Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence

Toward Mobile Robots Reasoning Like Humans

Jean Oh, Arne Suppé, Felix Duvallet, Abdeslam Boularias, Luis Navarro-Serment, Martial Hebert, Anthony Stentz
Robotics Institute, Carnegie Mellon University, Pittsburgh, PA 15213

Jerry Vinokurov, Oscar Romero, Christian Lebiere
Department of Psychology, Carnegie Mellon University, Pittsburgh, PA 15213

Robert Dean
General Dynamics Robotic Systems, Westminster, MD 21157

Abstract

Robots are increasingly becoming key players in human-robot teams. To become effective teammates, robots must possess profound understanding of an environment, be able to reason about the desired commands and goals within a specific context, and be able to communicate with human teammates in a clear and natural way. To address these challenges, we have developed an intelligence architecture that combines cognitive components to carry out high-level cognitive tasks, semantic perception to label regions in the world, and a natural language component to reason about the command and its relationship to the objects in the world. This paper describes recent developments using this architecture on a fielded mobile robot platform operating in unknown urban environments. We report a summary of extensive outdoor experiments; the results suggest that a multidisciplinary approach to robotics has the potential to create competent human-robot teams.

1 Introduction

As robots become an integral part of human-robot teams, they will require new modalities for control, as well as sensing and reasoning capabilities that operate at the same level of their human counterparts. To become useful teammates, these robots will need to be able to understand natural language, recognize semantically meaningful objects around them, and perform high-level reasoning once given a task. This level of understanding would enable human teammates to easily cooperate with complex robots without requiring specialized interfaces, protocols, or training.

For example, in Figure 1, a human operator might command the robot to "navigate quickly to the back of the building that is behind the car." Completing this task successfully requires that the robot understand the components of the command, locate a car and a building and reason about their spatial relationships (without a map), hypothesize the complete shape of the building (which it may only see the front of) to identify an actual goal location that can be referred to as the back of the building, and finally plan a path around the building to that goal. In a field application, the robot must also deal with an incomplete model of the world and uncertainty due to perception and motion constraints.

While numerous previous studies have focused on some components of the overall system (*e.g.*, outdoor robot navigation (Bonin-Font, Ortiz, and Oliver 2008), cognitive capabilities for simulated environments (Anderson et al. 2004), scene classification and object detection (Lai et al. 2012), or language understanding for indoor robots (Tellex et al. 2011)), relatively little work exists to bring these capabilities together in one fielded system.

Existing cognitive approaches to robotics generally focus only on the cognitive side and do not necessarily address actual robotics challenges (Trafton et al. 2013), *e.g.*, the perception task is commonly simplified or is assumed to be complete and accurate. By contrast, our system acknowledges the current limitations of robotic perception; to cope with imperfect perception, we use a probabilistic world model that combines information from multiple sources including not only sensory perception but also general knowledge (learned offline) and linguistic clues.

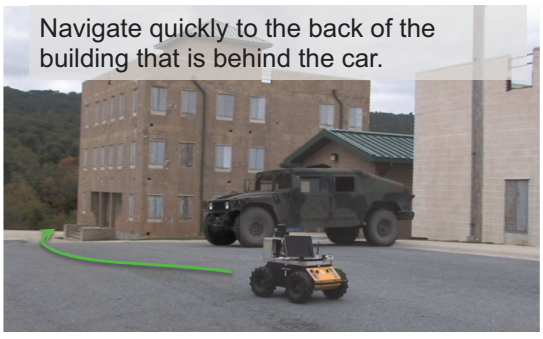


Fig. 1: The Husky platform executing symbolic navigation.

We have integrated multidisciplinary components into a combined intelligence architecture that enables a robot to

perform high-level cognitive tasks specified in natural languages in an unknown environment, leveraging recent developments from the fields of cognitive architectures, semantic perception, and natural language understanding. In contrast to black box approaches, our approach uses assumptive reasoning where the robot's rationale behind its decision making can be explained intuitively to human teammates by the use of predicted symbols and shapes with which people are familiar. This architecture is platform-independent and flexible to implement various tactical behaviors, *e.g.*, navigation, search, or manipulation. In this paper, we focus on outdoor navigation in urban environments, and report that our multidisciplinary approach enables the robot to carry out complex tasks in various real-life scenarios without the need of a map or priors of potential locations of objects and structures. We highlight unique opportunities and challenges engendered by integrating many different systems with complementary strengths.

The rest of this paper is organized as follows: We first introduce the high-level intelligence architecture in Section 2 and the language interface specifying tactical behaviors in Section 3, followed by an illustrative example in Section 4. Next, technical detail is described, focusing on perception (Section 5), prediction (Section 6), and language understanding (Section 7). Finally, we report the results in Section 8 and conclude the paper.

2 An Intelligence Architecture

We have devised a basic intelligence architecture that addresses the challenges discussed in Section 1 and have crafted a set of tasks and subtasks that add capabilities needed to achieve mission goals. The architecture is tightly coupled to the *world model* as shown in Figure 2. By world model, we mean a model for the world itself, the robot(s), and the interaction between the robot and world (Dean 2013).

