Integrating Answer Set Programming and POMDPs for Knowledge Representation and Reasoning in Robotics

by

Shiqi Zhang, B.S., M.S.

A Doctoral Dissertation

In

Computer Science

Submitted to the Graduate Faculty
of Texas Tech University in
Partial Fulfillment of
the Requirements for the Degree of

Doctor of Philosophy

Approved

Mohan Sridharan
Committee Chair

Hamed Sari-Sarraf

Michael Gelfond

Jeremy Wyatt

Dr. Dominick Casadonte

Interim Dean of the Graduate School

August, 2013

ACKNOWLEDGEMENTS

First of all, I would like to thank my Ph.D. advisor, Professor Mohan Sridharan, for his guidance and support during the past four years. The research presented in this dissertation will not be possible without his help. His help is on not only research ideas and solutions but also skills of professional presentations, technical writing and even spoken English (as my second language). I appreciate the opportunity of working with Professor Sridharan and I enjoyed it a lot.

I would like to thank the other committee members, Professors Michael Gelfond, Hamed Sari-Sarraf and Jeremy Wyatt, for their valuable comments and suggestions to this dissertation. I feel privileged and honored to have them as members of my dissertation committee.

I would like to thank my friends. Xiang Li and I joined Stochastic Estimation and Autonomous Robotics Lab (SEARL) and will graduate from Texas Tech both at the same time. We discussed a lot on many topics including research, career and life. Batbold Myagmarjav, Daniel Holman, Kimia Salmani, Sarah Rainge and Han Xu brought a lot of laughter to the lab. Pulkit Tomar and Shahnewaz Shuvro had been my friends since the year in Abilene, Texas. I love the memories of hiking, fishing, hunting, playing/watching basketball, eating out and playing guitar with Forrest Bao, Changzhan Gu, Xiaozhen Xue, Bach Nguyen and Patrick Kahl. Thank you all, my dear friends.

I would like to thank my wife, Meng Wang. When I decided to move to the U.S. to pursue my Ph.D., it did not seem to be a hard decision for her to leave everything behind and come to join me. I appreciate her effort for me and our family. Finally and most importantly, I would like to thank my parents. Whenever I feel tired or depressed, they are always there waiting for me.

TABLE OF CONTENTS

Acknowledgements ii

Abstract v

1. Introduction 1

2. Related Work 5

2.1 Robots and Applications 5

2.2 Knowledge Representation and Answer Set Programming 9

2.3 Partially observable Markov Decision Processes 9

2.4 Hierarchical POMDPs 12

2.5 Techniques of Learning from Human 14

2.6 Existing Representation and Reasoning Architectures 15

2.7 Multirobot Collaboration 20

2.8 Other Related Work 21

2.9 Summary 23

3. Overview of the ASP+POMDP Architecture 25

4. Knowledge Representation and Logical Inference using ASP 29

5. Probabilistic Planning using Hierarchical POMDPs 37

5.1 Partially Observable Markov Decision Process (POMDP) 38

5.2 VS-POMDP for Visual Sensing 40

5.3 SP-POMDP for Scene Processing 43

5.4 Convolutional Policy of POMDP 48

5.5 Directed Re-weighting and Full-Path Planning 53

5.6 Summary 57

6. Combining ASP and POMDPs 58
6.1 Bias Generation 60
6.2 Belief Merging 64
6.3 Using Positive and Negative Observations 66
6.4 Knowledge Acquisition 68
6.5 Summary 70
7. Multirobot Collaboration 71
8. Experimental Results 77
8.1 Experiments with KRR Using ASP 78
8.2 Experiments with Hierarchical POMDPs 79
8.3 Experiments with ASP+POMDPs 93
9. Conclusions and Future Work 111
List of Tables 113
List of Figures 114
Bibliography 116

ABSTRACT

Mobile robots equipped with multiple sensors and deployed in real-world domains frequently find it difficult to efficiently process all sensor inputs, operate without any human input and have possibly-relevant domain knowledge in advance. At the same time, robots cannot be equipped with all relevant domain knowledge in advance, and humans are unlikely to have the time and expertise to provide elaborate and accurate feedback.

This dissertation presents a novel architecture for knowledge representation and reasoning in robotics. These challenges are addressed by integrating high-level logical inference with low-level probabilistic sequential decision-making. Answer Set Programming (ASP), a non-monotonic logic programming paradigm, is used to represent, reason with and revise domain knowledge obtained from sensor inputs and high-level human feedback. In parallel, a novel hierarchical decomposition of partially observable Markov decision processes (POMDPs) uses adaptive observation functions, constrained convolutional policies and automatic belief propagation to automatically adapt visual sensing and information processing to the task at hand. This POMDP hierarchy serves as the first key contribution of this dissertation.

