UNIVERSITY OF LEEDS

This is a repository copy of *BWIBots: A platform for bridging the gap between AI and human–robot interaction research*.

White Rose Research Online URL for this paper:
http://eprints.whiterose.ac.uk/112584/

Version: Accepted Version

Article:
Khandelwal, P, Zhang, S, Sinapov, J et al. (10 more authors) (2017) BWIBots: A platform for bridging the gap between AI and human–robot interaction research. International Journal of Robotics Research, 36 (5-7). pp. 635-659. ISSN 0278-3649

https://doi.org/10.1177/0278364916688949

© 2017, The Author(s). This is an author produced version of a paper published in International Journal of Robotics Research. Uploaded in accordance with the publisher's self-archiving policy.

Reuse
Unless indicated otherwise, fulltext items are protected by copyright with all rights reserved. The copyright exception in section 29 of the Copyright, Designs and Patents Act 1988 allows the making of a single copy solely for the purpose of non-commercial research or private study within the limits of fair dealing. The publisher or other rights-holder may allow further reproduction and re-use of this version - refer to the White Rose Research Online record for this item. Where records identify the publisher as the copyright holder, users can verify any specific terms of use on the publisher's website.

Takedown
If you consider content in White Rose Research Online to be in breach of UK law, please notify us by emailing eprints@whiterose.ac.uk including the URL of the record and the reason for the withdrawal request.

White Rose university consortium
Universities of Leeds, Sheffield & York

eprints@whiterose.ac.uk
https://eprints.whiterose.ac.uk/

Journal Title
XX(X):1–26
©The Author(s) 2015
Reprints and permission:
sagepub.co.uk/journalsPermissions.nav
DOI: 10.1177/ToBeAssigned
www.sagepub.com/
SAGE

BWIBots: A platform for bridging the gap between AI and Human-Robot Interaction research

Piyush Khandelwal[1], Shiqi Zhang[1,2], Jivko Sinapov[1], Matteo Leonetti[1,3], Jesse Thomason[1], Fangkai Yang[4], Ilaria Gori[5], Maxwell Svetlik[1], Priyanka Khante[1], Vladimir Lifschitz[1], J.K. Aggarwal[5], Raymond Mooney[1] and Peter Stone[1]

Abstract

Recent progress in both Artificial Intelligence (AI) and Robotics have enabled the development of general purpose robot platforms that are capable of executing a wide variety of complex, temporally extended service tasks in open environments. This article introduces a novel, custom-designed multi-robot platform for research on AI, robotics, and especially Human-Robot Interaction (HRI) for service robots. Called BWIBots, the robots were designed as a part of the Building-Wide Intelligence (BWI) project at the University of Texas at Austin. The article begins with a description of, and justification for, the hardware and software design decisions underlying the BWIBots, with the aim of informing the design of such platforms in the future. It then proceeds to present an overview of various research contributions that have enabled the BWIBots to better (i) execute action sequences to complete user requests, (ii) efficiently ask questions to resolve user requests, (iii) understand human commands given in natural language, and (iv) understand human intention from afar. The article concludes with a look forward towards future research opportunities and applications enabled by the BWIBot platform.

1 Introduction

Research in Artificial Intelligence (AI) has long assumed that one day there would be general purpose robotic platforms that could execute symbolic actions, and especially long and complex sequences of such actions. However, until recently, most robots have been limited to performing small sets of actions in very limited configuration spaces for relatively short periods of time.

Recent progress in both the hardware robustness and software sophistication of mobile robots has finally enabled the integration of modern AI planning, reasoning, sensing, and acting all onboard physical robots that are capable of long-term autonomy in open, dynamic, and human-inhabited environments. On the other hand, this progress has exposed the integration challenges of combining low-level action with high-level planning, especially in the face of the inherent uncertainty that comes from Human-Robot Interaction (HRI). In this article, we demonstrate how an intelligent service robot, capable of high-level planning and reasoning, can be used for robust HRI.

The aim of this article is two-fold. First, we introduce a novel, custom-designed multi-robot platform for research on such integration of AI, robotics, and especially HRI on indoor service robots. Called BWIBots, the robots were designed as a part of the Building-Wide Intelligence (BWI) project at the University of Texas at Austin. The long-term goal of the BWI project is to deploy a pervasive autonomous system inside a building, with end effectors such as robots, to better serve both inhabitants and visitors.

Second, we illustrate the overall purpose of our robotic system, which is to enable novel research in the context of the human-interactive service robot domain. In particular, we briefly summarize five research contributions enabled by the BWIBots, that are geared towards improving the ability of indoor service robots to understand human intention during interaction, and execute actions as necessary to carry out human commands. The collective breadth of these loosely-related research projects illustrate the research

[1]Department of Computer Science, UT Austin, Austin, TX, USA
[2]Department of EECS, Cleveland State University, Cleveland, OH, USA
[3]School of Computing, University of Leeds, Leeds, UK
[4]Schlumberger Software Technology, Katy, TX, USA
[5]Electrical and Computer Engineering, UT Austin, Austin, TX, USA

Corresponding author:
Piyush Khandelwal, Department of Computer Science, University of Texas at Austin, 2317 Speedway, Stop D9500, Austin TX 78712, USA.
Email: piyushk@cs.utexas.edu

versatility of the platform, having enabled contributions to a variety of AI sub-areas beyond HRI, including AI planning, knowledge representation and reasoning, natural language processing, and machine learning.

Specifically, we cover the following contributions using the BWIBots in this article:

Planning using action language $\mathcal{BC}$: We describe how domain knowledge and planning descriptions for robots can be written using action language $\mathcal{BC}$, allowing robots to achieve complex goals using defeasible reasoning* and indirect/recursively defined fluents (Khandelwal et al. 2014).

Integrating probabilistic and symbolic reasoning: We describe how robots can incorporate probability distributions with symbolic reasoning to implement a spoken dialog system, allowing them to intelligently ask questions in order to quickly understand human instructions (Zhang and Stone 2015).

Understanding natural language requests: Since one of the most convenient means for humans to convey instructions is natural language, we describe how natural language requests can be understood by robots by grounding requests using a robot's existing domain knowledge, and how robots can incrementally learn larger vocabularies through conversation (Thomason et al. 2015).

Grounded multimodal language learning: We describe how a robot can learn to ground certain human instructions, such as "Bring me a full, red bottle.", in its perception and actions (Thomason et al. 2016).

Robot-centric human activity Recognition: We describe how a robot can categorize human activity using standard machine learning techniques, in order to better understand the behavior of humans in its vicinity (Gori et al. 2015).

The remainder of the article is organized as follows. In the next section, we discuss other indoor service robot systems that aim to solve similar problems as the BWIBots. In Sections 3 and 4, we present the hardware and software design decisions behind the BWIBots, along with their justifications relative to considered alternatives. A main aim of this component of the article is to share our development insights and experience with future developers of similar platforms for service robotics and HRI, and these two sections serve as the main novel contributions of this paper. In Sections 5–9, we summarize the five research contributions outlined above. The article then concludes with a look forward towards future research opportunities, especially in multi-robot coordination, that we expect will be enabled by the BWI platform.

2 Related Work

This section discusses other multi-robot systems that share some of the same research goals as the BWI project. Sections 5–9 independently cover work related to the research areas presented within those sections.

In recent years, multiple autonomous service robot systems have been developed that are designed to interact with humans and operate within human-inhabited environments. Mobile robot platforms range from service robots such as the Care-O-bot 3 (Reiser et al. 2009) and research robots such as the uBot-5 (Kuindersma et al. 2009) to personal robots such as the PR2 (Cousins 2010) and Herb 2.0 (Srinivasa et al. 2012). In this section, we discuss representative single-robot and multi-robot systems that are used for research similar to that presented in this paper.

The Collaborative Robot (CoBot) platform (Veloso et al. 2015) is a multi-robot system that exists symbiotically with humans. CoBots establish a symbiotic relationship with humans, as they fulfill human commands while requesting human help for achieving difficult tasks such as using an elevator (Rosenthal et al. 2010). This technique is also employed on the BWIBots. Furthermore, CoBots use mixed integer programming for scheduling tasks, and use a web-based interface to accept user requests (Coltin et al. 2011). In contrast, BWIBots are used to research the complimentary problem of robust planning, where it is necessary to select the best sequence of actions to complete a single user request efficiently.

The SPENCER project aims to enable a robot to treat humans in the environment as more than simple obstacles (*SPENCER Project* 2016). Specifically, this project focuses on allowing robots to perform socially-aware task, motion, and interaction planning, while interacting with groups of people. Research contributions are targeted at tracking multiple people as social groups (Luber and Arras 2013), and performing robust navigation in the midst of crowds (Vasquez et al. 2014). While some of the research performed using the BWIBots focuses on recognizing human activity in the robot's vicinity, research contributions described in this paper aim to improve direct interaction with a single human via natural language dialog systems.

The STRANDS project is concerned with allowing robots to gather knowledge about the environment over an extended period of time, as well as learn spatio-temporal dynamics in human-inhabited environments (*STRANDS* 2016). By learning the dynamics of obstacles such as humans and non-stationary furniture, the goal of the STRANDS project is to allow a robot to run

*Defeasible reasoning allows a planner to draw tentative conclusions which can be retracted based on further evidence.

(a) BWIBotV1 **(b)** BWIBotV2 **(c)** BWIBotV3

Figure 1. The evolution of the BWIBot platform. BWIBotV2 features a smaller profile and improved DC converters when compared to the BWIBotV1. BWIBotV3 makes further improvements by using the new RMP 110 base, onboard auxiliary battery, desktop computer and touchscreen, and the Velodyne VLP-16 for navigation.

autonomously for significantly long periods such as 120 days. Similar to the CoBots, research contributions within the STRANDS project have focused more on scheduling (Mudrova and Hawes 2015) than general purpose planning.

The RoboCup@Home competition (Wisspeintner et al. 2009) aims to enhance service robots by providing benchmark tests that evaluate a robot's ability to perform in realistic home environments. These benchmark tasks require manipulation, object recognition, and robust navigation among other features necessary for domestic service robots. The Kejia robot, winner of Robocup@Home in 2014 (Chen et al. 2014), has been used to identify what knowledge is necessary to completely ground human requests, and search for missing information using open knowledge, i.e. free-form knowledge available online (Chen, Xie, Ji and Sui 2012). While the RoboCup@Home competition is designed to test the versatility of service robots, and benchmarks test a breadth of capabilities, research contributions performed using the BWIBots are more focused and improve the state-of-the-art on somewhat more specialized, but deeper, problems than those typically defined by RoboCup@Home.

3 Hardware

In this section, we briefly describe the hardware design of the BWIBots. The design goals behind these robots include robust navigation inside a building, continuous operation for 4–6 hours, ease of interaction with humans, and a configurable array of sensors and actuators depending on the research application. The robots have continually evolved while following these design goals, based on research applications that have emerged since their inception (see Figure 1).

The main aim of this section is to share our development insights and experience with future developers of similar platforms for service robotics and human-robot interaction inside a building, especially for the purpose of academic research. It also serves as an introduction to the substrate platform that is used for research presented in the remainder of this article.

3.1 Mobile Base and Customized Chassis

The latest iteration of the BWIBot platform (BWIBotV3) is built on top of the differential drive Segway RMP 110 mobile base available from Stanley Innovation. Prior to the RMP 110, the RMP 50 was used to build the BWIBotV1 and BWIBotV2 versions[†]. The RMP platform was selected to construct the BWIBots because it balances cost with many different features such as maximum payload capacity (100lbs), size (radius = 30cm), and maximum speed (2m/s for the RMP 50, 5m/s for the RMP 110). Additionally, it provides sufficiently accurate odometry estimates for robust navigation. Compared to most other RMP platforms, the RMP 110 does not have an external user interface box and is extremely space efficient, allowing more space for the customized chassis, and also provides power for auxiliary devices, as explained in Section 3.2.

A customized chassis that holds the computer, sensors, and touchscreen is mounted on top of the RMP 110 mobile base. The chassis is constructed using aluminum (6061-T6 alloy) sheet metal and aluminum framing from 80/20 Inc[‡]. All sheet metal parts were designed using the open-source CAD software FreeCAD. Prior to fabrication, all parts were prototyped in acrylic using a Full Spectrum P-Series 20"x12" CO2 laser cutter[§], allowing design revision with a fast turnaround. The final parts were fabricated in aluminum using commercial waterjet cutting service BigBlueSaw.

The computer controlling the robot is not directly screwed into the chassis; rather it is mounted on a plate which is then latched to the chassis. This feature allows easy removal of the computer (and plate) for diagnosis, repair, and replacement. Additionally, the surface of the chassis above the computer and exposed electronics has been waterproofed using IP54 cable glands and washers, even though the entire chassis is not water-proof, providing some resistance against accidental spills on the robot.

[†]The RMP 50 is no longer available for sale.

[‡]80/20 framing has already been used on other research robots such as the Cobot (Veloso et al. 2015)

[§]Parts larger than 20"x12" were split to fit on the cutting bed, and then joined together using joining plates from 80/20 Inc.

Furthermore, the chassis on the BWIBotV2 and BWIBotV3 has been designed to fit within the smallest circumscribed circle possible given the size of the RMP50 and RMP110, respectively. Most navigation algorithms consider robots to be circular, and a small circular footprint simplifies navigation around obstacles. In BWIBotV1, the circumscribed radius induced by the chassis was larger than the one induced by the mobile base, but the navigation algorithm was provided with a smaller radius in order to navigate through narrow corridors and doors. Consequently, on rare occasions, the back of the BWIBotV1 would hit obstacles when turning in place.

3.2 Auxiliary Power and Power Distribution

The RMP 110, used to construct the BWIBotV3, contains two 384Wh Lithium Iron Phosphate (LiFePo) batteries. One is used for peripherals such as the computer and various sensors, and the other for driving the mobile base. In contrast, in previous versions of the BWIBot, the RMP 50 did not provide a power source for peripherals. A single 12V 1280WH LiFePo battery was used on those platforms to power both the drive system and peripherals. Batteries with a LiFePo chemistry have been used as they are extremely safe, and have a longer lifespan than other chemistries when repeatedly deep-discharged.

The RMP 110 provides a regulated 12V 150W power source using the auxiliary battery, which is sufficient to power all peripherals. On the RMP 50, the same regulated power source has been constructed using a Vicor DC-DC converter with the LiFePo battery as the source. Since some peripherals require an input voltage of 5V or 19V at low currents, the 12V source is re-regulated using 5V 45W and 19V 35W DC-DC converters from Pololu Robotics. These additional DC-DC converters, along with Anderson Powerpole and Molex power connectors, are soldered on a power distribution PCB designed using the open-source software Fritzing, and manufactured using the PCB fabrication service OSH Park.

3.3 Computation and Interface

The BWIBotV3 contains a desktop computer powered by an Intel i7-4790T/i7-6700T processor, placed in HD-Plex H1.S fanless case, with 6 Gigabit Ethernet Network Interfaces, along with 4 USB3 and 2 USB2 interfaces. A 20” touchscreen is mounted at a human-operable height to serve as the primary user interface with the robot. Earlier versions of the BWIBot contained a laptop powered by an Intel i7-3612QM processor mounted at a human-operable height, serving both computational and user interface requirements on the robot. This laptop contained 1 Gigabit Ethernet and 3 USB3 connectors, which was insufficient for the number of peripherals on the robot, and required the placement of an additional USB Hub and Gigabit Ethernet Switch on the robot.