The architecture consists of a hierarchy of tasks in three levels: *mission*, *attention* and *interaction*. Each task is a computational node that encapsulates a particular functionality. The node interleaves perception, planning, and execution monitoring to transform the world model from one belief state into another belief state. At each level, the world model stores all of the data for matching the task's pre- and post-conditions. The world model also stores resource models for the robot, the current task/subtask execution trace (for monitoring and inspection), and the history of this trace (for offline learning).

Mission Level: The tasks at the mission level implement specific doctrines. The tasks determine how to sequence atomic actions at the next level down (*e.g.*, navigation, manipulation, and perception) in order to achieve the mission. Mission-level tasks mimic human functionality in compliance with doctrines. The basic actions are pre-determined, but the challenge is to figure out when and how to apply these doctrine templates to particular cases, or modify actions based on situational awareness and context. The world model includes state data that represents contexts and situations as well as tasks and subtasks that are planned and being executed.

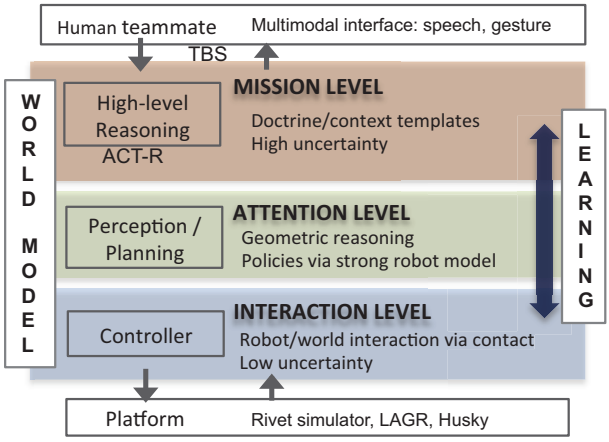


Fig. 2: An intelligence architecture for human-robot teams.

Attention Level: The mission-level tasks call attention-level tasks to navigate from place to place, grasp and manipulate objects, and perceive semantic objects and hazards. Rather than encoding doctrines, the reasoning at this level is primarily geometric. For example, a task reasons about how to move a manipulator to avoid obstructions and get into position to grasp an object. The world model includes grids of hazard data (*e.g.*, interpreted relative to a particular robot model); semantic objects such as doors or buildings; scrolling maps; and planned/executing tasks and their subgoals.

Interaction Level: The attention-level tasks call interaction-level tasks to affect motion and interact with the world via contact. Tasks at this level are essentially controllers. They typically cycle at 10 Hz or faster. The world model includes robot kinematics and dynamics and metric data, such as geometric shape, appearance, and material properties for objects in the world. The model includes robot/world interactions, such a forceful contact, tire-soil effects, etc. The uncertainty is typically confined to a small number of parameters such that standard parameter identification techniques can be used online (*e.g.*, solving for tire slip by comparing commanded trajectory to actual trajectory from an IMU).

In this paper, we specifically focus on the development of *tactical behaviors* that showcase how the task levels are interleaved inside the architecture. In general, a tactical behavior operates with partial information about the world. By leveraging both metric and semantic information, the intelligence architecture enables the robot to manage uncertainty in handling tactical behaviors in real-life environments.

3 Tactical Behavior Specification

This section describes the scope of tactical behaviors using a semi-structured language, known here as Tactical Behavior Specification (TBS)[1]. TBS grammar in Backus-Naur Form (BNF) is shown in Figure 3.

[1]The speech interface and the natural language unit that translates from free-form English to TBS is not covered in this paper.

```
<tbs> ::= <action><direct-obj>[<mode>][<action-constraints>]<goal>[<goal-constraints>]
<action> ::= navigate | search | observe | grasp
<direct-obj> ::= <named-obj>
<goal> ::= [ <relation> ] <landmark-object>
<goal-constraint> ::= <constraint-list>
<action-constraint> ::= <constraint-list>
<constraint-list> ::= <constraint-term> | <constraint-term> { <operator> <constraint-term> }
<constraint-term> ::= [not] <relation> <named-object> [<constraint-list>]
<mode> ::= "quickly" | "covertly" | "safely"
<relation> ::= "to" | "left" | "right" | "behind" | "front" | "around" | "near" | "away"
<landmark-object> ::= <named-object>
<operator> ::= and | or
<named-obj> ::= "Robot" | "Building" | "Wall" | "Door" | "Grass" | "Asphalt" | "Concrete" |
                "Person" | "TrafficBarrel" | "Car" | "GasPump" | "FireHydrant"
```

Fig. 3: Tactical Behavior Specification (TBS) language in BNF.

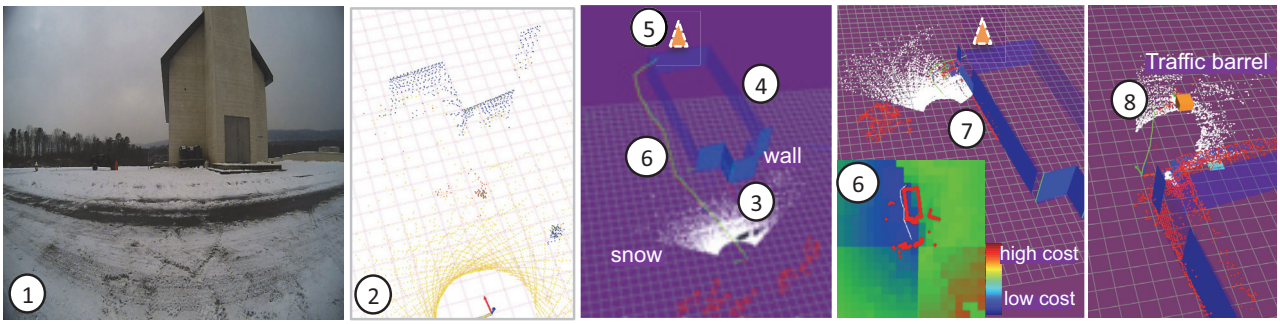


Fig. 4: Navigate left of the building to a traffic barrel that is behind the building. This paper is best viewed in color.