The second key contribution is the merging strategy of ASP-based logical inference with POMDP-based probabilistic belief. This dissertation presents a principled generation of prior beliefs from the knowledge base represented by ASP and the prior beliefs are then merged with POMDP beliefs using Bayesian updates to adapt sensing and acting to the tasks at hand. In addition, the entropy of belief states is used to determine the need for human feedback and hence robots ask questions only when needed. At last, robots are enabled to learn from positive and negative observations to identify the situations where

the current task should no longer be pursed.

As a result, mobile robots are able to represent and reason with domain knowledge, retain capabilities for many different tasks, direct sensing to relevant locations and determine the sequence of sensing and processing algorithms best suited to any given task, using human feedback based on need and availability. Furthermore, the architecture is augmented with a communication layer to enable belief sharing and collaboration in a team of robots. All algorithms are evaluated in simulation and on physical robots localizing target objects in indoor domains.

CHAPTER 1

INTRODUCTION

Autonomous robots are robots that can work without continuous human guidance. People have studied autonomous robots for tens of years and still pay great attention on it because human guidance is frequently expensive and even impossible for robots. This dissertation will focus on autonomous robots that can create plans toward given goals with minimum help from human. As shown in Figure 1.1, autonomous robots work in a loop that includes three steps: *sensing*, *planning* and *acting*. First, a variety of embedded sensors enable robots to observe the environment that includes its collaborators and itself. For autonomous robots, human-robot interaction is a type of sensing as well. However, comparing with the non-autonomous robots with continuous human guidance, autonomous robots frequently have to learn from humans without the expertise to the tasks in hand. Second, autonomous robots need to create a plan on what to do in the next step(s) according to the observation history. Third, the plan will be executed using motors, actuators and/or other devices that can make changes in environments.

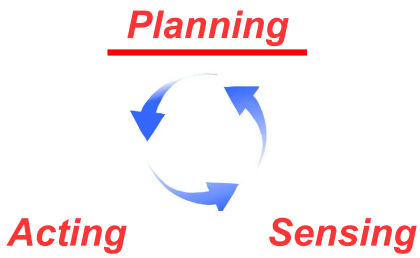

Figure 1.1: Overview process diagram for autonomous robots.

Sophisticated learning, planning and control algorithms have enabled the use of mo-

bile robots and agents to collaborate with humans in domains such as disaster rescue, reconnaissance and health care. These domains characterized by partial observability, nondeterminism and unforeseen changes frequently make it difficult for robots to process all sensor inputs or operate without substantial domain knowledge and human feedback. At the same time, it is impossible to provide all relevant domain knowledge in advance, especially for dynamic environments where domain knowledge can change by time. In human-robot coexisting environments, human can provide helpful information for robot planning. However, human feedback can be inaccurate and extremely expensive. Widespread deployment of robots in our homes, offices and other complex real-world domains thus poses formidable knowledge representation and reasoning (KRR) challenges—robots need to: (a) represent, revise and reason with incomplete knowledge; (b) automatically adapt sensing and navigation to the task at hand; and (c) learn from unreliable, high-level human feedback.

Although there is a rich body of research on knowledge representation and reasoning, the research community is fragmented. For instance, declarative programming paradigms provide commonsense reasoning capabilities essential for robotics but do not support probabilistic modeling of uncertainty, which is essential in robot application domains. In parallel, sophisticated algorithms based on probabilistic graphical models are being designed to model the uncertainty in sensing and navigation on robots, but it is difficult to use these algorithms for commonsense reasoning. Furthermore, algorithms developed to combine logical and probabilistic reasoning do not provide the desired expressiveness for commonsense reasoning capabilities such non-monotonic reasoning and default reasoning. This dissertation presents a novel planning architecture that exploits the complementary properties of ASP and hierarchical POMDPs by combining high-level logical inference with

low-level probabilistic modeling of uncertainty. As a result, the following contributions are made in this dissertation.

1. A novel POMDP hierarchy is developed to enable efficient and reliable operations on robots by decomposing complex problems into layers of simple POMDP problems with small numbers of states. The layers of POMDPs individually answer the questions of *where to look?, what to process?* and *how to process?*. This hierarchy is special in the adaptive observation functions, policy re-weighting and automatic belief propagation. In real-world domains, the number of states of POMDP can be large and the environment can change dynamically. A novel convolutional policy enables the robot to use the local symmetries (shift and rotation invariance) to firstly extract a policy kernel from a baseline policy and then extend the kernel to compute a large policy for the goal problem as needed. Other contributions in hierarchical POMDPs include full-path planning, directed re-weighting and multirobot collaboration algorithms.