3.4 Perception

Perception is used for both navigation (robot localization and obstacle avoidance) and object-of-interest detection. To acheive both of these ends, the BWIBots can make use of a configurable set of sensors. In this section, we briefly outline various combinations of sensors used for both purposes.

Certain key requirements need to be met by the sensor suite responsible for localization and obstacle avoidance. The sensors should have a sufficiently large horizontal field of view for robust robot localization, and some vertical field of view is also necessary to prevent the robot from crashing into concavely shaped objects. For instance, only the central column of an office chair may be visible to a robot with a 2D planar LIDAR. A 3D sensor, or a 2D sensor on a servo, is necessary to sense other parts of these objects in order to avoid them.

Furthermore, the sensor suite may need to detect landmarks at long distances for robust robot localization, especially in large open areas. Finally, direct or reflected sunlight may affect LIDAR or RGBD sensors, and it is useful to have a sensor resistant to being affected by sunlight for robust operation near glass windows. In Table 1, we outline the performance of some combination of sensors that have been used on the BWIBot platform, in increasing order of cost.

Table 1. Various sensors and combinations used for navigation and localization on the BWIBot in increasing order of cost. The URG-04 and UST-20 are 2D LIDARs available from Hokuyo, and the VLP-16 is a 3D LIDAR from Velodyne.

Sensors	Sufficient HFOV	Sufficient VFOV	Sufficient Range	Sunlight Resistant
Kinect	No (60°)	Yes (40°)	No (4m)	No
URG-04	Yes (240°)	No	No (4m)	No
Kinect + URG-04	Yes (240°)	Yes (40°)	No (4m)	No
UST-20	Yes (270°)	No	Yes (20m)	No
Kinect + UST-20	Yes (270°)	Yes (40°)	Yes (20m)	No
VLP-16	Yes (360°)	Yes (30°)	Yes (60m)	Yes

While the VLP-16 satisfies all the requirements outlined in Table 1, its minimum range (45cm) creates a blind spot around the robot body (radius = 30cm). This blind spot can be eliminated with an additional URG-04 sensor, which is undesirable. In our opinion, the ideal sensor (or combination) for an indoor robot needs to have all the properties satisfied by the VLP-16 in Table 1, as well as

having a minimum range of 20cm or less, while not being prohibitively expensive.

For person and object detection, three different sets of sensors have been used:

1. PointGrey BlackFly GigE camera - This camera is mounted on a pan-tilt unit constructed using Dynamixel MX-12W servos, and is useful for collecting video data in high-resolution. It has primarily been used for detecting objects using SIFT visual features (Lowe 2004).
2. KinectV1 - The KinectV1 sensor was used for detecting people in 3D point clouds. For person detection, we used the method of Munaro and Menegatti (2014), as implemented in the Point Cloud Library (Rusu and Cousins 2011). While the implementation provides reasonable accuracy, the detection frame rate is low (about 4Hz when concurrently run with other BWIBot software).
3. KinectV2 - The Microsoft SDK with the KinectV2 allows for extremely fast and robust person detection. The raw data from the Kinect is processed via the SDK running on a Microsoft Surface Pro separate from the primary robot computer.

3.5 Mobile Manipulation

One BWIBot incorporates a Kinova MicoV1 6-DOF arm for manipulation. The Mico arm was chosen primarily because it is safe to operate around humans. Specifically, the arm includes force sensors in each joint which enable it to be software-complaint when interacting with humans. In addition, the force sensors allow the arm to perform various manipulation tasks, such as drawing on a board with a marker and handing off objects to humans.

4 Software

In the previous section, we described the hardware design choices that went into constructing the BWIBots. Next, we describe the software architecture used on the BWIBots, which has been built on top of the Robot Operating System (ROS) middleware framework (Quigley et al. 2009). ROS provides abstractions for data formats commonly used in robotics, along with message passing mechanisms allowing different software modules on a robot, as well as multiple robots, to communicate with one another.

An overview of the software architecture is illustrated in Figure 2. The robot can be controlled at many different levels of control, where each level balances the granularity of control with the robot's autonomy. This architecture has been designed in a hierarchical manner, as different research applications require different granularities of control. Specifically, the software architecture provides five hierarchical levels of control:

Velocity Level Control: The robot has no autonomy, and is controlled directly via linear and angular velocities.

Navigation Level Control: The robot is given a physical location and orientation as a destination in Cartesian space (x, y, θ), and the robot autonomously navigates to this destination while avoiding obstacles.

High-Level Action Control: At this level of control, the robot can execute navigation actions to symbolic locations. For instance, the robot can be instructed to autonomously navigate to a specific door without requiring specification of the door's location in Cartesian space. Furthermore, at this level the robot also provides some tools for interacting with humans, such as a GUI, speech synthesis, and speech recognition.

Planning Level Control: The robot can achieve high-level goals, such as those that require it to navigate to a different part of the building via doors and elevators using a sequence high-level actions.

Multi-Robot Control: This level of control allows multiple robots to be controlled at any one of the four previously mentioned levels using a centralized server.

In the following subsections, we describe the modules that comprise the software architecture and how these modules can be used to achieve the aforementioned hierarchical levels of control.

4.1 Map Server

For the robot to navigate autonomously, it requires a map of the world. Standard ROS Navigation is designed to allow a robot to navigate using a single 2D grid map (Marder-Eppstein et al. 2010), and these maps can be built using Simultaneous Localization and Mapping (SLAM) approaches such as GMapping (Grisetti et al. 2007). While a single grid map is sufficient to allow an intelligent service robot to perform navigation on a single floor inside a building, it has the following limitations:

1. Without semantic information encoded within a grid map, autonomous navigation cannot be performed using symbolic locations. For instance, a user cannot request the robot to navigate to a particular room by name only.
2. Navigation based on a single 2D map does not work if the robot is required to use an elevator to navigate to a different floor.

The software architecture overcomes these limitations without modifying the existing ROS Navigation stack. We implement a MultiMap Server that contains all 2D maps necessary to perform navigation across all floors

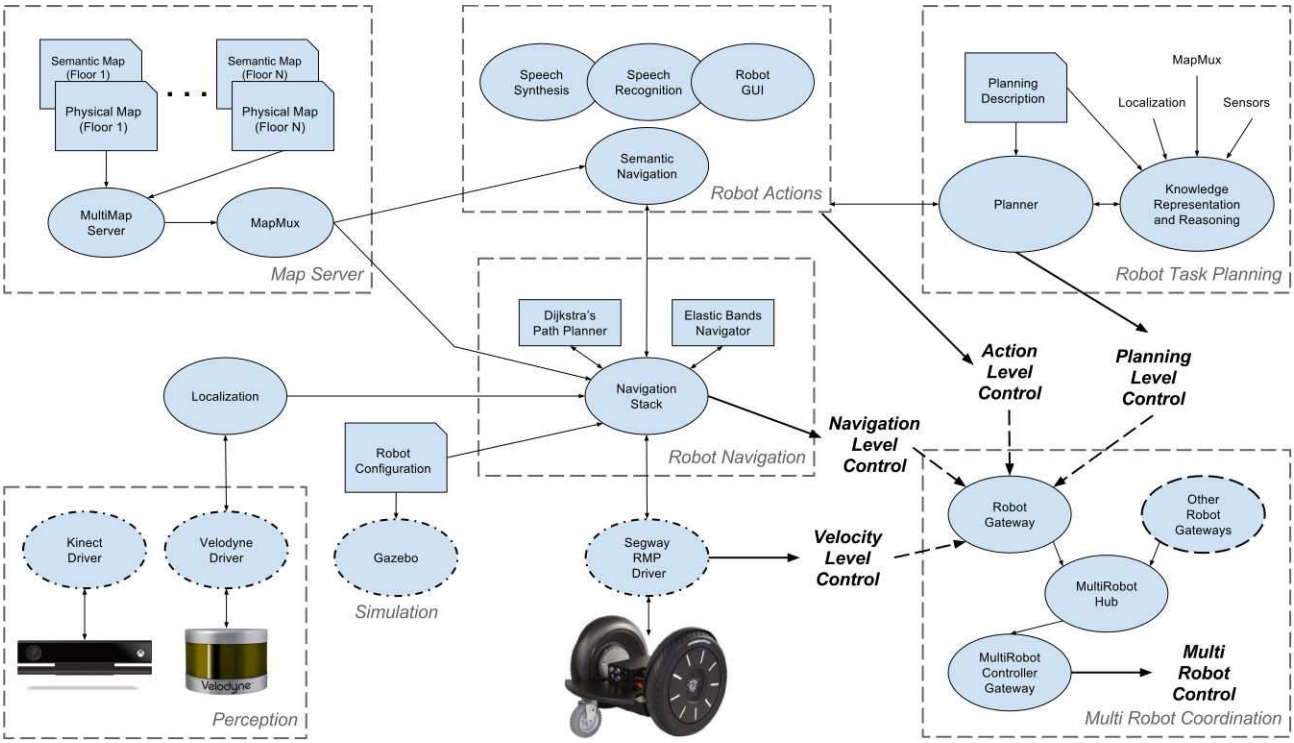


Figure 2. The software architecture for the BWIBots. The figure depicts all the various software modules and how they are connected, implementing the various levels of control used by different research applications.

of the building. The correct map is selected using a multiplexer node (MapMux), which is then passed to the ROS Navigation Stack. Should the robot change floors, navigation is reinitialized with the correct map using this multiplexer node.

The MultiMap Server also adds secondary semantic maps to each floor alongside the physical maps. These maps contain information such as the symbolic names of all doors, a mapping from physical to symbolic locations, and the physical locations of objects of interest in the environment (such as printers). There has been previous research on how this semantic information should be attached to a physical map (Bastianelli et al. 2013) while the physical map is being built. In contrast, we use a simple tool that allows manual yet quick labeling of semantic information after the physical map has been constructed.

4.2 Perception

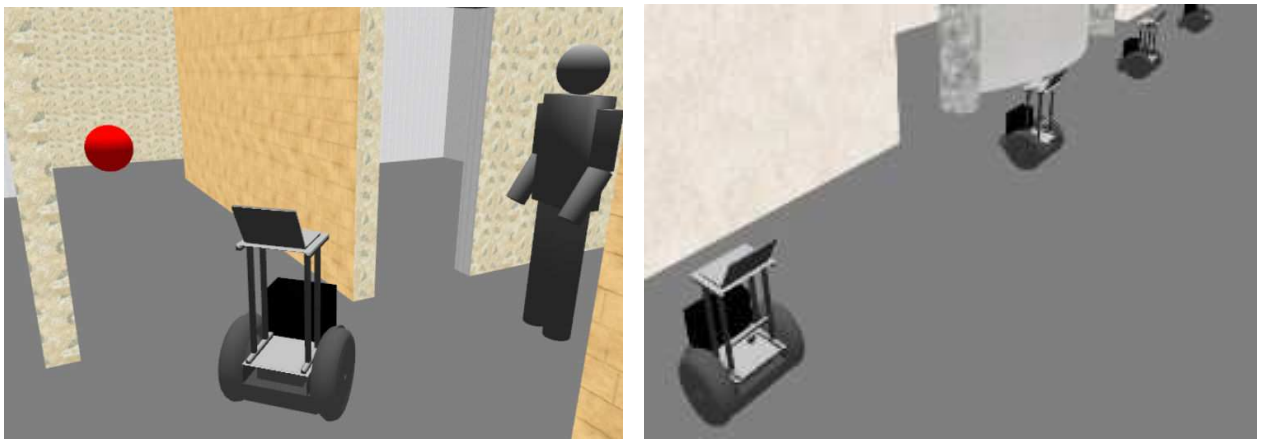
The choice of physical sensors on the BWIBots has already been discussed in Section 3.4. The perception module is responsible for providing sensory information in the common data abstractions used by ROS, as well as filtering raw sensor data. For example, any points returned by the depth sensors described in Section 3.4 that belong to the chassis of the robot are filtered out. An additional filter also updates raw sensor data to remove any potential stale obstacle readings constructed from previous sensor data.

4.3 Simulation

We have developed 3D simulation models for the BWIBots using Gazebo (Koenig and Howard 2004), allowing us to run simulations with one or many robots, as shown in Figure 3. The focus of this module is not to accurately simulate the dynamics of the robot, but rather to provide a platform for testing various single-robot and multi-robot applications. Consequently, in order to speed up the simulation, especially when multiple robots are being reproduced, we use an extremely low fidelity model of the robot that ignores the dynamics of the wheels and simulates the entire collision model of the robot as a cylinder. It then applies simple lateral forces to the robot to emulate real motion in the environment, allowing the simulation to run many times faster than real time. In contrast, the visualization of the robot continues to use an accurate high-fidelity model, allowing demonstrations to look realistic.

4.4 Robot Navigation

While the BWIBots can be controlled directly via *velocity level control*, most applications require the BWIBot platform to at least be able to autonomously navigate to



(a) Human avatar interacting with simulated robot

(b) Multi-robot simulation

Figure 3. Figure 3a demonstrates a robot guiding a human-controlled avatar to the red ball (Khandelwal and Stone 2014). Figure 3b depicts multiple robots being simulated within a single environment.

a given physical location within a 2D map. This second control layer, called the *navigation level control*, can be provided using a more sophisticated autonomous navigation system built on top of the velocity level control.

Autonomous navigation on the BWIBots is built using the ROS Navigation stack (Marder-Eppstein et al. 2010). The ROS navigation stack keeps track of the obstacles in the environment using an occupancy grid representation. Given the current locations of obstacles, it makes use of a global planner to find a path to a desired destination. It then uses a local planner to compute linear and angular velocities that need to be executed by the robot to approximately follow the global path while avoiding obstacles.

In our instantiation of the navigation stack, Dijkstra's algorithm is used to find a path to a destination, and low-level control is implemented via the Elastic Bands approach (Quinlan and Khatib 1993). This approach makes use of active contours (Kass et al. 1988) to execute local control that balances the straightness of the executed path with the distance of obstacles to this path.

The navigation stack also needs to estimate the position of the robot for navigation, and uses Adaptive Monte Carlo Localization (AMCL) (Fox et al. 1999) for robot localization. In this approach, the distribution of possible locations the robot may be in is represented via samples called particles, and the mean of this distribution gives the current estimate of the location of the robot.

4.5 High-Level Robot Actions

In many research applications, it is useful to have the robot interact with the environment without specifying low-level details. For instance, an algorithm may call for executing a sequence of actions using symbolic instructions, such as approach door d_1 and go through it, rather than specifying physical locations for the robot to navigate to. The third level of control in the software architecture provides this functionality, which is termed the *high-level action control*. At this level, symbolic navigation instructions to the robot can be specified to the robot; this level is built on top of *navigation level control*.

At this layer, the robot can also perform a number of actions that require human interaction. A GUI built using Qt[¶] allows displaying text and images to the user, as well as asking text or multiple choice questions. Speech recognition using Sphinx (Walker et al. 2004) and speech generation using Festival (Taylor et al. 1998) are also available at this layer, allowing interaction via spoken natural language.