Example 1 (An annotated TBS command)

Navigate	covertly	left of the building
<action>	<mode>	<action-constraint>

to a traffic barrel	behind the building.
<goal>	<goal-constraint>

Example 1 shows an annotated example of a TBS command for the navigate tactical behavior; we use this example throughout the paper.

The TBS language supports a rich set of constraints that can leverage spatial relationships among objects in an environment. Whereas `goal-constraints` are used to specify a goal using its relative location with respect to landmarks, `mode` and `action-constraints` are used to specify how the robot should approach a goal, *e.g.*, keep to the left (as opposed to the right) of the building when navigating to the back of the building. In particular, a `mode` can impose subtle behavioral preferences, such as "covertly," that are generally difficult for humans to elicit in a clear formula. A spatial constraint is defined in terms of a spatial relation and a landmark object used as a reference. In the action-constraint in Example 1, for instance, "left of" and "the building" are referred to as `relation` and `landmark-object`, respectively. The symbols and their relationships referenced in the TBS are translated into actual objects and costmaps in the robot's world model via the language grounding process described in Section 7.

Using `and` and `or` operators, both conjunctive and disjunctive constraints can be expressed, *e.g.*, "left of the car and near the building."

The types of actions supported include "navigate," "search" and "observe." In this paper, we specifically focus on the navigate tactical behavior.

4 Example: Navigate Tactical Behavior

Figure 4 illustrates an end-to-end process for executing the TBS command in Example 1 and how various components interact within the architecture. Steps ① and ② show the robot's camera view and the LADAR scan inputs when the command is received, respectively. Step ③ shows that the front walls of a building are detected and new objects of type `wall` are created in the world model. This change in the world model triggers a context reasoner in the mission level, which predicts a composite object of type `building` from the set of detected walls as shown in ④. Example 1 includes two landmark-objects, `building` and `traffic barrel`, but the robot's current world model contains only a set of walls and a building. This inconsistency causes low grounding confidence, which, in turn, enables geometric spatial reasoning in the attention level, *i.e.*, based on the context in the command, a traffic barrel *must be* behind the building; an object is thus hypothesized behind the building as shown in ⑤. Now, the world model includes a building

and a traffic barrel, both predicted. After symbol grounding is done with sufficiently high confidence, the robot computes a navigation costmap that best satisfies the action constraint to stay to the "left of the building," and plans a path accordingly in step ⑥. Step ⑦ shows the predicted building being confirmed by the sensed data. Hitherto, the robot's goal is based on a hypothesized object behind the building. When the robot is near the rear corner of the building shown in ⑧, it detects a real traffic barrel via sensors and re-plans to complete the command.

5 Semantic Perception

Semantic objects are constructed incrementally using data from two types of sensors: a 2D camera and a 3D scanning LADAR sensor. The camera is used for scene classification and identification of the named objects, with the output of the classifier mapped to the LADAR point cloud. The labels include: wall, grass, asphalt, concrete, traffic barrel, car[2], gas pump, and fire hydrant.

In the next subsections, we describe the 2D camera-based semantic classification algorithm, followed by the 3D object classification.

5.1 2D Scene Classification

As opposed to employing a specific object detector for each type of object, we currently use a high-performance camera-based scene classifier (Munoz 2013) which has been demonstrated to perform well on outdoor scenes (Lennon et al. 2013). In addition to identifying large landmarks such as building façades, trees, pavement, and vehicles, we have also used it to identify regions of the image where traffic barrels, gas pumps and other smaller objects are present as shown in Figure 5.

This scene classifier labels superpixels using SIFT (Lowe 2004), LBP (Ojala, Pietikainen, and Maenpaa 2002), and texton features (Shotton et al. 2009) in a coarse-to-fine segmentation hierarchy, with segmentation of the smallest regions based on (Felzenszwalb and Huttenlocher 2004), and with larger regions constructed by combining neighboring regions with similar pixel statistics. A decision-forest classifier generates a predicted distribution of labels for each region in a coarse scale that is then passed to classifiers in the next level, which further refine the result using information at a more local scale. Ultimately, we use the most probable label for each of the finest superpixel regions to label the image. This strategy is effective for scenes, but dedicated object detectors as in (Zhu et al. 2014) can perform better for specific objects. We intend to supplement our system with these detectors especially for smaller objects in the future.

The scene classifier labels pixels, not the discrete objects that the intelligence architecture requires. For this reason, we use the 3D LADAR data to separate labeled pixels into discrete objects with coordinates in the world, relative to the robot, as well as to filter mislabeled border pixels and validate results.

[2]The label "car" is a generic term for vehicles. The type of vehicles used in the experiments are small cars, trucks, and HMMWVs.