2. ASP, a non-monotonic logic programming paradigm, is well-suited for common sense knowledge representation and reasoning (especially *default reasoning*). In this dissertation, robots use ASP to represent, reason with and revise spatial knowledge of the application domain (and domain objects), based on online repositories and information extracted from sensory cues and human feedback. A novel architecture is developed to benefit from the complementary strengths of ASP and POMDPs: POMDP belief is revised (or initialized) based on answer sets using a smooth biasing strategy; the knowledge learned from the expensive human feedback can contribute to the knowledge base represented by ASP—the need of human-robot interaction is measured by the entropy of POMDP belief.

All the algorithms are implemented and tested in both simulated and mobile robot platforms on localizing visually-distinguishable targets in human-robot coexisting indoor (office) environments.

The reminder of this dissertation is organized as follows. Section 2 discusses related work. Section 3 presents an overview of the architecture. The knowledge representation and reasoning using ASP are described in Section 4. The hierarchical POMDPs are presented in Section 5. The novel ASP+POMDP architecture is then discussed in details in Section 6 with an extension for multirobot systems described in Section 7. Experimental results in simulation and on robots are presented in Section 8. The last part is the conclusions and future work in Section 9.

CHAPTER 2

RELATED WORK

As stated in Chapter 1, the main contribution of this dissertation is a novel architecture for knowledge representation and reasoning (KRR) in robotics by combining probabilistic graphical models (POMDPs) and a non-monotonic knowledge representation paradigm (ASP). This related-work section will firstly focus on a few typical applications of autonomous and non-autonomous robots in Section 2.1. The first key part of the developed architecture is Answer Set Programming and existing work related to ASP would be introduced in Section 2.2. Existing POMDP techniques would be presented in Section 2.3, followed by related work to the second key part of this dissertation—hierarchical POMDPs, in Section 2.4. Human-robot interaction plays an important role in KRR for autonomous robots. A few HRI-based learning techniques are then presented in Section 2.5. Other existing planning and cognitive architectures are summarized in Section 2.6. There is a big community working on multirobot collaboration algorithms for years, and this dissertation extends the ASP+POMDP architecture with a multirobot collaboration algorithm. Therefore, in Section 2.7, I will briefly introduce a few existing solutions related to the POMDP-based multirobot collaboration strategy. Finally, the implementation of the architecture relies on many existing techniques and solvers, e.g., vision and speech processing, indoor mapping and localization, ASP and POMDP solvers, and Section 2.8 would focus on the well-studied techniques used in implementing the developed algorithms.

2.1 Robots and Applications

Sophisticated learning, planning and control algorithms have enabled the use of robots and agents in so many application domains that no article can cover all of them. The below

is just a small subset of the applications that can (hopefully) give an picture of how robots are used in real-world tasks.

The most widely used autonomous robot is probably Roomba produced by iRobot Corporation [41]. Roomba is an automated vacuum cleaning robot that can do cleaning tasks in indoor environments without human control and recent versions can return to recharge at the end of its cleaning cycle. Although the cleaning task is relatively simple and the working environments are well structured, Roomba is automated in creating plans within the three steps in Figure 1.1. According to iRobot [41], more than 8 million home service robots like Roomba have been sold worldwide by 2012.

Casper and Murphy presented robots as part of the specialized equipment in search and rescue in the World Trade Center disaster incident [17]. The paper discussed the skills displayed and needed by robots and humans, the details of the Urban Search and Rescue tasks, and what information should be communicated at what time. A more recent publication summarizing the research on rescue robots is [63].

A survey paper by Goodrich published in 2007 presented even more applications of robots in Social Interaction for autistic children, patrol support robotic classroom assistant, robotic museum tour guide, etc [34]. A more recent publication in 2010 by Hoey focused on a specific task—robotic handwashing assistance for persons with dementia [39]. Pineau's group developed SmartWheeler, a multi-modal intelligent wheelchair, and the goal is to increase autonomy and safety of individuals with severe mobility impairments [74]. Stanley, the Champion autonomous vehicle in 2005 DARPA Grand Challenge, has become a milestone in autonomous car development [96]. The Stanley project rooted in Stanford Artificial Intelligence Lab was led by Professor Sebastian Thrun. In 2005, only 5 out of 23 finalist vehicles finished the 212 km off-road course.