4.6 Robot Task Planning

Given the ability to perform various high-level actions, sequences of such actions can be constructed to achieve high-level goals. For instance, the robot may need to deliver an object to person p_1, but may not know p_1's location. However, it may know that it can acquire p_1's location by asking person p_2. Achieving this goal requires multiple symbolic navigation actions, as well as use of the GUI and speech recognition/generation actions to interact with people. Furthermore, to achieve these high-level goals, the robot needs to track knowledge about the environment, such as the location of person p_2. Such information is stored within a knowledge base on the robot, and is used both for planning and for reasoning about the environment. In this

[¶] http://www.qt.io/

section, we describe the module responsible for knowledge representation, reasoning, and planning, which provides the fourth control layer on the robot, called *planning level control*.

The module for symbolic reasoning and decision making is composed of two processes (ROS nodes), one responsible for managing knowledge on the robot, and the other for overseeing action execution. The Knowledge Representation and Reasoning (KRR) node handles the knowledge base and provides access to it from outside of the module. Other nodes can request updates to the knowledge base or retrieve information about the current state. The planner node manages the execution, generates planning queries, and monitors the outcome of actions at run time. The planner can receive planning tasks to be carried out from other nodes, and uses the robot's action-level control to execute the sequence of actions necessary to complete the task. Since this module provides a layer of high-level intelligence and is relatively non-standard, we elaborate on it in more detail than the other modules.

The symbolic knowledge representation is based on Answer Set Programming (Lifschitz 2008), and the system delegates the actual automated reasoning to the answer set solver CLINGO (Gebser et al. 2011). The module and the reasoner exchange information through ASP files containing the knowledge base, the queries, and the output of the reasoning process. In section 5, we discuss how knowledge can be described using action language $\mathcal{BC}$, and we compare against other related approaches for planning and knowledge representation therein.

At the heart of the module, shared by both nodes, is the ACTASP library[‖]. ACTASP abstracts the syntax of answer set programming and the parameters of the reasoner (in our case CLINGO, but interfaces to other reasoners can be seamlessly implemented). It implements and makes available reasoning and planning to the rest of the system in the following ways:

Current State Inquiry: Other modules may require verification of whether the knowledge base entails a specific piece of information at the current time: in other words, whether the robot currently knows something in particular. Such queries are the simplest ones, and are just forwarded to the underlying reasoner.

KB Update: Updates to the knowledge base are performed in two steps, and they make use of the model of the system described by the planning description to ensure that the knowledge base is not left in an inconsistent state after the update. In the first step, the reasoner is invoked to simulate the special action NOOP, which does not actively modify the current state, but allows the default dynamics of the system to update the fluents as predicted by the model under no action. Most fluents are just carried over by inertia, meaning that they do not change between subsequent time steps, but others may change simply due to the passage of time. For instance, if the model predicted that a door would close by itself if not held open, then the door would be assumed closed after the execution of NOOP. ACTASP then generates a query containing the new observations as part of the next state. If the query is satisfiable, the second step is to incorporate the new observations into the new current state. If the query is unsatisfiable, on the other hand, the observations conflict with the prediction of the system model and must be discarded. An example of an unacceptable observation is one in which the robot is at two locations at the same time, which can arise if the robot localization jumps from one location to another. The model does not allow such a possibility, and the query to generate the next state would be unsatisfiable.

Planning: Planning is a classic type of reasoning in which a query is satisfied if there exists a sequence of actions that starts in the current state and ends in a state that satisfies a goal condition. ACTASP implements, alongside the classic notion of a planner, the notion of a *multi-planner*, that is a planner that returns not just one plan but all the plans which reach the goal in a given number of actions. These plans can be used by an appropriate action executor to have several options in case one should fail, or to learn which one of the available paths is optimal according to a user-specified criterion.

Monitoring: Execution monitoring is traditionally associated with verifying that the current sequence of actions being followed still achieves the original task. In ACTASP, monitoring is implemented through a query which appends the remaining sequence of actions in the plan to the original planning query. The reasoner will be able to satisfy the query if and only if the remaining plan can lead the agent to a goal state. This is a looser condition than having the prediction on the outcome of the last action verified, since the action may actually have given an unpredicted outcome, while the rest of the plan could still be valid. For example, during action execution the robot may have noticed unexpected changes in the environment and have updated the knowledge base in response. Even if the resulting next state is not the sole effect of the application of the last action, if the new changes do not disrupt the rest of the plan, the monitoring query will still report the plan to be

[‖] https://github.com/mleonetti/actasp

valid. This robustness is of great practical importance since, without it, if the environment is inhabited by humans, the inevitable continual changes would also continually trigger computationally expensive replanning.

The ACTASP library also provides two types of action executors: a Replanning Action Executor and a Learning Action Executor. The replanning action executor has a simple, intuitive behavior. It uses an underlying planner to generate a plan, then requests the execution of the actions to the rest of the system, while monitoring the validity of the plan between one action and the next. As previously mentioned, the only planner currently implemented uses the answer set solver itself, but any other planner can be interfaced with the library. If the remaining plan appears to be invalid, the executor uses the planner to generate a new plan from the current state. A solution also provided by the library is a planner called Any Plan, which uses an underlying *multi-planner* to generate all plans of a maximum length and returns a random one. This behavior allows the robot to randomly explore several possible paths in the case of being stuck on a plan that keeps failing. As with the planner, the only *multi-planner* currently implemented is based on the answer set solver CLINGO, but other implementations are possible.

The Learning Action Executor is more sophisticated. It makes use of an underlying *multi-planner* to generate a number of options, and then it learns from experience, through reinforcement learning, the *value* of each action in every encountered state (Leonetti et al. 2016). Given a cost function for the actions, the value of an action in a given state is the expected total cost incurred by taking the action and acting optimally afterwards. Through this mechanism, the learning executor improves the robot's efficiency, over time, at reaching the goals that are repeatedly requested. The cost function can be anything the user intends to minimize: time, energy, interactions with users, action failures, etc. In our system, we use the action execution time, so that the robot learns to minimize the total time taken to reach the goals.

4.7 Multi Robot Coordination

The software components described up to this point are sufficient to enable robust autonomous control of an *individual* robot. However, we have not addressed any of the issues that arise when multiple robots are operating in the same environment. In particular, the core ROS infrastructure does not support robust multi-robot communication and coordination. We therefore make use of the *RObotics in CONcert* (ROCON) ROS modules to enable centralized control over multiple BWIBots (Stonier et al. 2015).

This multi-robot coordination framework introduces the fifth and final layer available for controlling the robots: *multi-robot control*. Using this framework, it is possible to execute any one of the other (single-robot) layers of control on multiple robots.

4.8 Summary

Sections 3-4 describe the hardware and software design choices behind the BWIBots. All the software outlined in this section is available open-source**. Next, we summarize a set of representative research applications that have utilized this platform. These research contributions interface with the software architectures using different modules and control levels.

5 Planning using Action Language $\mathcal{BC}$

In Section 4.6, we explained how the planning module is implemented, but did not explain how the knowledge contained within the robot is described, nor how action effects are encoded. These descriptions are necessary for the robot to perform planning and reasoning. In this section, we briefly describe how action language $\mathcal{BC}$ (Lee et al. 2013) can be used for constructing a general purpose planning description for robot task planning (Khandelwal et al. 2014). Prior to this work, action language $\mathcal{BC}$ had not been used for robot task planning. Thus, this section summarizes one of the main research contributions that utilizes the BWIBots.

General purpose planning domain descriptions can be written using various modes. Action languages such as $\mathcal{BC}$ are attractive in task planning for mobile robots because they solve the *frame problem*, which states that many axioms are necessary to express that things in the environment do not change arbitrarily (McCarthy and Hayes 1969). For example, when a robot picks up an object from the table, it does not change the location of a different object on the table. $\mathcal{BC}$ solves this problem by easily expressing rules of inertia. In addition, $\mathcal{BC}$ can solve the *ramification problem*, which is concerned with the indirect consequences of an action (Finger 1986). For example, when a robot picks up a tray from the table, it indirectly changes the location of any object on the tray. $\mathcal{BC}$ can also easily express indirect and recursive effects of actions.

Existing tools such as COALA (Gebser et al. 2010) and CPLUS2ASP (Babb and Lee 2013) allow us to translate $\mathcal{BC}$ action descriptions into logic programs under answer set semantics (Gelfond and Lifschitz 1988, 1991), and planning can be accomplished using computational methods of ASP (Marek and Truszczynski 1999; Niemelä 1999).

**https://github.com/utexas-bwi/

In this section, we demonstrate how action language $\mathcal{BC}$ can be used for robot task planning in domains requiring planning in the presence of missing information and indirect/recursive action effects. While we demonstrate using $\mathcal{BC}$ to express a mail collection task, the overall methodology is applicable to any other planning domains that require: recursive and indirect action effects, defeasible reasoning, and acquiring previously unknown knowledge through human-robot interaction. In addition, we also demonstrate how answer set planning under action costs (Eiter et al. 2003) can be applied to robot task planning in conjunction with $\mathcal{BC}$.

Before we describe how $\mathcal{BC}$ is used to construct a general purpose planning description, we briefly discuss other related approaches for solving the same problem. Task planning problems for mobile robots have also been described using the Planning Domain Definition Language (PDDL) (Quintero et al. 2011), which are then solved using planning algorithms such as Fast-Forward (Hoffmann and Nebel 2001) and Fast-Downward (Helmert 2006). While PDDL has primarily been used with an emphasis on *efficient* plan generation, it has rarely been used in domains with many indirect or recursive action effects[††], or in domains where defeasible reasoning is necessary for succinct expressivity. In such domains, $\mathcal{BC}$ provides a viable alternative.

Apart from PDDL, action language $\mathcal{C}+$ (Giunchiglia et al. 2004) has also been used for robot task planning (Caldiran et al. 2009; Chen et al. 2010; Chen, Jin and Yang 2012; Erdem and Patoglu 2012; Erdem et al. 2013; Havur et al. 2013). Unlike $\mathcal{BC}$, $\mathcal{C}+$ cannot encode recursive action effects. In addition, most of these existing applications do not consider knowledge acquisition, i.e. they assume that all the information necessary for planning is available in the initial state, and do not consider action costs. Recent work improves on existing ASP approaches for robot task planning by incorporating a constraint on the total time required to complete the goal (Erdem et al. 2012). While this previous work attempts to find the shortest plan that satisfies the goal within a prespecified time constraint, our work attempts to explicitly minimize the overall cost to produce the optimal plan.

5.1 Describing domains in $\mathcal{BC}$

The action language $\mathcal{BC}$, like other action description languages, describes dynamic domains as transition systems. A full description of $\mathcal{BC}$ can be found in Lee et al. (2013). Information about the state of the world is expressed using fluents, and each fluent has a finite domain. An *action description* in $\mathcal{BC}$ is a finite set consisting of dynamic and static laws. Dynamic laws represent how the values of fluents and actions in the current time step affect fluents in the next time steps, whereas static laws incorporate how fluents affect other fluents within the current time step.

In this section, we describe a small yet representative set of $\mathcal{BC}$ laws that can be used to express such a domain. These rules are not designed to completely represent the operation of a mobile robot, and a more elaborate description is available in Khandelwal et al. (2014). In this domain, a robot needs to collect outgoing mail (intended for delivery) from building residents. Furthermore, it has limited battery life and must recharge its battery before it runs out to continue operation. The floor plan for this building is illustrated in Figure 4. *alice*, *bob*, *carol* and *dan* are people who inhabit the building. o_1, o_2, o_3, lab_1, and *cor* are rooms in the building, connected via doors d_1, d_2, d_3, d_4, and d_5.

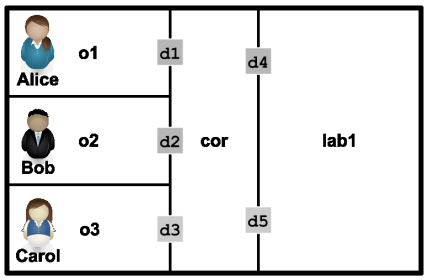


Figure 4. The layout of the example floor plan used in the text, along with depictions of the locations of Alice, Bob, and Carol and the robot charger. The location of Dan is not initially known.

Facts about the structure of the building can be easily represented in $\mathcal{BC}$. For instance, the following laws express which rooms have doors, and that two rooms are accessible to each other if they share the same door. In these laws, we use meta-variables R, R_i and D, D_i to refer to rooms and doors, respectively. Furthermore the **default** keyword is used to refer to defeasible reasoning.

default $\sim hasdoor(R, D)$.
$hasdoor(o_1, d_1)$. $hasdoor(o_2, d_2)$. $hasdoor(o_3, d_3)$.
$hasdoor(lab_1, d_4)$. $hasdoor(lab_1, d_5)$.
default $\sim acc(R_1, D, R_2)$.
$acc(R_1, D, R_2)$ **if** $hasdoor(R_1, D)$, $hasdoor(R_2, D)$.
$acc(R_1, D, R_2)$ **if** $acc(R_2, D, R_1)$.

Additionally, a robot can only approach a door in the same room as itself, and it can go through this door once it is adjacent. These navigation actions can only be performed if the robot has sufficient battery and makes use of the semantic navigation node. Action preconditions are imposed by making actions invalid if these preconditions

[††]The use of PDDL axioms allows PDDL to encode indirect and recursive action effects (Thiébaux et al. 2003), but this feature is typically not tested in the International Planning Competition, where different PDDL solvers are evaluated.

are not met, using the **nonexecutable** keyword.

$approach(D)$ **causes** $beside(D)$.
nonexecutable $approach(D)$ **if** $loc = R$, $\sim hasdoor(R, D)$.
nonexecutable $approach(D)$ **if** $beside(D)$.
nonexecutable $approach(D)$ **if** $battery = 0$.

$gothrough(D)$ **causes** $\sim beside(D)$.
$gothrough(D)$ **causes** $loc = R_2$ **if** $loc = R_1$, $acc(R_1, D, R_2)$.
nonexecutable $gothrough(D)$ **if** $\sim beside(D)$.
nonexecutable $gothrough(D)$ **if** $battery = 0$.

We also need to encode the change in battery life as time progresses, and the following example demonstrates how $\mathcal{BC}$ uses defeasible reasoning to express the change in battery state without affecting other actions, and how the battery can be recharged using the *recharge* action.

default $battery = max(a - 1, 0)$ **after** $battery = 0$.
$recharge$ **causes** $battery = 5$.
nonexecutable $recharge$ **if** $loc \neq lab_1$.

Note that the above example is simplistic, and the update rule can update the battery state based on the passage of time and the time spent by the robot recharging. Next, we encode whether a robot knows the location of a person P, ensuring that the robot does not believe that a person is in two rooms at the same time. Additionally, we assume that a person's location remains the same in the next time step, using the **inertial** keyword.

default $\sim inside(P, R)$.
$inside(alice, o_1)$. $inside(bob, o_2)$. $inside(carol, o_3)$.
inertial $inside(P, R)$.
$\sim inside(P, R_2)$ **if** $inside(P, R_1)$, $R_1 \neq R_2$.