Fig. 5: Semantic labeling in the 2D images.

5.2 3D Object Classification

For general object detection, we currently do not use predefined object models specifying geometric shapes. An exception to this are buildings that, due to their large size, can only be detected incrementally by combining multiple frames of data. We use RANSAC (Fischler and Bolles 1981) to look for planar surfaces from the input point cloud to detect a wall; neighboring walls can be merged into one or connected through corners to constitute a rudimentary structure for a building that can be used in cognitive reasoning at the mission level.

Fig. 6: The 3D classification: Both Object 1 (traffic barrel) and Object 2 (fire hydrant) are classified correctly with high confidence despite the fact that nearly a half of data points are mislabeled.

Semantic objects of other general types are constructed as follows. Given a set of semantically labeled points, we cluster points only according to the Euclidean distance between points. Note that the labels of input points are not used as a criterion for clustering, *i.e.*, a cluster is initially given a uniform prior over the set of class labels. This is to reduce potential false-positive detection due to misclassified border pixels. This step can be efficiently done by exploiting the octree data structure as in Point Cloud Library (PCL) (Rusu 2009).

Each resulting cluster is composed of a mixture of semantic labels. Using the labels of the constituent points, we update the cluster's posterior using the naïve Bayes approach. Let C denote a set of classes, and $L = \{l_1, ..., l_n\}, l_i \in C$, a set of labels from n points in a new cluster. For each class c in C, the posterior probability of the cluster belonging to class c given set L from a cluster can be written as:

$$p(c|l_1, ..., l_n) = \frac{1}{z} p(c) \prod_{i=1}^{n} p(l_i|c)$$

where z is a normalization factor. Here, the probability $p(l|c)$ of observing label l given class c represents how often the 2D classifier mistakes an object of class label c for

class label l; this probability distribution is experimentally obtained from the classifier's confusion matrix.

For example, as illustrated in Figure 6, traffic barrels frequently include some car pixels whereas cars rarely have traffic barrel pixels. In this case, if a new cluster contains both traffic barrel and car pixels evenly, the traffic barrel label will result in a higher probability for the cluster.

6 Prediction

If an object is too far from the robot or if the object is occluded by another object, then the robot may not be able to detect it. Due to various sensor range limitation, robots–unless operating in a pre-explored environment–generally do not have sufficient knowledge about the objects referred to in a TBS command. In the following subsections, we describe two different methods for hypothesizing objects in an environment.

6.1 Prediction Using Declarative Memory

Adaptive Control of Thought - Rational (ACT-R) is a cognitive architecture created to model human performance on psychological tasks (Anderson et al. 2004). ACT-R is mainly used in the mission level where it manages a high-level plan, monitors action status, and resolves conflicts or ambiguity. Additionally, ACT-R provides intelligence to hypothesize unseen parts of an environment. Navigation within a complex, dynamic environment requires an autonomous agent to possess a sense of context and an ability to project its "thought" processes into the future. To enable such capability for our robotic system, we constructed a cognitive model in the ACT-R framework that attempts to predict what building the robot is looking at by matching it to patterns in its declarative memory.

Encoded within ACT-R's declarative memory are geometrical patterns that represent buildings. These patterns may be learned explicitly or implicitly from experience, but in our particular case they were compiled by the experimenters. The ACT-R model begins by fetching clusters of walls from the world model; a cluster is simply any collection of walls identified by the perceptual front-end as belonging to a single physical structure. For example, two walls properly identified as meeting at a corner would both be members of the same cluster. Once all the clusters have been retrieved, ACT-R chooses the "most informative" cluster on the basis of three factors: cluster size (i.e. number of walls), cluster connectivity (i.e. number of corners), and the distance of the cluster center of mass from the robot's location. This heuristic encodes the common-sense knowledge that more information is better, and that the object located closer to the robot is likely to be the one of most interest.

After selecting the most informative cluster, the model attempts to match this cluster to a geometrical pattern in declarative memory. Due to limited vertical angle of sight for the LADAR ($\pm 10^{\circ}$), unless approached from a distance, the robot does not have a good view of the entire height of a structure, which means that only the length information of a wall is informative. If the model retrieves a pattern, it then attempts to project that pattern back into the world model. This is done by registering the points from the cognitive model to the data points observed in the world via a variant of the iterative closest point (ICP) algorithm (Besl and McKay 1992).

Once all the alignments have been tried, we select the one that has the smallest mean squared distance from the data. That prediction is now taken to be our current prediction, and updates to it occur only when they improve on the error of the original prediction. As the perceptual sensors gather more information from the world, the quality of the prediction should improve, thereby driving the error to a global minimum.

6.2 Prediction Based on Linguistic Context

In this section, we describe an approach that exploits contextual clues from commands.

When a command is given, the symbol grounder (described in Section 7.2) attempts to resolve uncertainty by grounding object symbols referred to in the command with actual object instances in the world model. The confidence of the grounder is determined based on how well those selected object instances satisfy spatial constraints specified in the command. Therefore, a low confidence value implies that the robot's current world model is inconsistent with the command. Such inconsistency can be addressed by relaxing a command that may be overly constrained, or by modifying the world model–*e.g.*, by introducing new variables (or objects) that satisfy the constraints. Here, we focus on the latter. In Figure 7, for instance, the robot's grounding confidence may be low if no objects can be found behind the building in the world model. Context-based object prediction is triggered in such a case.

