \), from the spatial constraints in trees \( T_l \) and \( T_o \). 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.

5.4 Learning

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

We used twenty examples for learning the spatial relations \( R \in \{“left”, “right”, “front”, “behind”, “near”, “away”\} \). Each example \( i \) contains a set of objects in a simulated environment, a position \((x_i, y_i)\) of the robot, a command with spatial constraints, in addition to the best answer \( o_j^* \) according to a human teacher. Weight vector \( w_R \) of each relation \( R \) is obtained by maximizing the log-likelihood of all the training examples using gradient descent, with the \( l_1 \) regularization for sparsifying the weights [Bishop, 2006].

6 Experiments
6.1 Simulation experiments

These experiments are 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 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 1 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 (style) 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.

<table>
<thead>
<tr>
<th></th>
<th>Simple</th>
<th>Complex</th>
</tr>
</thead>
<tbody>
<tr>
<td>Best goal</td>
<td>80%</td>
<td>80%</td>
</tr>
<tr>
<td>Valid goal</td>
<td>100%</td>
<td>100%</td>
</tr>
<tr>
<td>Consensus</td>
<td>40%</td>
<td>20%</td>
</tr>
<tr>
<td>Best navigation mode</td>
<td>60%</td>
<td>60%</td>
</tr>
<tr>
<td>Valid navigation mode</td>
<td>100%</td>
<td>100%</td>
</tr>
<tr>
<td>Consensus</td>
<td>60%</td>
<td>60%</td>
</tr>
</tbody>
</table>

Table 1: Comparing the learned model to human subjects, using simple and complex commands. Navigation mode refers to style. Notice how low is the consensus among the subjects on the best answers, which are chosen by a majority vote.

6.2 Robot experiments

We performed extensive experiments using the robotic platform shown in Figure 1. 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. In each scene, we 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 (style), 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. Overall, we notice that complex commands help finding the right goals because they are less ambiguous than simple commands (88±11 vs. 84±17 success rate).

7 Conclusion

To become useful team-mates, robots will need to understand natural language commands given to them. This problem is highly challenging when the environment is unknown. 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 probabilistic 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.

Acknowledgment

This work was conducted in part through collaborative participation in the Robotics Consortium sponsored by the U.S Army Research Laboratory under the Collaborative Technology Alliance Program, Cooperative Agreement W911NF-10-2-0016, and in part by ONR under MURI grant “Reasoning in Reduced Information Spaces” (no. N00014-09-1-1052). The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the Army Research Laboratory of the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation herein.
References