If the robot knows where person P is, it can collect mail from that person using the *collectmail* action. If another person P_2 passed their mail to P, then P_2's mail is collected as well, which is a recursive indirect action effect of the *collectmail* action:

$collectmail(P)$ **causes** $mailcollected(P)$
$mailcollected(P_2)$ **if** $mailcollected(P)$, $passto(P_2, P)$.
nonexecutable $collectmail(P)$ **if** $loc = R$, $\sim inside(P, R)$.

5.2 Planning using $\mathcal{BC}$ Description

Given a $\mathcal{BC}$ description, planning is performed as described in Section 4.6. During execution, should the robot not know the location of person P, it can ask person P_1 for P's location. The *askploc* action asks person P's location from person P_1:

$askploc(P_1, P)$ **causes** $inside(P, R)$ **if** $loc = R$.
nonexecutable $askploc(P_1, P)$ **if** $loc = R$, $\sim inside(P_1, R)$.

For planning purposes, it is assumed that P's location is the same as that of the robot. During execution, person P_1 should return the true location of P, which is then used to update the knowledge base. Should the location of P be different from the robot's current location, execution monitoring determines the remaining plan is invalid, and replanning then determines a plan that considers person P's correct location.

Planning using $\mathcal{BC}$ can be computationally expensive, especially when the total plan cost is minimized instead of the number of actions. It is possible to use multiple domain abstractions in $\mathcal{BC}$, where each description encodes a different level of detail and hierarchical planning techniques can speed up planning time (Zhang, Yang, Khandelwal and Stone 2015). Hierarchical planning requires some modifications to task planning module presented in Section 4.6, such that planning is performed across multiple layers of the domain abstraction hierarchy, and is not covered in this article.

5.3 Experimental Results

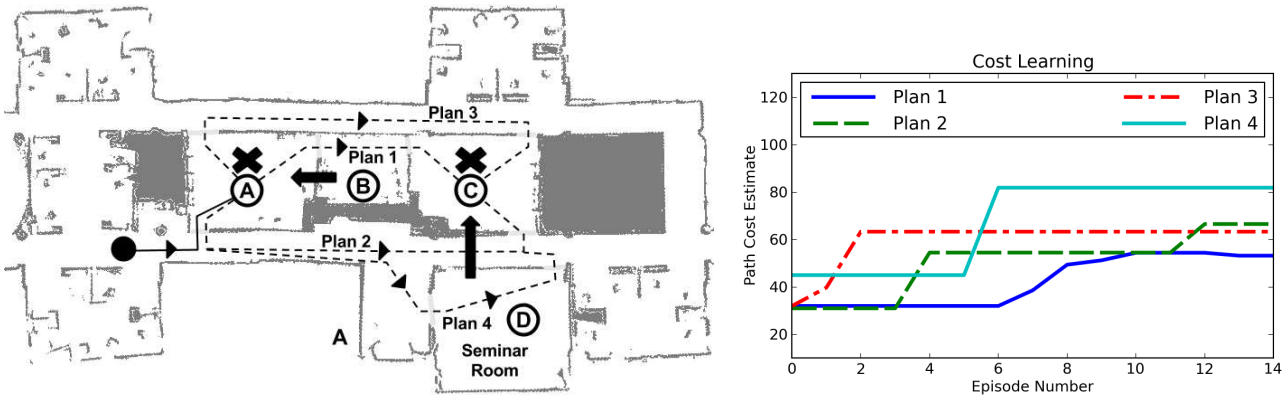
We demonstrate a simple experiment that performs cost-based planning on a BWIBot while learning these action costs on the fly. The goal of this experiment is to learn actions costs sufficiently well enough that cost-based planning always chooses the optimal plan. The real world domain contains 5 rooms, 8 doors, and 4 people from whom mail has to be collected, and is illustrated in Figure 5a. Two people have passed mail such that the robot only needs to visit a total of 2 people to collect everyone's mail.

We present the cost curves of 4 different plans in Figure 5b, where Plan 1 is optimal. In this experiment, the robot starts in the middle of the corridor while not beside any door as shown in Figure 5a. The learning curves show that the planner discovers by the episode 12 that plan 1 is optimal. After the optimal plan is found, no other plans are selected for execution and their costs do not change.

In this section, we demonstrated how action language $\mathcal{BC}$ can be used to describe general purpose planning descriptions, and demonstrated how such a description can be used by the BWIBots. Using action language $\mathcal{BC}$ allows us to easily formalize indirect effects of actions on recursive fluents, as well as default knowledge.

6 Incorporating Uncertainty into Planning

In the previous section, we discussed how a robot could achieve a goal by executing multiple high-level actions on the BWIBots. While action language $\mathcal{BC}$ can express defeasible reasoning, it cannot express probabilities, and consequently cannot be used for stochastic planning. In the research contribution summarized in this section, we introduce a method for robots to efficiently and robustly fulfill service requests in human-inhabited environments by simultaneously reasoning about commonsense knowledge expressed using defeasible reasoning and computing plans under uncertainty. We illustrate this planning paradigm using a Spoken Dialog System (SDS), where the robot



(a) Floor plan **(b)** Learning Results

Figure 5. The real world domain contains 5 rooms, 8 doors, and 4 people from whom mail has to be collected. The filled circle marks the robot's start position, the crosses mark the people who have all the mail (A, C), and the arrows mark how mail was recursively passed to them. The 4 plans compared in Figure 5b are also marked on the floor plan.

identifies a spoken shopping request from the user in the presence of noise and/or incomplete instructions. The goal of the system is to identify the shopping request as quickly as possible while minimizing the cost of asking questions. Once confirmed, the robot attempts to deliver the item as explained in Section 4.6. While this planning paradigm is described in the context of an SDS, it can just as easily be applied to other stochastic planning problems as well.

Commonsense knowledge is the knowledge that is *normally* true but not always, e.g., office doors are closed during holidays and people prefer coffee in the mornings. Logical commonsense knowledge needs to be expressed via defeasible reasoning, and probabilistic commonsense knowledge needs to be expressed via probability distributions. In parallel with commonsense reasoning, robots frequently need to compute a plan including more than one action to accomplish tasks that cannot be completed through single actions. To do so, it is necessary to model the uncertainty in the robot's local, unreliable observations and nondeterministic action outcomes while planning toward maximizing long-term reward.

In this section, we describe the CORPP (COmmonsense Reasoning and Probabilistic Planning) algorithm (Zhang and Stone 2015). While commonsense reasoning and planning under uncertainty have been studied separately, CORPP, for the first time, exploits their complementary features by integrating POMDPs and P-LOG (Baral et al. 2009) and enables robots to simultaneously reason about both logical and probabilistic commonsense knowledge and plan toward maximizing long-term reward under uncertainty.

Different methods have been developed to combine commonsense reasoning and probabilistic planning. For instance, Zhang, Sridharan and Wyatt (2015) combined ASP and POMDPs for integrating logical reasoning and probabilistic planning, but bridging the gap between answer sets (i.e., the reasoning results of ASP) and POMDP beliefs requires significant domain knowledge. Hanheide et al. (2015) used a switching planner for deterministic and probabilistic planning and used commonsense knowledge for diagnostic tasks and generating explanations. In contrast, CORPP is an algorithm that integrates commonsense reasoning and probabilistic planning while exploiting their complementary features in a principled way. Young et al. (2013) have reviewed existing techniques and applications of POMDP-based SDSs, and, similar to other POMDP applications, such SDSs are ill-equipped to represent and reason with commonsense knowledge.

Before we describe the CORPP algorithm and present an experimental evaluation, we briefly discuss the logic programming language P-LOG used within the algorithm.

6.1 Background

In this subsection, we briefly introduce logic programming languages ASP and P-LOG. P-LOG is a probabilistic extension of ASP. More detailed descriptions of ASP and P-LOG are available by Gelfond and Kahl (2014). An ASP program can be described using a set of rules of the form:

$$\mathtt{l_0\ or\ \cdots\ or\ l_k} \leftarrow \mathtt{l_{k+1}, \cdots, l_m,\ not\ l_{m+1}, \cdots,\ not\ l_n}.$$

where $\mathtt{l}$'s are expressions of the form $\mathtt{p}(\bar{\mathtt{t}}) = \mathtt{true}$ or $\mathtt{a}(\bar{\mathtt{t}}) = \mathtt{y}$. Symbol $\mathtt{not}$ is a logical connective called *default negation*; $\mathtt{not\ l}$ is read as "it is not believed that $\mathtt{l}$ is true", which does not imply that $\mathtt{l}$ is believed to be false. E.g., $\mathtt{not\ prof(alice)}$ means it is unknown that $\mathtt{alice}$ is a professor. A rule is separated by the symbol "←". The left side is called the *head* and the right side is called the *body*. A rule is read as "head is true if body is true".

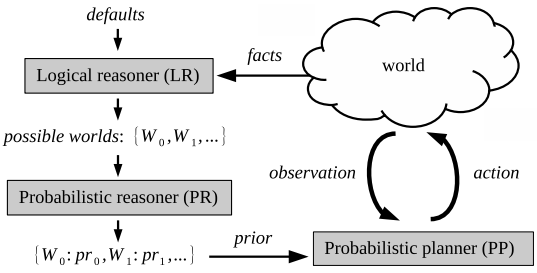


Figure 6. Overview of algorithm CORPP for combining commonsense reasoning with probabilistic planning

Default negation is used in ASP to express defeasible reasoning. For instance, the rule: $\mathtt{p(X)} \leftarrow \mathtt{c(X)}, \mathtt{not}\ \neg\mathtt{p(X)}.$ expresses that if object X has attribute c, it is believed that X has attribute p unless there is evidence to the contrary. Inertia can be expressed similarly.

Probabilistic extensions of ASP have been developed for enabling both logical and probabilistic reasoning using a single set of syntax and semantics, such as P-LOG (Baral et al. 2009). P-LOG allows *random selections*– saying that if B holds, the value of $\mathtt{a}(\bar{\mathtt{t}})$ is selected randomly from the set $\{\mathtt{X} : \mathtt{q(X)}\} \cap \mathtt{range(a)}$, unless this value is fixed elsewhere:

$$\mathtt{random}(\mathtt{a}(\bar{\mathtt{t}}) : \{\mathtt{X} : \mathtt{q(X)}\}) \leftarrow \mathtt{B}.$$

where B is a collection of extended literals and q is a predicate. P-LOG also allows directly specifying probabilities using *probability atoms* (or *pr-atoms*):

$$\mathtt{pr}(\mathtt{a}(\bar{\mathtt{t}}) = \mathtt{y} | \mathtt{B}) = \mathtt{v}.$$

that states if B holds, the probability of $\mathtt{a}(\bar{\mathtt{t}}) = \mathtt{y}$ is v with $\mathtt{v} \in [0, 1]$. In this work, we use P-LOG for commonsense reasoning.

6.2 The CORPP algorithm

Before introducing the CORPP algorithm, it is necessary to classify domain attributes based on their observability. If an attribute's value can only be observed using sensors, we say this attribute is partially observable. For instance, *current location* (of a robot) is partially observable, because self-localization relies on sensors. The values of attributes that are not partially observable can be specified by facts, defaults, or reasoning with other attributes' values. For instance, the value of attribute, *is it within working hours now*, can be inferred from *current time*. Similarly, identities of people as facts can be available but not always. The value of an attribute can be *unknown*.

We propose algorithm CORPP for reasoning with commonsense and planning under uncertainty, as shown in Figure 6. The *logical reasoner* (LR) includes a set of logical rules in ASP and takes defaults and facts as input. The facts are collected by querying internal memory and databases. It is possible that facts and defaults try to assign values to the same attributes, in which case default values will be automatically overwritten by facts. The output of LR is a set of possible worlds $\{W_0, W_1, \cdots\}$. Each possible world, as an answer set, includes a set of literals that specify the values of attributes—possibly unknown.

The *probabilistic reasoner* (PR) includes a set of random selection rules and probabilistic information assignments in P-log and takes the set of possible worlds as input. Reasoning with PR associates each possible world with a probability:

$$\{W_0 : pr_0,\ W_1 : pr_1, \cdots\}$$

Unlike LR and PR, the *probabilistic planner* (PP), in the form of a POMDP, is specified by the goal of the task and the sensing and actuating capabilities of the agent. The prior in Figure 6 is in the form of a distribution and denoted by α. The ith entry in the prior, α_i, is calculated by summing up the probabilities of possible worlds that are consistent with the corresponding POMDP state s_i. In practice, α_i is calculated by sending a P-log query of this form:

$$?\{\mathtt{s_i}\} | \mathtt{obs}(\mathtt{l}_0), \cdots, \mathtt{obs}(\mathtt{l}_\mathtt{m}), \mathtt{do}(\mathtt{l}_{\mathtt{m}+1}), \cdots, \mathtt{do}(\mathtt{l}_\mathtt{n}).$$

where l's are facts. If a fact l specifies the value of a random attribute, we use obs(l). Otherwise we use do(l). do(l) adds l into a program before calculating the possible worlds, while obs(l) is used to remove the calculated possible worlds that do not include literal l.

The prior is used for initializing POMDP beliefs in PP. Afterwards, the robot interacts with the world by continually selecting an action, executing the action, and making observations in the world. A task is finished after falling into a terminating state.

CORPP is fully implemented and tested on a shopping request identification problem. In a campus environment, the shopping robot can buy an item for a person and deliver to a room, so a shopping request is in the form of $\langle$*item, room, person*$\rangle$. A person can be either a *professor* or a *student*. Registered students are authorized to use the robot for free, and professors need to pay for the service of using the robot. The robot has access to a database to query about registration and payment information, but the database may be incomplete. The robot can initiate spoken dialog to gather information for understanding shopping requests and take a delivery action when it becomes confident in the estimation. This task is challenging for the robot because of its imperfect speech recognition ability. The goal is to identify shopping requests, e.g. $\langle$*coffee, office*1*, alice*$\rangle$, efficiently and robustly.