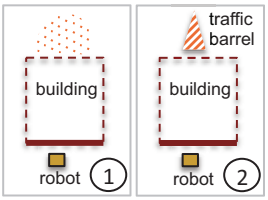


Fig. 7: Given a command "Navigate to a traffic barrel behind the building," a traffic barrel is hypothesized behind the building.

In the TBS, constraints are defined in the order of dependence, *e.g.*, a traffic barrel (that is behind a building (that is near a car (that is in front of the robot))). Constraints are propagated from the innermost variables–thus the most significant landmark objects–that are constrained only by their labels. In Figure 7, for instance, objects in the world model are first evaluated for their candidacies for "building". Next, variables that are referenced with respect to a building are evaluated, *i.e.*, for each object, the probability is computed for its being a traffic barrel and being located behind a building. During the evaluation process, if the set of objects satisfying the constraint is empty, then a new object is hypothesized such that the constraint is satisfied in that level.

In Figure 7, Step (1) shows that there are no objects that are behind the building in the robot's world model. The goal-constraint, *behind building*, is evaluated to generate a set of candidate locations to hypothesize a traffic barrel. In step

②, a goal object, *traffic barrel*, is hypothesized in the candidate location that best satisfies the goal-constraint. The algorithm for evaluating a spatial constraint cost is described in Section 7.2.

7 Language Grounding

The TBS decomposition presented in Section 3 enables us to leverage properties of spatial language, and enables to reason over the individual TBS components independently. We must understand *how* to travel such that our path obeys the linguistic command (spatial navigation), and *which* objects correspond to those specified by the user (symbol grounding). We address these challenges here, using models trained by imitation learning from expert demonstrations of correct behavior.

Other approaches to these types of issues include learning to ground objects and actions in the world from a natural language command (Tellex et al. 2011), hand-coded linguistic parsers (MacMahon, Stankiewicz, and Kuipers 2006) or ones learned from robot controller examples (Matuszek et al. 2012), and policy-based approaches for navigating in completely unknown indoor environments (Duvallet, Kollar, and Stentz 2013). Our focus is instead on understanding language while being robust to uncertainty due to changes in perceived and predicted objects (for example, misclassified or misplaced landmarks), and being efficient to allow for quick re-planning to utilize new information.

7.1 Language-Driven Spatial Navigation

We treat grounding spatial language as learning a mapping from terms (such as "left of", "around", or "covertly") to a cost function f which can be used to generate a matrix of costs known as a costmap. A planner can then optimize to produce the minimum cost path under the cost function f. Figure 8 shows two sample costmaps along with their minimum cost paths for two terms.

More specifically, given a spatial term in the TBS σ, the robot solves the planning problem of finding the minimum cost path ξ^* under the cost function f_σ:

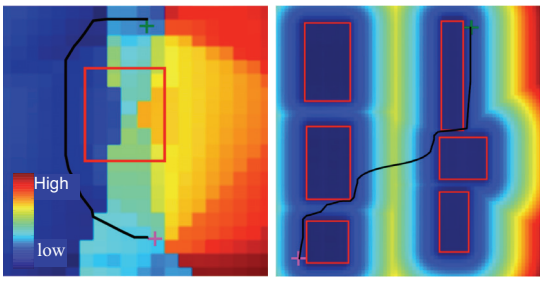
$$\xi^* = \underset{\xi \in \Xi}{\operatorname{argmin}} f_\sigma(\xi) = \underset{\xi \in \Xi}{\operatorname{argmin}} w_\sigma^T \phi(\xi) \qquad (1)$$

where the set of valid paths is Ξ, and we assume that the cost function f_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 two (Tellex et al. 2011). 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. This loss is optimized using maximum margin planning (Ratliff, Bagnell, and Zinkevich 2006; Ratliff, Silver, and Bagnell 2009).



Constraint "left of" Mode "covertly"

Fig. 8: Learned navigation cost functions and resulting paths (drawn in black, starting at the pink plus symbol), through several environments containing buildings (outlined in red).

7.2 Symbol Grounding Using Spatial Constraints

The symbol grounding module receives as inputs a TBS command, a set of objects $\mathcal{O} = \{o_1, o_2 \ldots, o_n\}$ in the world model, and the current position (x, y) of the robot at the time when the command is given. Each object o in set $\mathcal{O}$ is associated with probability distribution $p_o(l)$ over the class labels L, $l \in L$, given by the perception module (Section 5). A TBS command includes a set of symbols referred to as their labels. The symbols of particular interest in the path planning module are the landmark-objects in a goal and an action constraint, denoted by ψ_g and ψ_a, respectively. Let $\mathcal{L}_\psi$ denote the label of symbol ψ in a TBS. In Example 1, for instance, symbols ψ_g and ψ_a are labeled as "traffic barrel" and "building."

For each symbol in a TBS command, we associate an object in the world model that best matches what the user was referring to in a process known as *symbol grounding*. Therefore, the result of the symbol grounding is a joint probability distribution $P(\psi_g, \psi_a)$ on each pair of objects $(o_g, o_a) \in \mathcal{O} \times \mathcal{O}$.