The following two logical reasoning rules state that professors who have paid and students who have registered are authorized to place orders.

$$\texttt{authorized(P)} \leftarrow \texttt{paid(P)}, \texttt{prof(P)}.$$
$$\texttt{authorized(P)} \leftarrow \texttt{registered(P)}, \texttt{student(P)}.$$

Since the database can be incomplete about the registration and payment information, we need default knowledge to reason about unspecified variables. For instance, if it is unknown that a professor has paid, we believe the professor has not; if it is unknown that a student has registered, we believe the student has not.

$$\neg\texttt{paid(P)} \leftarrow \texttt{not paid(P)}, \texttt{prof(P)}.$$
$$\neg\texttt{registered(P)} \leftarrow \texttt{not registered(P)}, \texttt{student(P)}.$$

ASP is strong in default reasoning in that it allows prioritized defaults and exceptions at different levels (Gelfond and Kahl 2014). LR has the Closed World Assumption (CWA) for some predicates, e.g., the below rule guarantees that the value of attribute $\texttt{authorized(P)}$ must be either $\texttt{true}$ or $\texttt{false}$ (cannot be unknown):

$$\neg\texttt{authorized(P)} \leftarrow \texttt{not authorized(P)}.$$

The following two *pr-atoms* state the probability of delivering for person $\texttt{P}$ to $\texttt{P}$'s working place (0.8) and the probability of delivering $\texttt{coffee}$ in the morning (0.8).

$$\texttt{pr}(\texttt{req_room(P)} = \texttt{R} \mid \texttt{place(P,R)}) = 0.8.$$
$$\texttt{pr}(\texttt{req_item(P)} = \texttt{coffee} | \texttt{curr_time} = \texttt{morning}) = 0.8.$$

Random selection rules and pr-atoms, such as the ones above, allow us to represent and reason about commonsense with probabilities. Finally, a shopping request is specified as follows:

$$\texttt{task(I,R,P)} \leftarrow \texttt{req_item(P)} = \texttt{I}, \texttt{req_room(P)} = \texttt{R}, \texttt{req_person} = \texttt{P}, \texttt{authorized(P)}.$$

PR takes queries from PP and returns the joint probability. For instance, if it is known that Bob, a professor, has paid, and the current time is morning, a query for calculating the probability of $\langle\texttt{sandwich}, \texttt{office1}, \texttt{alice}\rangle$ is of the form:

$$?\{\texttt{task(sandwich, office1, alice)}\} \mid \texttt{do(paid(bob))}, \texttt{obs(curr_time} = \texttt{morning)}.$$

The fact that $\texttt{bob}$ paid increases the uncertainty in estimating the value of $\texttt{req_person}$ by bringing in additional possible worlds that include $\texttt{req_person} = \texttt{bob}$.

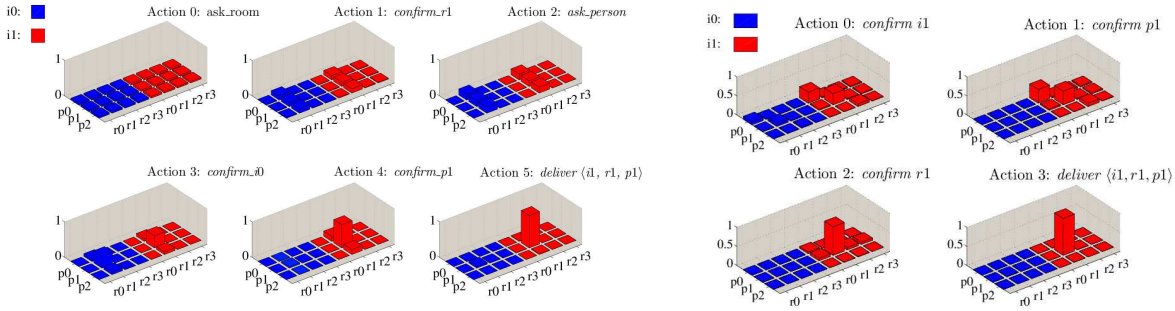
A POMDP needs to model all partially observable attributes relevant to the task at hand. In the shopping request identification problem, an underlying state is composed of an item, a room and a person. The robot can ask polar questions such as "*Is this delivery for Alice?*", and wh-questions such as "*Who is this delivery for?*". The robot expects observations of "yes" or "no" after polar questions and an element from the sets of items, rooms, or persons after wh-questions. Once the robot becomes confident in the request estimation, it can take a delivery action that deterministically leads to a terminating state. Each delivery action specifies a shopping task.

6.3 Experimental results

We have implemented the proposed approach on a BWIBot to identify shopping request tasks. The planner helps the robot decide whether to ask more questions (and what to ask) or to take a delivery action (and which delivery action), balancing the cost of asking questions and the penalty of wrong deliveries. The robot has to model the uncertainty in observations to account for the unreliable speech recognition techniques. The robot keeps asking questions and updates its belief about the shopping requests being identified. This question-asking process ends when the robot is certain about the shopping request and decides to take a delivery action using the planning module explained in Section 4.6.

We present the belief change in an illustrative trial in Figure 7, where i, r and p are an item, room and person. $i0$ is sandwich and $i1$ is coffee. The robot first reads its internal memory and collects a set of facts such as the current time is "morning", $p0$'s office is $r0$, and $p1$'s office is $r1$. Reasoning with commonsense produced a prior shown in the top-left of Figure 7b, where the most probable two requests were $\langle i1, r0, p0\rangle$ and $\langle i1, r1, p1\rangle$. The robot took the first action to confirm the item was coffee. After observing a "*yes*", the robot further confirmed $p1$ and $r1$. Finally, it became confident in the estimation and successfully identified the shopping request. Therefore, reasoning with domain knowledge produced an informative prior, based on which the robot could directly focus on the most likely attribute values and ask corresponding questions. In contrast, when starting from a uniform prior (Figure 7a), the robot would have needed at least six actions before the delivery action. A demo video is available at: http://www.cs.utexas.edu/~larg/bwi_web/research/.

Figure 8 shows the experimental results. Each set of experiments has three data points because we assigned different penalties to incorrect identifications in PP. Generally, a larger penalty requires the robot to ask more questions before taking a delivery action. POMDP-based PP without commonsense reasoning produced the worst results. Combining LR with PP improves the performance by reducing the number of possible worlds. Finally, the proposed algorithm, CORPP, produced the best performance in both efficiency and accuracy.



(a) Belief change using baseline (PP only)

(b) Belief change using CORPP

Figure 7. Belief change using both approaches in an illustrative trial. As illustrated, CORPP takes fewer questions to reach the same conclusion using informative priors.

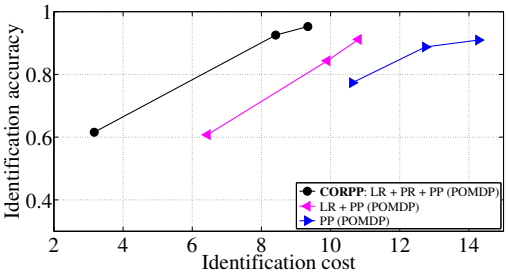


Figure 8. CORPP performs better than the other approaches in both efficiency and accuracy. Three data points on each curve correspond to different penalties of incorrect identifications. From left to right, the penalties are 10, 60 and 100 respectively.

In this section, we described an approach that integrates commonsense reasoning and probabilistic planning and allows the robot to handle dialog management with a human while using commonsense reasoning to specify a state space and instantiate a prior belief on the dialog.

7 Understanding Natural Language Requests

While the research contributions of the previous sections pertained mainly to fully autonomous planning, control, and reasoning, both for task planning and dialog systems, human responses during interaction are expected to be exact, and from a given range of possible responses. One of the most natural forms of human-robot interaction for humans is through natural language. However natural language processing remains a challenging research area within AI, and intelligent service robots should be able to efficiently and accurately understand commands from human users speaking in natural language.

In this section, we describe our research contributions pertaining to language learning to facilitate on-line improvement of the robots' understanding of spoken commands. We use a dialog agent embodied in a BWIBot to communicate with users through natural language and improve language understanding over time using data from these conversations (Thomason et al. 2015). By learning from conversations, our approach can recognize more complex language than keyword-based approaches without needing the large-scale, hand-annotated training data associated with complex language understanding tasks.

We train a semantic parser with a tiny set of expressions paired with robot goals. The natural language understanding component of our system is this semantic parser together with a conversational dialog agent. The dialog agent keeps track of the system's partial understanding of the goal the user is trying to convey and asks clarification questions to refine that understanding.

For example, given a high-level directive like "bring some java to Alice," our dialog agent uses follow-up questions to clarify any missing piece of needed information. If the agent does not recognize the phrase "some java," it may ask "What should I bring to Alice?" User clarifications provide training data pairs for a semantic parser. In this example, the user specifying "coffee" also lets the system know that "some java" and "coffee" mean the same thing. Less trivially, the agent may ask the user to rephrase his or her whole query, ultimately resulting in training pairs of commands to fully-formed action goals.

Using the conversation from the dialog agent to build training examples for the semantic parser, the natural language component as a whole is able to correctly interpret user commands faster over time.

7.1 Related Work

The work presented in this section is the first approach to intersect semantic parsing, dialog, and robot language grounding.

At the intersection of semantic parsing and language grounding, prior work uses restricted language and a static, hand-crafted lexicon to map natural language to action specifications (Matuszek et al. 2013). These specifications are grounded against a knowledge base onboard a robot, similar to how we can resolve semantic forms for expressions like "Alice's office" to physical rooms in the environment. We also use the knowledge base used for planning on the robot to ground semantic expressions.

At the intersection of dialog and language grounding, past work presented a dialog agent used together with a knowledge base and understanding component to learn new referring expressions during conversations that instruct a mobile robot (Kollar, Perera, Nardi and Veloso 2013). They use semantic frames of actions and arguments extracted from user utterances, while we use λ-calculus meaning representations. Our agent reasons about arguments like "Mallory Morgan's office", by considering what location would satisfy the expression, while semantic frames instead add a lexical entry for the whole phrase explicitly mapping to the appropriate room. Our method is more flexible for reasoning (e.g. "the person whose office is next to Mallory Morgan's office") and changes to arguments (e.g. "George Green's office").

Learning from conversations in our work is inspired by past work at the intersection of semantic parsing and dialog (Artzi and Zettlemoyer 2011). That work used logs of conversations users had with an air-travel information system to train a semantic parser for understanding user utterances. Our approach to learning is similar, but done incrementally from conversations the agent has with users, and our training procedure is integrated into a complete, interactive robot system.

7.2 Methodology

Figure 9 shows the interaction workflow between a human user and the embodied dialog agent. Users interacted with a BWIBot through the GUI by typing in natural language. In the example interaction, the underspecified command "go to the office" is parsed, grounded against the knowledge representation and reasoning node, which contains the knowledge base, and used to update the dialog agent's belief about the user's intent. The agent generates the response "Where should I walk?", having understood the action it should take but correctly recognizing that the destination was not specific enough. When the agent is confident in the user's intended command, a planning task with an appropriate goal is generated and passed to the software module responsible for task planning and execution, which

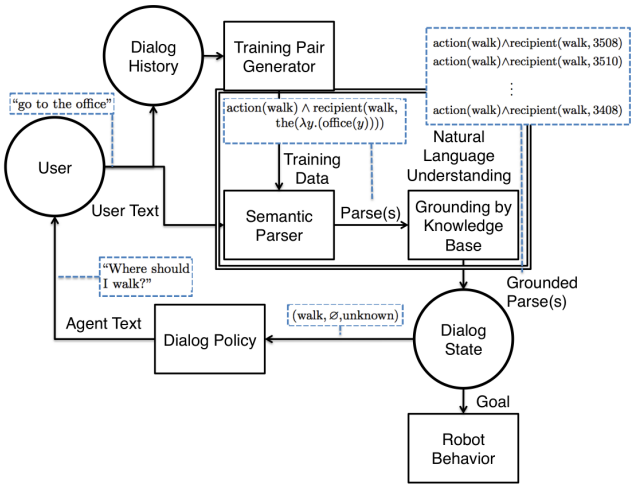


Figure 9. Dialog agent workflow. Dashed boxes show processing of user command "go to the office". When a command is understood, ASP generates a series of actions realized as robot behavior to carry out that command.

generates the necessary sequence of actions that the robot executes to accomplish that task. Consequently, this research contribution makes use of *high-level action control* for interacting with the user, and *planning level control* for grounding language in the knowledge base and executing requests.

For testing, users were asked to instruct the robot for one navigation task and one delivery task. These tasks were fixed for our 20 test users, who were divided into before- and after-training groups. Users could skip tasks if they felt they could not convey specified goals to the robot. Users filled out an experience survey after they were finished: "The tasks were easy to understand" (*Tasks Easy*); "The robot understood me" (*Understood*); "The robot frustrated me" (*Frustrated*); "I would use the robot to find a place unfamiliar to me in the building" (*Use Navigation*); and "I would use the robot to get items for myself or others" (*Use Delivery*). Users answered on a 5-point Likert scale: "Strongly Disagree"(0), "Somewhat Disagree"(1), "Neutral"(2), "Somewhat Agree"(3), "Strongly Agree"(4).

The initial group of 10 users (INIT TEST) interacted with the robot-embodied dialog agent with the semantic parser bootstrapped with a tiny set of expression/goal pairs.

We then allowed the system to perform incremental learning for four days in our office space. People working at the University of Texas at Austin Computer Science Department were encouraged to chat with the robot, but were not instructed on how to do so beyond a panel displaying information about people, offices, and items for delivery and a brief prompt saying the robot could only perform "navigation and delivery tasks". After understanding and carrying out a goal, the robot prompted

TASK TO COMPLETE

Dave Daniel wants the item in slot 5.

ROBOT	How can I help?
YOU	please bring the item in slot 5 to dave daniel
ROBOT	What should I bring to dave daniel?
YOU	a calander
ROBOT	I'm sorry, but I couldn't pinpoint what you meant by that.
ROBOT	What should I bring to dave daniel?
YOU	a day planner
ROBOT	I'm sorry, but I couldn't pinpoint what you meant by that.
ROBOT	What should I bring to dave daniel?
YOU	calendar
ROBOT	You want me to bring calendar to dave daniel?
YOU	yes
ROBOT	I thought so
ROBOT	Happy to help

Figure 10. This abridged conversation is from when the system had only been bootstrapped and not yet trained. Because of this conversation, the agent learned that "calander" and "day planner" mean "calendar" during retraining.

the user for whether the actions taken were correct. If they answered "yes" and the goal was not in the test set, the agent retrained its semantic parser with new training examples aligned from the conversation. Thirty-five such successful conversations were used to retrain the system before further evaluation.

To exemplify these training examples, Figure 10 shows a conversation the dialog agent had with a user in a prior, controlled experiment where users were told what goal to convey (similar to the methodology when testing performance). In addition to the prompt for the task to be completed, the user was shown a table of pictures with numbered slots; in slot 5 was a picture of a calendar. From this conversation, the agent pairs "please bring the item in slot 5 to dave daniel" with the correct semantic form understood after all clarifying questions, enabling it to learn that the construction "item in slot 5" can mean "calendar." Additionally, when trying to clarify the item to be brought, it learns the synonym "day planner" and the misspelling "calander" for "calendar." A video demonstrating the learning process on the BWIBot is available at: https://youtu.be/FL9IhJQOzb8.

We evaluated the retrained agent as before with the 10 remaining test users (TRAINED TEST) and the same set of testing goals.

7.3 Results

During training, the robot understood and carried out 35 goals, learning incrementally from these conversations. Table 2 compares the survey responses of users and the number of goals users completed of each task type in the INIT TEST and TRAINED TEST groups.

Table 2. Average survey responses from the two test groups and the proportion of task goals completed. Means in bold differ significantly ($p < 0.05$). Means in italics trend different ($p < 0.1$).

	INIT TEST	TRAINED TEST
Survey Question	Likert [0-4]	
Tasks Easy	3.8	3.7
Robot Understood	1.6	**2.9**
Robot Frustrated	2.5	*1.5*
Use Navigation	2.8	2.5
Use Delivery	1.6	2.5
Goals Completed	Percent	
Navigation	90	90
Delivery	20	*60*

We note that there is significant improvement in user perception of the robot's understanding and trends towards less user frustration and higher delivery-goal correctness. Though users did not significantly favor using the robot for tasks after training, several users in both groups commented that they would not use guidance only because the BWIBot moved too slowly.