Given symbol ψ mentioned in a TBS command, for each object o in $\mathcal{O}$, we find out probability $p(\psi = o|\mathcal{L}_\psi)$ that object o is what the commander intended by symbol ψ given its label. Here, probability $p_o(\mathcal{L}_\psi)$ of object o having the matching label with symbol ψ is used as a prior; this value is available from semantic perception.

If spatial constraints exist for symbol ψ–*e.g.*, traffic barrel ψ "behind the building"–then the next step consists of updating the probability distribution conditioned on the spatial constraints using Bayes' rule. Let r_γ and ψ_γ denote `relation-name` and a symbol representing the `landmark-object` of constraint γ. The posterior distribution given constraint γ is computed as:

$$p(\psi = o|\mathcal{L}_\psi, \gamma) \propto \sum_{o' \in \mathcal{O}} p_{o'}(\mathcal{L}_{\psi_\gamma}) \exp\left(w_{r_\gamma}^T \phi(x, y, o, o')\right),$$

where $\phi(x, y, o, o')$ is a vector of spatial features that are relative to the shape and positions of objects o and o' as well as the robot's current position (x, y); and $w_{r_\gamma}^T$, a vector of weights for relation r_γ that is learned by demonstration. The features considered here are the distance between the two objects, and features of the angle between the robot-o' axis

and the o-o' axis. The weights are obtained by maximizing the likelihood of training examples using gradient descent, with l_1 regularization for obtaining sparse weights (Bishop 2006).

Finally, we calculate the costs of the optimal paths that go from (x, y) to each candidate goal. The object distribution is re-weighted by penalizing the answers that lead to costly paths. The grounding module also returns a confidence value that indicates how well the answer satisfies the constraints.

8 Experimental Results

The complete end-to-end system has been integrated on Clearpath^{TM}Husky (Clearpath) equipped with the General Dynamics XR 3D LADAR sensor and Adonis camera (shown in Figure 1). The LADAR sensor is mounted $0.7\,\mathrm{m}$ above ground which creates approximately $4\,\mathrm{m}$ radius dead zone around the robot. A Hokuyo UTM-30LX scanning laser sensor is installed at $0.25\,\mathrm{m}$ for obstacle detection in the dead zone.

For path planning, we used PMAP with Field D* (Stentz 1994; Ferguson and Stentz 2005; Gonzalez, Nagy, and Stentz 2006). Field D* is a costmap-based algorithm that can efficiently compute an optimal path. The PMAP planner can combine multiple layers of costmaps, where each layer represents a different aspect of the map cells such as navigation difficulty, path preferences, or safety. The overall costs are computed using a weighted sum.

The examples and results reported in this paper are based on outdoor experiments conducted in a $1\,\mathrm{km}^2$ military training facility located in central Pennsylvania, US, that includes 12 buildings in a simulated town. The dates of the experiments spread between December 2013 to August 2014, covering varying conditions in terms of weather, sunlight, background, and terrain conditions. Table 3 lists the TBS commands, for clarity, in a less structured English format that can be fed to the system through a speech interface. Table 1 shows a summary of experimental results[3]

Table 1: Overall results.

Number of unique TBS commands	30
Total number of runs	46
Number of successful runs	35
Number of incomplete runs	11
Distance traveled per run (m)	21.0 ± 14.3
Number of runs traveled more than 30m	11

Here, an incomplete run is one where the robot started executing a valid initial plan but stopped prematurely due to various issues such as network communication failure and platform errors. In addition, actions were aborted by an operator when the robot's current plan appeared unsafe, *e.g.*, driving over challenging terrain.

Achieving high precision in object detection is crucial in carrying out tactical behaviors because a false positive detection can lead the robot to an arbitrary direction, often resulting in a failure. Detecting large-sized objects such as buildings is only limited by the sensor range ($\approx 50\,\mathrm{m}$), whereas it is more challenging to detect small-sized objects. In our experiments, the farthest building detected was $46\,\mathrm{m}$ away from the robot, whereas fire hydrants could be detected accurately only when they were within $10\,\mathrm{m}$ from the robot on average. The maximum range of $25\,\mathrm{m}$ for non-building type objects was empirically chosen to yield high precision, at the cost of a lower recall rate; that is, detections that are made outside this maximum range are disregarded to decrease the false positive error rate.

Table 2: Object composition in the world model.

Number of detections	8.3 ± 3.4
Number of predictions	3.2 ± 3.3

Since the robot is myopic, being able to hypothesize the unseen part of the world is an essential capability in symbolic navigation. As shown in Table 2, on average, 40% of the robot's world model was composed of predicted objects; more importantly, these predicted objects play key roles during the early stage of execution when the level of uncertainty is high. Predictions guide the robot to make plans with tentative goals; as the robot navigates, when a predicted object is within the robot's field of view, the prediction is either confirmed or invalidated based on sensed data. Subsequently, the robot continuously re-plans until a goal is reached.

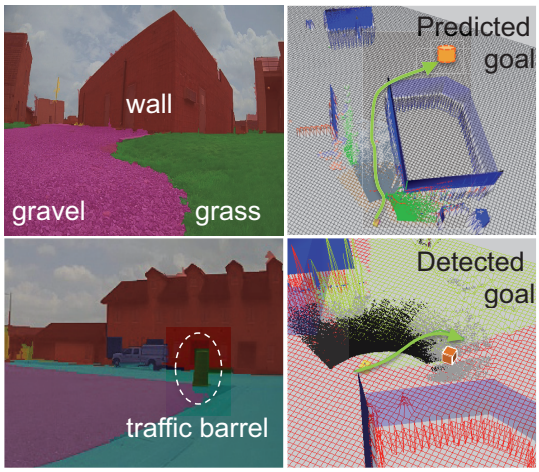


Fig. 9: Keep to the left of the building; navigate to a traffic barrel that is behind the building.

Figure 9 shows another run of the same TBS command used in Figure 4. Although it was for the same command, Figure 9 exhibits a more challenging case because there were several buildings of various shapes and sizes clustered closely. The first row shows that the robot's initial plan was created based on a hypothesized traffic barrel predicted behind the building that the robot was facing. The prediction was invalidated when it came into the robot's field of view. Here, if the real goal had not been detected, then another prediction could be made outside the robot's field of view. As shown in the second row, the robot had successfully detected

[3]Videos available at: http://www.nrec.ri.cmu.edu/projects/rcta/

Table 3: Sample TBS commands used in the experiments in free-form English.

Navigate covertly to the back of the building that is to the right of the car.
Navigate to a traffic barrel that is front/behind/left/right of the robot.
Navigate covertly/quickly to the left/right of the building/gas pump.
Navigate quickly to a car that is near the fire hydrant.
Navigate covertly/quickly to the building that is to the left/back of the car/fire hydrant.
Navigate quickly to the building that is near the traffic barrel.
Navigate covertly to the left of the building that is behind the fire hydrant and to the right of the fire hydrant.
Stay to the left/right of the building; navigate quickly to a traffic barrel that is to the left/right/back of the building.
Stay to the left of the car; navigate quickly to the building that is near the car.
Stay to the right of the car; navigate quickly to a traffic barrel that is behind the car.
Stay to the right of the car; navigate quickly to a traffic barrel that is to the left of the building.
Keep to the left of the gas pump; navigate quickly to a traffic barrel that is to the left of the gas pump.
Stay around the traffic barrel; navigate quickly to the building that is to the left of the traffic barrel.
Keep to the left/right of the gas pump; navigate quickly to the building that is to the back/left of the gas pump.
Keep to the right of the traffic barrel; navigate quickly to the right of the building that is near the traffic barrel.
Keep to the right of the fire hydrant; navigate covertly to the left of the building that is behind the fire hydrant.
Stay to the left of the building; navigate quickly to a fire hydrant that is to the left of the building
Stay to the left of the building; navigate quickly to the building that is near the car.
Stay to the right of the car; navigate covertly to the right of the car/building that is behind the car.

a real goal object and completed the action.

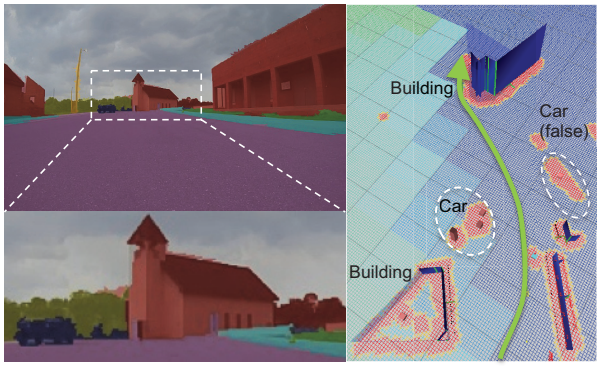


Fig. 10: Keep to the right of a car; navigate to a building that is behind the car. The map (right) displays $10\,\mathrm{m}$ grid; darker blue indicates a path preference of navigating the right side of the car.

Figure 10 illustrates one of the longest runs. Here, the robot's world model included a false detection of car (white circle on the right). Despite this error, the robot was able to ground the correct car as a landmark because the real car better satisfied the spatial constraint specified in the command–a building is behind a car–and the cost was lower.

In Figure 11, the robot needed to choose among three cars in front to ground an action-constraint landmark. Because navigating left of car1 or car2 was highly costly due to obstacles and a longer distance to travel to a traffic barrel, the robot selected car3 to be the landmark with high confidence.

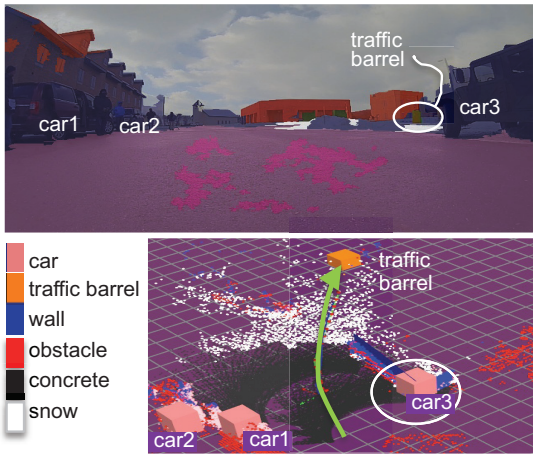


Fig. 11: Keep to the left of a car; navigate to a traffic barrel.

9 Conclusion

This paper describes an intelligence architecture for human-robot teams and details its application to a mobile robot system that goes beyond metric navigation to rich symbolic navigation. By leveraging high-level contextual reasoning capabilities and semantic understanding of metric information, the system enables mobile robots to navigate and perform nontrivial tasks in real-world environments. The system has been verified through extensive outdoor experiments; results suggest that our integrated approach to robotics has the potential to create competent human-robot teams.