In this section, we have implemented an agent that expands its natural language understanding incrementally from conversations with users by combining semantic parsing and dialog management. We have demonstrated that this learning on the BWIBot platform yields significant improvements in user experience and dialog efficiency when learning was restricted to natural, uncontrolled, in-person conversations the agent had over a few days' time.

8 Grounded Language Learning through Human-Robot Interaction

In the previous section, the research contribution focused on how commands can be provided via natural language, and the responses were grounded using the knowledge base on the robot. However, often it is necessary for a robot to ground language using its own perception and actions with respect to objects. Consider the case where a human asks a service robot, "Please bring me the full red bottle". To fulfill such a request, a robot would need to detect objects in its environment and determine whether the words "full", "red", and "bottle" match a particular object detection. Furthermore, such a task cannot be solved using static visual object recognition methods as detecting whether an object is full or empty may often require the robot to perform a certain action on it (e.g., lift the object to measure the force it exerts on the arm).

In this section, the research contribution focuses on solving the *symbol grounding problem* (Harnad 1990), a longstanding challenge in AI, where language is grounded

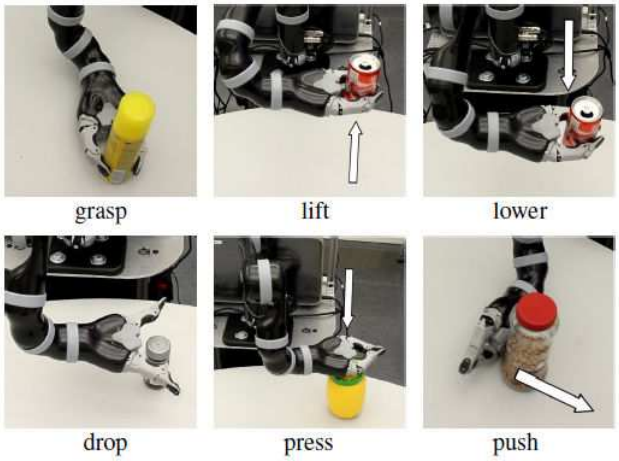


Figure 11. The exploratory behaviors used by the robot. The *look* action is not depicted.

using the robot's perception and action (Tellex et al. 2011; Matuszek et al. 2012; Krishnamurthy and Kollar 2013; Perera and Allen 2013; Kollar, Krishnamurthy and Strimel 2013; Tellex et al. 2014; Matuszek et al. 2014; Parde et al. 2015; Spranger and Steels 2015). To address this problem, we enable a robot to undergo two distinct developmental stages:

1. *Object Exploration Stage* – the robot interacts with objects using a set of exploratory behaviors designed to produce different kinds of multi-modal feedback.
2. *Social Learning Stage* – the robot interacts with humans in order to learn mappings from its sensorimotor experience with objects to words that can be used to described the objects.

8.1 Object Exploration Stage

To fulfill the first stage, the BWIBot featuring the Kinova Mico arm was equipped with several different exploratory behaviors, such as grasping on object, lifting it, pushing it, etc. These actions were modeled after the types of behaviors infants and toddlers use to learn about objects in the early months and years of life (Power 1999).

In a preliminary experiment, the robot explored 32 common household and office objects including various containers, cups, toys, etc. The robot's behavior repertoire consists of 7 different exploratory actions: *grasp*, *lift*, *hold*, *lower*, *drop*, *push*, and *press*. During the execution of each action the robot recorded visual, auditory and haptic sensory feedback. In addition, the robot is also equipped with the static *look* behavior which captures the object's visual appearance before the robot begins to interact with it. Figure 11 shows the exploratory actions used by the robot.

During the execution of the *look* behavior, the robot's visual system segments the 3D point cloud of the

Figure 12. Left: the robot guesses an object described by a human participant as silver, round, and empty. **Right**: a human participant guesses an object described by the robot as light, tall, and tub.

object from the tabletop and computes *color* histogram features in RGB space, *shape* histogram features as implemented by Rusu et al. (2009), and deep visual features computed by the 16-layer VGG network proposed by Simonyan and Zisserman (2014). During the execution of each of the remaining 7 exploratory behaviors, the robot computes *auditory* and *haptic* features as described by Sinapov et al. (2014). In addition, when performing the *grasp* behavior, the robot used the same methodology to extract *proprioceptive* features capturing how the fingers' joint positions change over time.

A more detailed description of the objects and data collection methods used for this dataset can be found in a paper on object ordering using haptic and proprioceptive behavior (Sinapov et al. 2016).

8.2 Social Learning Stage

To learn words describing individual objects, our robot uses a variation on the children's game "I Spy". During each game session, the human and the robot take turns describing objects from among 4 on a tabletop, as shown in Figure 12. On the human's turn, the robot asks him or her to pick an object and describe it in one phrase. The robot subsequently attempts to guess which object matches the words heard from the human. To do so, over the course of multiple sessions the robot learns a behavior-grounded classifier for each word that it observes using the methodology of Sinapov et al. (2014). Given the words uttered by the human, the robot then picks the object that has the highest scores from the classifiers corresponding to the words. To indicate its pick, the robot moves the arm, points to the object, and asks the human if the choice is correct.

During the robot's turn, an object is chosen at random from those on the table and described by the robot using 3 words corresponding to the 3 classifiers with the highest score for that object. The robot then asks the human to make a guess by physically touching or lifting the object. After a correct guess, the robot asks questions about the object

in the form of "would you use the word X to describe the object?" where X is one of the words that the robot has observed.

8.3 Experiment

To test our system, we conducted an experiment involving 42 human participants, consisting of undergraduate and graduate students, staff, and faculty. To measure the robot's learning progress over time, we divided an object set into four folds. For each fold, at least 10 participants each played 4 rounds of "I Spy" with the robot. After each fold, the robot's classifiers were re-training using the newly gathered data, and new classifiers were created for words that were novel to that fold.

We measured the number of guesses it took the robot and the human to correctly identify the object during their respective turns. The experiment was conducted under two conditions: *vision-only* during which the robot attempts to ground words using only visual sensory feedback detected during the *look* behaviors, and *multi-modal*, during which the robot used all available sensory feedback from all behaviors.

8.4 Results

By the end of the experiment, the robot had learned behavior-grounded classifiers for around 70 words that the participants used to describe objects (Thomason et al. 2016). Most noticeably, in the *multi-modal* condition, there was a statistically significant decrease in the number of guesses it took the robot to identify the object as a result of the robot's interactive game-play experience. During the first fold, it took the robot an average of 2.5 guesses to solve each task. During the second fold, the robot was able to identify the object with an average of 1.98 guesses, which dropped to 1.73 during the third fold.

Figure 13 details these results. Because we had access to the scores the robot assigned each object, we calculated the *expected* number of robot guesses for each turn. For example, if all 4 objects were tied for first, the expected number of robot guesses for that turn was 2.5, regardless of whether it got (un)lucky and picked the correct object (last)first. (The expected number for 4 tied objects is 2.5 because the probability of picking in any order is equal, so the expected turn to get the correct object is $\frac{1+2+3+4}{4} = \frac{10}{4} = 2.5$)

A close look at the classifiers learned by the robot showed that for many words, such as "full", "empty", and "heavy", visual features alone were insufficient for accurate grounding. Using the framework for grounding semantic categories proposed by Sinapov et al. (2014), the robot was able to estimate the reliability of particular combinations of a sensory modality and a behavior for the task of recognizing whether a particular word fits an object. These

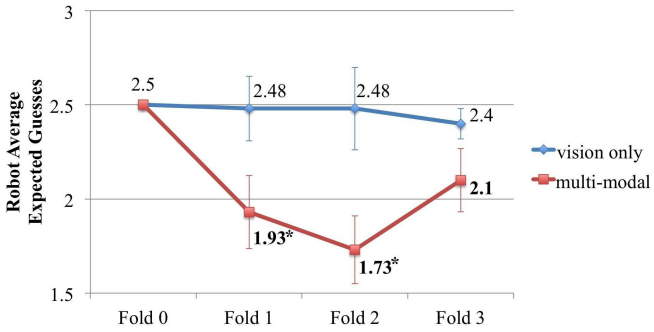


Figure 13. Average expected number of guesses the robot made on each human turn with standard error bars shown. **Bold**: significantly lower than the average at fold 0 with $p < 0.05$ (unpaired Student's t-test). *: significantly lower than the competing system on this fold on participant-by-participant basis with $p < 0.05$ (paired Student's t-test).

Metric	System	
	vision only	**multi-modal**
precision	.250	.378+
recall	.179	.348*
F_1	.196	.354*

Table 3. Average performance of predicate classifiers used by the **vision only** and **multi-modal** systems in leave-one-object-out cross validation. *: significantly greater than competing system with $p < 0.05$. +: $p < 0.1$ (Student's un-paired t-test).

estimates show that for words describing the internal state of objects, the robot largely relied on the haptic sensory feedback produced when manipulating the object. Words describing the shape (e.g., "cylindrical") and color of the object were in turn best recognized using visual features. Auditory features were most useful for words denoting the object's material (e.g., "metal" vs "plastic") as well as compliance (e.g., objects that are "soft" produce less sound when dropped and pushed).

To demonstrate the effectiveness of multi-modal grounding quantitatively, we obtained agreement scores between the multi-modal versus vision only classifiers with human labels on objects. Training the predicate classifiers using leave-one-out cross validation over objects, we calculated the average precision, recall, and F_1 scores of each against human predicate labels on the held-out object. Table 3 gives these metrics for the 74 predicates used by the systems.[‡‡]

Across the objects our robot explored, our multi-modal system achieves consistently better agreement with human

[‡‡]There were 53 predicates shared between the two systems. The results in Table 3 are similar for a paired t-test across these shared predicates with slightly reduced significance.

assignments of predicates to objects than does the vision only system.

Ongoing and future work will focus on expanding our service robots' ability to learn about objects from humans. While our focus thus far was on a game-play scenario in which participants were brought to the lab, we envision that in the near future our robot will be able to autonomously find people and engage in dialogue with the propose of learning. Towards that goal, we are currently implementing a system for autonomous object exploration and fetching which will enable a robot to find an interesting object, explore it, and finally engage a person in dialogue about the object for the purpose of grounded language acquisition.

9 Robot-centric Human Activity Recognition

In the research contributions described in the previous sections, the robot aims to understand human intention via direct means such as spoken or written commands specified in natural language. For a robot to effectively function in a human-inhabited environment, it would also be useful for it to be aware of the activities and intentions of humans around it based on its own observations. For example, consider the case where a BWIBot is navigating a crowded environment such as an undergraduate computer lab. If the robot could recognize when a person needs help, or when a person is trying to approach or engage it (or avoid it), its social and navigational skills would improve dramatically. In this section, we describe a research contribution which explores how human activity can be recognized, making it possible for a BWIBot to understand the intent of humans in its vicinity.

To address visual activity recognition, the computer vision research community has produced a wide array of methods for recognizing human activities (see Aggarwal and Ryoo (2011) for a review). Most relevant to our work are studies in which the video is captured by a robot. Such studies are relatively new and include the works of Ryoo et al. (2015); Xia et al. (2015); Ryoo and Matthies (2013); Chrungoo et al. (2014). This existing work is subject to several limitations: 1) The activities were not carried out spontaneously but rather, were rehearsed or commanded by the experimenters; 2) The activities were performed by a small number of people, typically 5-8; 3) The robot was typically either stationary or teleoperated.

Our work on activity recognition overcomes these limitations in several important ways. First, our robot uses its autonomous navigation capability in a large, unstructured, and human-inhabited environment, as opposed to a laboratory. Second, the activities learned by our robot were performed spontaneously by many different people who interacted with (or were observed by) the robot, as opposed to the standard methodology of asking study participants to perform certain actions. And third, in contrast to classic computer vision approaches, our system uses both visual and non-visual cues when recognizing the activities of humans that it interacts with.

Next, we describe the robot's activity recognition system and present experimental results conducted from a week long experiment in which the BWIBot autonomously patrolled through an undergraduate and a graduate student lab via randomly generated planning tasks. Video captured during this experiment was then processed offline to categorize different human activities.

9.1 Overview of Activity Recognition System

We formulate the problem of activity recognition as a multi-class classification problem, i.e., the robot has to recognize an observed activity as one of k activity classes. As input, the robot is given some visual and non-visual sensory feature descriptors computed from the set of frames during which the robot's sensor detected and tracked a person.

To perform human detection and tracking, the robot uses the KinectV2, as explained in Section 3.4. The Kinect SDK is capable of simultaneously detecting and tracking up to 6 people at a time, as well as estimating the positions of 21 joint markers corresponding to joints such as the neck, shoulders, waist, elbows, knees, etc. Whenever a new person is detected by the robot, the robot's system recorded a sequence of RGB images, $\mathcal{I} \in \mathbb{R}^{512\times424\times3\times t}$, a sequence of depth images $\mathcal{D} \in \mathbb{R}^{512\times424\times t}$, and a sequences of joint markers, $\mathcal{J} \in \mathbb{R}^{21\times3\times t}$, where t is the number of frames during which the system detected and tracked the person.

The raw image and joint-marker data are too highly dimensional to be used as direct input to standard classification algorithms. To reduce dimensionality, we implemented five different visual feature extraction algorithms:

- Covariance of the joint positions over time (COV) as described by Hussein et al. (2013).
- Histogram of the joints in 3D (HOJ3D) as described by Xia et al. (2011).
- Pairwise Joint Relation Matrix features (PRM) as described by Gori et al. (2015).
- Histogram of Direction Vectors (HODV) as described by Chrungoo et al. (2014).
- Histogram of Oriented 4D Normals (HON4D) as described by Oreifej and Liu (2013).