Building on the results presented in this paper, we are working on more complex team tasks that require a mission-level planning capability to compose various actions in sequence or in parallel and a bidirectional communication capability through a multimodal interface.

Acknowledgments

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

Anderson, J. R.; Bothell, D.; Byrne, M. D.; Douglass, S. A.; Lebiere, C.; and Qin, Y. 2004. An integrated theory of the mind. *Psychological Review* 111:1036–1060.

Besl, P. J., and McKay, N. D. 1992. Method for registration of 3-d shapes. In *Robotics-DL tentative*, 586–606. International Society for Optics and Photonics.

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

Bonin-Font, F.; Ortiz, A.; and Oliver, G. 2008. Visual navigation for mobile robots: A survey. *Journal of intelligent and robotic systems* 53(3):263–296.

Clearpath. Clearpath Robotics™, Husky. http://www.clearpathrobotics.com.

Dean, R. 2013. Common world model for unmanned systems. In *SPIE*.

Duvallet, F.; Kollar, T.; and Stentz, A. 2013. Imitation learning for natural language direction following through unknown environments. In *Proc. International Conference on Robotics and Automation (ICRA)*.

Felzenszwalb, P. F., and Huttenlocher, D. P. 2004. Efficient graph-based image segmentation. *Int. J. Comput. Vision* 59(2):167–181.

Ferguson, D., and Stentz, A. 2005. Field D*: an interpolation-based path planner and replanner. In *Proc. the International Symposium on Robotics Research (ISRR)*.

Fischler, M. A., and Bolles, R. C. 1981. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM* 24(6):381–395.

Gonzalez, J. P.; Nagy, B.; and Stentz, A. 2006. The geometric path planner for navigating unmanned vehicles in dynamic environments. In *Proc. ANS 1st Joint Emergency Preparedness and Response and Robotic and Remote Systems*.

Lai, K.; Bo, L.; Ren, X.; and Fox, D. 2012. Detection-based object labeling in 3d scenes. In *Proc. International Conference on Robotics and Automation (ICRA)*, 1330–1337. IEEE.

Lennon, C.; Bodt, B.; Childers, M.; Camden, R.; Suppé, A.; Navarro-Serment, L.; and Florea, N. 2013. Performance evaluation of a semantic perception classifier. Technical Report ARL-TR-6653, United States Army Research Laboratory, Adelphi, Maryland.

Lowe, D. G. 2004. Distinctive image features from scale-invariant keypoints. *Int. J. Comput. Vision* 60(2):91–110.

MacMahon, M.; Stankiewicz, B.; and Kuipers, B. 2006. Walk the talk: Connecting language, knowledge, and action in route instructions. In *Proc. National Conference on Artificial Intelligence*.

Matuszek, C.; Herbst, E.; Zettlemoyer, L.; and Fox, D. 2012. Learning to parse natural language commands to a robot control system. In *International Symposium on Experimental Robotics (ISER)*.

Munoz, D. 2013. *Inference Machines: Parsing Scenes via Iterated Predictions*. Ph.D. Dissertation, The Robotics Institute, Carnegie Mellon University.

Ojala, T.; Pietikainen, M.; and Maenpaa, T. 2002. Multiresolution gray-scale and rotation invariant texture classification with local binary patterns. *Pattern Analysis and Machine Intelligence, IEEE Transactions on* 24(7):971–987.

Ratliff, N. D.; Bagnell, J. A.; and Zinkevich, M. A. 2006. Maximum margin planning. In *Proc. International Conference on Machine Learning (ICML)*.

Ratliff, N. D.; Silver, D.; and Bagnell, J. A. 2009. Learning to search: Functional gradient techniques for imitation learning. *Autonomous Robots*.

Rusu, R. B. 2009. *Semantic 3D Object Maps for Everyday Manipulation in Human Living Environments*. Ph.D. Dissertation, Computer Science department, Technische Universitaet Muenchen, Germany.

Shotton, J.; Winn, J.; Rother, C.; and Criminisi, A. 2009. Textonboost for image understanding: Multi-class object recognition and segmentation by jointly modeling texture, layout, and context. *Int. J. Comput. Vision* 81(1):2–23.

Stentz, A. 1994. Optimal and efficient path planning for partially-known environments. In *Proc. International Conference on Robotics and Automation (ICRA)*, 3310–3317.

Tellex, S.; Kollar, T.; Dickerson, S.; Walter, M. R.; Banerjee, A. G.; Teller, S. J.; and Roy, N. 2011. Understanding natural language commands for robotic navigation and mobile manipulation. In *Proc. AAAI Conference on Artificial Intelligence (AAAI)*.

Trafton, G.; Hiatt, L.; Harrison, A.; Tamborello, F.; Khemlani, S.; and Schultz, A. 2013. Act-r/e: An embodied cognitive architecture for human-robot interaction. *Journal of Human-Robot Interaction* 2(1):30–55.

Zhu, M.; Atanasov, N.; Pappas, G. J.; and Daniilidis, K. 2014. Active deformable part models inference. In *Proc. European Conference on Computer Vision (ECCV)*, 281–296. Springer.