Each of these methods computes a real-valued feature vector for each frame in a given sequence of joint-marker data or depth image data. To further reduce dimensionality, the feature vectors that were extracted for each frame were quantized using k-means and represented using Bag Of Words (BoW). Thus, each sequences of frames

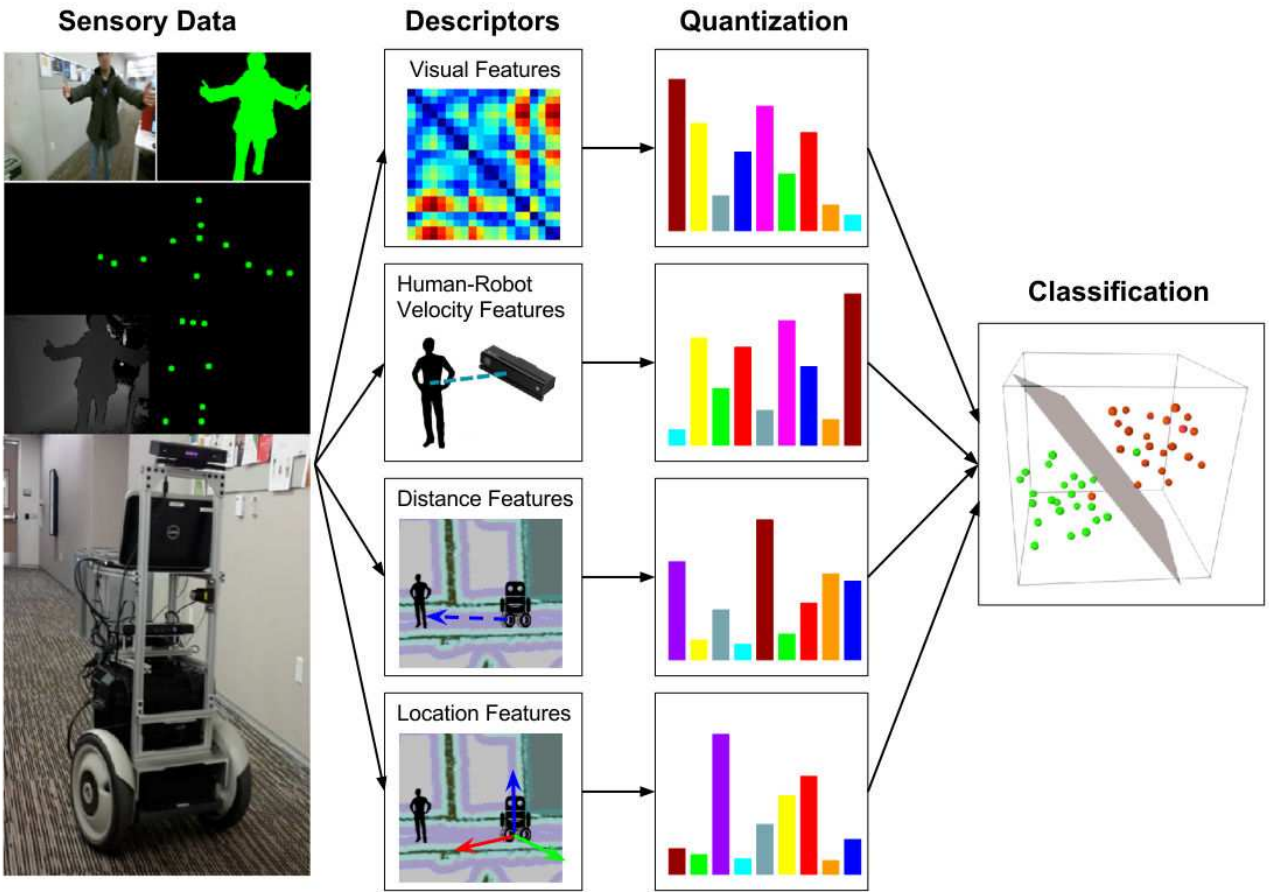


Figure 14. An overview of the robot's activity recognition system. As the robot navigates the environment, it uses the Kinect sensor to detect humans in its environment. Subsequently, the robot computes visual and non-visual features for each detection, quantizes the features, and uses them as an input to a Support Vector Machine for activity recognition.

was represented as a single feature vector encoding the distribution of visual "words".

In addition to visual features, our system also uses non-visual data as input to the activity recognition classifier. We hypothesized that the types of activities that humans may perform in front of the robot may be influenced by the distance between the robot and the person. In addition, it is likely that different activities may be more likely to occur at different locations in the robot's environment (e.g., the activity of sitting down on a desk is more likely to be observed in the open lab area where there are many desks as opposed to a hallway). Therefore, as described in Gori et al. (2015), we added three additional non-visual features:

- Human-Robot velocity features representing the movement of the person with respect to the robot
- Human-Robot distance features representing the distance between the human and the robot
- Robot location features representing the robot's pose (i.e., position and orientation) in the map over the course of the observation

The non-visual features were also computed for each frame of each observation, quantized with k-means, and represented using BoW. Note that these non-visual features are specific to our robot and our environment and thus, the learned activity recognition model may not always be applicable on a different robot in a different building. Figure 14 shows an overview of the activity recognition system.

9.2 Experimental Evaluation and Results

The robot's activity recognition system was evaluated by collecting a dataset over the course of the robot's autonomous navigation of the environment, which consisted of a graduate and an undergraduate student lab, connected by two door ways. The robot traversed the

environment for 1-2 hours per day, for 6 days, traveling a total of 14.03 km. After the observations were recorded, each detection of a person was manually labeled with one of several activity labels: *approach, block, pass by, take picture, side pass, sit, stand, walk away, wave, false*. The label *false* corresponded to false detections by the Kinect SDK, which typically corresponded to fixed objects in the environment. In total, there were 1204 detections, each labeled with one of the 10 activity classes.

The classifier implemented by our activity recognition system was a non-linear Support Vector Machine using the $\mathcal{X}^2$ kernel function. Other kernel functions (e.g., Gaussian and Polynomial) and other classifiers (e.g., Naive Bayes, C4.5 decision tree) achieved comparable results. The classifier's performance was evaluated using stratified 6-fold cross-validation, which was performed 10 different times with random fold splits. The dataset is very imbalanced with respect to the activity labels (i.e., some activities are much more common than others) and therefore, the performance was measured in terms of Cohen's *kappa* coefficient (Cohen 1960) which compares the classifier's accuracy against chance accuracy:

$$K = \frac{Pr(a) - Pr(e)}{1 - Pr(e)},$$

where $Pr(a)$ is the probability of correct classification by the classifier, and $Pr(e)$ is the probability of correct classification by chance. A *kappa* of 1.0 corresponds to a perfect classifier, while 0.0 corresponds to a classifier that randomly assigns a class labels based on the prior label distribution.

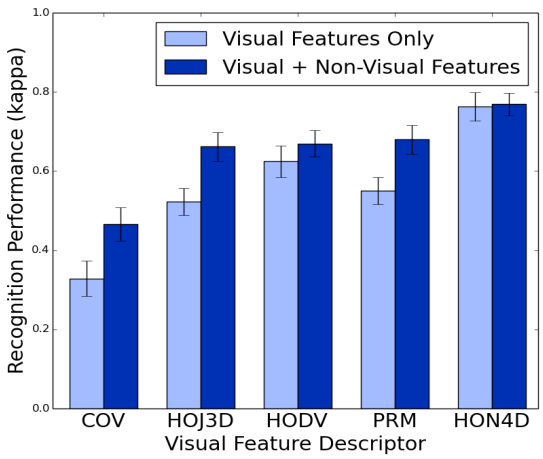


Figure 15. Activity recognition results using 5 different visual feature descriptors (described in Section 9.1) under 2 different conditions: visual features only, and visual + non-visual features. The error bars represent standard error.

Figure 15 shows the results of the cross-validation test with 5 different visual feature descriptors and two different conditions: visual features only, and visual features concatenated with non-visual features. The HON4D visual feature descriptor performs the best out of all 5 – unlike the rest which are computed from joint-marker data, the HON4D descriptor is computed from the saved depth image sequences which may explain why it performs substantially better (a drawback to the HON4D descriptor is that it is much more computationally expensive to compute than the rest). Adding the three non-visual features to the representation improves the SVM's performance and, depending on the visual descriptor, the improvement can be quite substantial and significant.

In ongoing and future work, we are exploring how the robot's activity recognition system can be used for activity-aware autonomous navigation. For example, if the robot recognizes that a person is taking a picture of it, it would be intuitive for it to pause its current task and motion for a moment. In addition, while the existing system focuses only on activities performed by individual persons, we plan to extend it by adding the ability to learn about interactions between multiple people performing activities in relation to each other and/or the robot. We believe that enabling a robot to learn and reason about the activities of people around it has the potential to greatly improve its ability to navigate around and interact with people, particularly in large and crowded environments.

10 Conclusion

In this paper, we have presented an overview of the BWIBots, both from a hardware and software perspective. We have also outlined how these robots have enabled research on a variety of projects pertaining to robot reasoning, action planning, and human-robot interaction. Specifically, the first research contribution presented in this paper has demonstrated how action language $\mathcal{BC}$ can be used construct a planning and action execution system that is able to express defeasible reasoning and recursively defined fluents. The second contribution has integrated probabilistic and symbolic reasoning for constructing a spoken dialog system that uses commonsense reasoning to resolve queries efficiently. The third and fourth contributions have looked into how requests in natural language can be interpreted by a robot, how these requests can be grounded in a robot's perception and actions. Finally, the last contribution investigates how human activity can be categorized from afar.

While all the research contributions presented in this paper are used for single-robot applications, one of the main goals behind the development of the BWIBots is to enable multi-robot research and applications. When multiple robots share a physical environment, their plans

might interact such that their independently-computed optimal plans become suboptimal at runtime. Toward achieving the global optimality in a multirobot system, the robots need to compute plans to simultaneously share limited domain resources and realize synergy within the robot team. However, robots' noisy action durations pose a challenge to achieve such robot behaviors. In our ongoing research, we are investigating algorithms for multi-robot planning while considering the uncertainty in noisy action durations (Zhang et al. 2016).

Another multi-robot application that we intend to work on is a real-world implementation of a multi-robot human guidance system (Khandelwal et al. 2015). In this previous work, we have explored how multiple robots in simulation can be coordinated to efficiently guide a human to his destination, while simultaneously minimizing the time each robot is diverted from other duties to do so. A real-world implementation of this work helps verify many modeling assumptions made in the simulation, and helps explore how robots can effectively provide instructions with less ambiguity to people.

In addition to multi-robot research, we expect that the current and future BWIBots will continue to support research on HRI and other areas of AI and robotics. Our long-term goal is for the BWIBots to be an always-on, permanent fixture in the UT Austin Computer Science building, such that inhabitants of and visitors to the building expect to interact with them and find them useful and entertaining. We hope that this article will help inspire and inform other such systems throughout the world.

Acknowledgements

This work has taken place in the Learning Agents Research Group (LARG) at the Artificial Intelligence Laboratory, The University of Texas at Austin. LARG research has been supported in part by grants from the National Science Foundation (CNS-1330072, CNS-1305287), ONR (21C184-01), AFOSR (FA8750-14-1-0070, FA9550-14-1-0087), Yujin Robot, and the Freshmen Research Initiative (FRI) at UT Austin.

The authors would like to thank Chien-Liang Fok, Sriram Vishwanath, and Christine Julien for providing the Segway RMP bases used in the first two iterations of the BWIBots. Liang's help and design ideas were instrumental in constructing the first BWIBots, without which future evolution of the platform would not be possible.

The authors would also like to thank Jack O'Quin for maintaining many of the software packages used by the BWIBots, as well as streamlining the operation of the BWI Lab. His work has enabled many of the authors to focus on the core research contributions presented in this paper.

The authors would also like to thank many FRI students, and in particular, Yuqian Jiang, Rolando Fernandez, and Patricio Lankenau for their assistance in developing and maintaining the hardware and software behind the BWIBots.

References

Aggarwal, J. K. and Ryoo, M. S. (2011), 'Human activity analysis: A review', *ACM Computing Surveys (CSUR)* **43**(3), 16.

Artzi, Y. and Zettlemoyer, L. (2011), Bootstrapping semantic parsers from conversations, *in* 'Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)', Association for Computational Linguistics, Stroudsburg, PA, USA, pp. 421–432.
URL: *http://dl.acm.org/citation.cfm?id=2145432.2145481*

Babb, J. and Lee, J. (2013), Cplus2ASP: Computing action language $\mathcal{C}+$ in answer set programming, *in* 'International Conference on Logic Programming and Nonmonotonic Reasoning (LPNMR)'.

Baral, C., Gelfond, M. and Rushton, N. (2009), 'Probabilistic reasoning with answer sets', *Theory and Practice of Logic Programming* **9**(01), 57–144.

Bastianelli, E., Bloisi, D., Capobianco, R., Cossu, F., Gemignani, G., Iocchi, L. and Nardi, D. (2013), On-line semantic mapping, *in* '16th International Conference on Advanced Robotics (ICAR)'.

Caldiran, O., Haspalamutgil, K., Ok, A., Palaz, C., Erdem, E. and Patoglu, V. (2009), Bridging the gap between high-level reasoning and low-level control, *in* 'International Conference on Logic Programming and Nonmonotonic Reasoning (LPNMR)'.

Chen, K., Lu, D., Chen, Y., Tang, K., Wang, N. and Chen, X. (2014), The intelligent techniques in robot kejia–the champion of robocup@ home 2014, *in* 'RoboCup 2014: Robot World Cup XVIII', Springer, pp. 130–141.

Chen, X., Ji, J., Jiang, J., Jin, G., Wang, F. and Xie, J. (2010), Developing high-level cognitive functions for service robots, *in* 'International Conference on Autonomous Agents and Multiagent Systems (AAMAS)'.

Chen, X., Jin, G. and Yang, F. (2012), Extending $\mathcal{C}+$ with composite actions for robotic task planning, *in* 'International Conference on Logical Programming (ICLP)'.

Chen, X., Xie, J., Ji, J. and Sui, Z. (2012), 'Toward open knowledge enabling for human-robot interaction', *Journal of Human-Robot Interaction* **1**(2), 100–117.

Chrungoo, A., Manimaran, S. and Ravindran, B. (2014), Activity recognition for natural human robot interaction, *in* 'Social Robotics', Springer, pp. 84–94.

Cohen, J. (1960), 'A coefficient of agreement for nominal scales', *Educational and Psychological Measurement* **20**(1), 37–46.

Coltin, B., Veloso, M. M. and Ventura, R. (2011), Dynamic user task scheduling for mobile robots., *in* 'Automated Action Planning for Autonomous Mobile Robots'.

Cousins, S. (2010), 'Ros on the pr2 [ros topics]', *Robotics & Automation Magazine, IEEE* **17**(3), 23–25.

Eiter, T., Faber, W., Leone, N., Pfeifer, G. and Polleres, A. (2003), 'Answer set planning under action costs', *Journal of Artificial Intelligence Research (JAIR)* .

Erdem, E., Aker, E. and Patoglu, V. (2012), 'Answer set programming for collaborative housekeeping robotics: representation, reasoning, and execution', *Intelligent Service Robotics (ISR)* .

Erdem, E. and Patoglu, V. (2012), Applications of action languages in cognitive robotics, *in* 'Correct Reasoning'.

Erdem, E., Patoglu, V., Saribatur, Z. G., Schüller, P. and Uras, T. (2013), 'Finding optimal plans for multiple teams of robots through a mediator: A logic-based approach', *Theory and Practice of Logic Programming (TPLP)* .

Finger, J. (1986), Exploiting Constraints in Design Synthesis, PhD thesis, Stanford University.

Fox, D., Burgard, W., Dellaert, F. and Thrun, S. (1999), 'Monte carlo localization: Efficient position estimation for mobile robots', *AAAI/IAAI* **1999**, 343–349.

Gebser, M., Grote, T. and Schaub, T. (2010), Coala: a compiler from action languages to ASP, *in* 'European Conference on Logics in Artificial Intelligence (JELIA)'.

Gebser, M., Kaufmann, B., Kaminski, R., Ostrowski, M., Schaub, T. and Schneider, M. (2011), 'Potassco: The potsdam answer set solving collection', *Ai Communications* **24**(2), 107–124.

Gelfond, M. and Kahl, Y. (2014), *Knowledge representation, reasoning, and the design of intelligent agents: The answer-set programming approach*, Cambridge University Press.

Gelfond, M. and Lifschitz, V. (1988), The stable model semantics for logic programming, *in* 'International Logic Programming Conference and Symposium (ICLP/SLP)'.

Gelfond, M. and Lifschitz, V. (1991), 'Classical negation in logic programs and disjunctive databases', *New Generation Computing* .

Giunchiglia, E., Lee, J., Lifschitz, V., McCain, N. and Turner, H. (2004), 'Nonmonotonic causal theories', *Artificial Intelligence* **153**(1), 49–104.

Gori, I., Sinapov, J., Khante, P., Stone, P. and Aggarwal, J. (2015), Robot-centric activity recognition in the wild, *in* 'Social Robotics', Springer International Publishing, pp. 224–234.

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

Hanheide, M., Göbelbecker, M., Horn, G. S., Pronobis, A., Sjöö, K., Aydemir, A., Jensfelt, P., Gretton, C., Dearden, R., Janicek, M. et al. (2015), 'Robot task planning and explanation in open and uncertain worlds', *Artificial Intelligence* .

Harnad, S. (1990), 'The symbol grounding problem', *Physica D: Nonlinear Phenomena* **42**(1), 335–346.

Havur, G., Haspalamutgil, K., Palaz, C., Erdem, E. and Patoglu, V. (2013), A case study on the Tower of Hanoi challenge: Representation, reasoning and execution, *in* 'International Conference on Robotics and Automation (ICRA)'.

Helmert, M. (2006), 'The fast downward planning system.', *J. Artif. Intell. Res.(JAIR)* **26**, 191–246.

Hoffmann, J. and Nebel, B. (2001), 'The ff planning system: Fast plan generation through heuristic search', *Journal of Artificial Intelligence Research* pp. 253–302.

Hussein, M. E., Torki, M., Gowayyed, M. A. and El-Saban, M. (2013), 'Human action recognition using a temporal hierarchy of covariance descriptors on 3d joint locations', *International Joint Conference on Artificial Intelligence (IJCAI)* .

Kass, M., Witkin, A. and Terzopoulos, D. (1988), 'Snakes: Active contour models', *International journal of computer vision* **1**(4), 321–331.

Khandelwal, P., Barrett, S. and Stone, P. (2015), Leading the way: An efficient multi-robot guidance system, *in* 'Proceedings of the 2015 International Conference on Autonomous Agents and Multiagent Systems', International Foundation for Autonomous Agents and Multiagent Systems, pp. 1625–1633.

Khandelwal, P. and Stone, P. (2014), Multi-robot human guidance using topological graphs, *in* 'AAAI Spring 2014 Symposium on Qualitative Representations for Robots'.

Khandelwal, P., Yang, F., Leonetti, M., Lifschitz, V. and Stone, P. (2014), Planning in action language $\mathcal{BC}$ while learning action costs for mobile robots, *in* 'International Conference on Automated Planning and Scheduling (ICAPS)'.

Koenig, N. and Howard, A. (2004), Design and use paradigms for gazebo, an open-source multi-robot simulator, *in* 'Intelligent Robots and Systems, 2004.(IROS 2004). Proceedings. 2004 IEEE/RSJ International Conference on', Vol. 3, IEEE, pp. 2149–2154.

Kollar, T., Krishnamurthy, J. and Strimel, G. (2013), Toward interactive grounded language acquisition, *in* 'Robotics: Science and Systems'.

Kollar, T., Perera, V., Nardi, D. and Veloso, M. (2013), Learning environmental knowledge from task-based human-robot dialog, *in* 'Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)', Karlsruhe, pp. 4304–4309.

Krishnamurthy, J. and Kollar, T. (2013), 'Jointly learning to parse and perceive: Connecting natural language to the physical world', *Transactions of the Association for Computational Linguistics* **1**, 193–206.

Kuindersma, S. R., Hannigan, E., Ruiken, D., Grupen, R. et al. (2009), Dexterous mobility with the ubot-5 mobile manipulator, *in* 'Advanced Robotics, 2009. ICAR 2009. International Conference on', IEEE, pp. 1–7.

Lee, J., Lifschitz, V. and Yang, F. (2013), Action Language $\mathcal{BC}$: A Preliminary Report, *in* 'Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)'.

Leonetti, M., Iocchi, L. and Stone, P. (2016), 'A synthesis of automated planning and reinforcement learning for efficient, robust decision-making', *Artificial Intelligence* **241**, 103 – 130.

Lifschitz, V. (2008), What is answer set programming?, *in* 'Proceedings of the 23rd National Conference on Artificial Intelligence - Volume 3', AAAI'08, AAAI Press, pp. 1594–1597.
URL: *http://dl.acm.org/citation.cfm?id=1620270.1620340*

Lowe, D. G. (2004), 'Distinctive image features from scale-invariant keypoints', *International journal of computer vision* **60**(2), 91–110.

Luber, M. and Arras, K. O. (2013), Multi-hypothesis social grouping and tracking for mobile robots., *in* 'Robotics: Science and Systems'.

Marder-Eppstein, E., Berger, E., Foote, T., Gerkey, B. and Konolige, K. (2010), The office marathon: Robust navigation in an indoor office environment, *in* 'Robotics and Automation (ICRA), 2010 IEEE International Conference on', IEEE, pp. 300–307.

Marek, V. and Truszczynski, M. (1999), Stable models and an alternative logic programming paradigm, *in* 'The Logic Programming Paradigm: a 25-Year Perspective', Springer Verlag.

Matuszek, C., Bo, L., Zettlemoyer, L. and Fox, D. (2014), Learning from unscripted deictic gesture and language for human-robot interactions, *in* 'Proceedings of the 28th AAAI Conference on Artificial Intelligence', Quebéc City, Quebéc, Canada.

Matuszek, C., FitzGerald, N., Zettlemoyer, L., Bo, L. and Fox, D. (2012), A joint model of language and perception, *in* 'Proceedings of the 29th International Conference on Machine Learning', Edinburgh, Scotland, UK.

Matuszek, C., Herbst, E., Zettlemoyer, L. and Fox, D. (2013), Learning to parse natural language commands to a robot control system, *in* 'Experimental Robotics', Springer International Publishing, pp. 403–415.

McCarthy, J. and Hayes, P. (1969), Some philosophical problems from the standpoint of artificial intelligence, *in* 'Machine Intelligence', Edinburgh University Press.

Mudrova, L. and Hawes, N. (2015), Task scheduling for mobile robots using interval algebra, *in* 'Robotics and Automation (ICRA), 2015 IEEE International Conference on', IEEE, pp. 383–388.

Munaro, M. and Menegatti, E. (2014), 'Fast rgb-d people tracking for service robots', *Autonomous Robots* **37**(3), 227–242.

Niemelä, I. (1999), 'Logic programs with stable model semantics as a constraint programming paradigm', *Annals of Mathematics and Artificial Intelligence* .

Oreifej, O. and Liu, Z. (2013), 'Hon4d: Histogram of oriented 4d normals for activity recognition from depth sequences', *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* .

Parde, N., Hair, A., Papakostas, M., Tsiakas, K., Dagioglou, M., Karkaletsis, V. and Nielsen, R. D. (2015), Grounding the meaning of words through vision and interactive gameplay, *in* 'Proceedings of the 24th International Joint Conference on Artificial Intelligence', Buenos Aires, Argentina, pp. 1895–1901.

Perera, I. and Allen, J. F. (2013), Sall-e: Situated agent for language learning, *in* 'Proceedings of the 27th AAAI Conference on Artificial Intelligence', Bellevue, Washington, USA, pp. 1241–1247.

Power, T. G. (1999), *Play and exploration in children and animals*, Psychology Press.

Quigley, M., Conley, K., Gerkey, B., Faust, J., Foote, T., Leibs, J., Wheeler, R. and Ng, A. Y. (2009), Ros: an open-source robot operating system, *in* 'ICRA workshop on open source software', Vol. 3, p. 5.

Quinlan, S. and Khatib, O. (1993), Elastic bands: Connecting path planning and control, *in* 'Robotics and Automation, 1993. Proceedings., 1993 IEEE International Conference on', IEEE, pp. 802–807.

Quintero, E., Garcia-Olaya, Á., Borrajo, D. and Fernández, F. (2011), 'Control of autonomous mobile robots with automated planning', *Journal of Physical Agents (JoPhA)* .
URL: *http://www.plg.inf.uc3m.es/ dborrajo/papers/jopha11.pdf*

Reiser, U., Connette, C., Fischer, J., Kubacki, J., Bubeck, A., Weisshardt, F., Jacobs, T., Parlitz, C., Hägele, M. and Verl, A. (2009), Care-o-bot® 3: creating a product vision for service robot applications by integrating design and technology, *in* 'Proceedings of the 2009 IEEE/RSJ international conference on Intelligent robots and systems', IEEE Press, pp. 1992–1998.

Rosenthal, S., Biswas, J. and Veloso, M. (2010), An effective personal mobile robot agent through symbiotic human-robot interaction, *in* 'Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems: volume 1-Volume 1', International Foundation for Autonomous Agents and Multiagent Systems, pp. 915–922.

Rusu, R. B., Blodow, N. and Beetz, M. (2009), Fast point feature histograms (fpfh) for 3d registration, *in* 'Robotics and Automation, 2009. ICRA'09. IEEE International Conference on', IEEE, pp. 3212–3217.

Rusu, R. B. and Cousins, S. (2011), 3d is here: Point cloud library (pcl), *in* 'Robotics and Automation (ICRA), 2011 IEEE International Conference on', IEEE, pp. 1–4.

Ryoo, M., Fuchs, T. J., Xia, L., Aggarwal, J. and Matthies, L. (2015), Robot-centric activity prediction from first-person videos: What will they do to me, *in* 'Proceedings of the Tenth Annual ACM/IEEE International Conference on Human-Robot Interaction (HRI)', ACM, pp. 295–302.

Ryoo, M. S. and Matthies, L. (2013), 'First-person activity recognition: What are they doing to me?', *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* .

Simonyan, K. and Zisserman, A. (2014), 'Very deep convolutional networks for large-scale image recognition', *arXiv preprint arXiv:1409.1556* .

Sinapov, J., Khante, P., Svetlik, M. and Stone, P. (2016), Learning to order objects using haptic and proprioceptive exploratory behaviors, *in* 'Proceedings of the 25th International Joint Conference on Artificial Intelligence (IJCAI)'.

Sinapov, J., Schenck, C., Staley, K., Sukhoy, V. and Stoytchev, A. (2014), 'Grounding semantic categories in behavioral interactions: Experiments with 100 objects', *Robotics and Autonomous Systems* **62**(5), 632–645.

SPENCER Project (2016), http://www.spencer.eu/.

Spranger, M. and Steels, L. (2015), Co-acquisition of syntax and semantics — an investigation of spatial language, *in* 'Proceedings of the 24th International Joint Conference on Artificial Intelligence', Buenos Aires, Argentina, pp. 1909–1915.

Srinivasa, S. S., Berenson, D., Cakmak, M., Collet, A., Dogar, M. R., Dragan, A. D., Knepper, R., Niemueller, T., Strabala, K., Vande Weghe, M. et al. (2012), 'Herb 2.0: Lessons learned from developing a mobile manipulator for the home', *Proceedings of the IEEE* **100**(8), 2410–2428.

Stonier, D., Lee, J. and Kim, H. (2015), 'Robotics in concert', http://www.robotconcert.org/.

STRANDS (2016), http://strands.acin.tuwien.ac.at/.

Taylor, P., Black, A. W. and Caley, R. (1998), 'The architecture of the festival speech synthesis system'.

Tellex, S., Knepper, R., Li, A., Rus, D. and Roy, N. (2014), 'Asking for help using inverse semantics', *Proceedings of Robotics: Science and Systems* .

Tellex, S., Kollar, T., Dickerson, S., Walter, M. R., Banerjee, A. G., Teller, S. and Roy, N. (2011), 'Approaching the symbol-grounding problem with probabilistic graphical models', *AI Magazine* **32**(4), 64–76.

Thiébaux, S., Hoffmann, J. and Nebel, B. (2003), In defense of PDDL axioms, *in* 'International Joint Conferences on Artificial Intelligence (IJCAI)'.

Thomason, J., Sinapov, J., Svetlik, M., Stone, P. and Mooney, R. (2016), Learning multi-modal grounded linguistic semantics by playing "I spy", *in* 'Proceedings of the 25th International Joint Conference on Artificial Intelligence (IJCAI)', pp. 3477–3483.

Thomason, J., Zhang, S., Mooney, R. and Stone, P. (2015), Learning to interpret natural language commands through human-robot dialog, *in* 'Proceedings of the 24th International Joint Conference on Artificial Intelligence (IJCAI)', pp. 1923–1929.

Vasquez, D., Okal, B. and Arras, K. O. (2014), Inverse reinforcement learning algorithms and features for robot navigation in crowds: an experimental comparison, *in* 'Intelligent Robots and Systems (IROS 2014), 2014 IEEE/RSJ International Conference on', IEEE, pp. 1341–1346.

Veloso, M., Biswas, J., Coltin, B. and Rosenthal, S. (2015), Cobots: robust symbiotic autonomous mobile service robots, *in* 'Proceedings of the 24th International Conference on Artificial Intelligence', AAAI Press, pp. 4423–4429.

Walker, W., Lamere, P., Kwok, P., Raj, B., Singh, R., Gouvea, E., Wolf, P. and Woelfel, J. (2004), 'Sphinx-4: A flexible open source framework for speech recognition'.

Wisspeintner, T., Van Der Zant, T., Iocchi, L. and Schiffer, S. (2009), 'Robocup@ home: Scientific competition and benchmarking for domestic service robots', *Interaction Studies* **10**(3), 392–426.

Xia, L., Chen, C.-C. and Aggarwal, J. K. (2011), 'View invariant human action recognition using histograms of 3d joints', *IEEE Conference on Computer Vision and Pattern Recognition Workshop (CVPRW)* .

Xia, L., Gori, I., Aggarwal, J. K. and Ryoo, M. S. (2015), Robot-centric activity recognition from first-person rgb-d videos, *in* 'IEEE Winter Conference on Applications of Computer Vision'.

Young, S., Gasic, M., Thomson, B. and Williams, J. D. (2013), 'Pomdp-based statistical spoken dialog systems: A review', *Proceedings of the IEEE* **101**(5), 1160–1179.

Zhang, S., Jiang, Y., Sharon, G. and Stone, P. (2016), Multirobot symbolic planning under temporal uncertainty, *in* 'IJCAI'16 Workshop on Autonomous Mobile Service Robots'.

Zhang, S., Sridharan, M. and Wyatt, J. L. (2015), 'Mixed logical inference and probabilistic planning for robots in unreliable worlds', *Robotics, IEEE Transactions on* **31**(3), 699–713.

Zhang, S. and Stone, P. (2015), Corpp: Commonsense reasoning and probabilistic planning, as applied to dialog with a mobile robot., *in* 'AAAI', pp. 1394–1400.

Zhang, S., Yang, F., Khandelwal, P. and Stone, P. (2015), Mobile robot planning using action language bc with an abstraction hierarchy, *in* 'Proceedings of the 13th International Conference on Logic Programming and Non-monotonic Reasoning (LPNMR)'.