In the discussion of autonomous robots, it is necessary to mention the Robot Soccer World Cup (RoboCup) [47]. RoboCup is an international robotics competition founded in 1997. The aim is to promote robotics and artificial intelligence research by providing a set of standard challenges that are publicly appealing, but formidable. Mostly importantly, all of the robots used in RoboCup are fully autonomous and once the game starts the only input from any human is from the referee. Key challenges for these autonomous soccer robots include sensor fusion, robotics, real-time reasoning, multiagent collaboration and strategy acquisition.

Target Localization

Mobile robots are increasingly being deployed in applications such as assistive technology and disaster rescue due to the availability of affordable high-fidelity sensors and actuators. To move or manipulate an object in these domains, the object has to be readily available in the robot's field of view first. Previously researchers assumed human can help bring the target object into such positions or at least provide its position in given environment. In practice, however, human may have difficulties to localize the target objects due to long traveling distance, physical challenges or environmental safety. Target localization (TL) aims to relax the assumption by enabling a mobile robot to autonomously localize target objects without much human involvement. The performance is evaluated in both completing time and accuracy that are needed to be balanced properly. A TL policy makes sense when it at least performs better than the exhaustive or random search strategies. TL at the same time is a challenging task because it incorporates a set of sub-tasks including mapping, object recognition, spatial reasoning that are hard in themselves.

Recent research also exploits the semantic and common sense knowledge, e.g., a kitchen is typically used for cooking and a coffee maker usually appears in a kitchen. Instead, we

use a readily-available categorical tree represented by ASP to achieve the same goals of using semantic knowledgelogical reasoning on possible positions of target objects. In this specific task, the robot is mostly interested in spatial informations, e.g., locations of objects and if door is closed or not, as the domain knowledge. On the other hand, the robot has a mathematical model about itself, e.g., motion and vision parameters. This dissertation will be focusing on the planning algorithm while the motion control, speech and vision will be built on existing approaches. The TL problem would be taken as an illustrative example to address the effectiveness and efficiency of the designed algorithms. Since this dissertation will take a target localization task as the illustrative example, a few related publications would be summarized below.

Najemnik and Geisler derived an optimal eye movement strategy for searching a known target embedded at an unknown localtion within a random background [64]. Butko and Movellan further developed an Infomax model called I-POMDP to formulate the eye movement in a visual-search task as an information-gathering POMDP (I-POMDP) [14]. More recently, Nunez-Varela, Ravindran and Wyatt dicussed where a robot's gaze should be directed in order to gain information that is relevant to the success of its physical actions [68].

Aydemir, Pronobis and Jensfelt focused on the problem of active visual search and developed a search strategy using uncertain semantics [6]. An early version of this work was published in [5]. The search environment was represented by a probabilistic model and the authors developed a method for reasoning about the unexplored part of the environment for object search. The proposed approach was compared with a greedy coverage-based search strategy and search strategies of human participants.

2.2 Knowledge Representation and Answer Set Programming

Gelfond and Lifschitz developed the Stable Model Semantics for Logic Programming in 1988 [30] and the work later served as the base of Answer Set Programming, an instance of non-monotonic knowledge representation paradigm, that has been well studied and widely used in many applications [7, 29, 12].

Research in classical planning has resulted in many sophisticated algorithms for knowledge representation (KR) and logical reasoning [31], which have also been used on mobile robots [24, 98]. However, these algorithms typically require a significant amount of prior knowledge regarding the domain, and the preconditions and effects of actions that an agent can perform in the domain. Many algorithms also do not support merging of new (and possibly unreliable) information from sensors and humans with the current beliefs in a knowledge base. Answer Set Programming (ASP), a non-monotonic logic programming paradigm, is well-suited for commonsense knowledge representation and reasoning, especially *default reasoning* [7, 28]. ASP has been used in different application domains, and it has been integrated with natural language processing for service robots [20]. However, real-world sensing and navigation are non-deterministic, and humans participants are unlikely to provide elaborate and accurate feedback. ASP is not well-equipped to model or reason with quantitative models of uncertainty in sensing and navigation for robots.

2.3 Partially observable Markov Decision Processes

First-order Markov property assumes that the next state relies on only the current state and is independent to all of the previous states. A Markov chain is a mathematical system whose underlying state transitions follow Markov assumption [67]. A Markov decision process (MDP) is an extension of Markov chains with action selections and a reward function. MDPs, as a set of optimization problems, have been broadly studied since 1950s [10].

If the reward function is well defined, a MDP problem can be solved by Dynamic Programming techniques [10]. Many Reinforcement Learning techniques have been developed to solve MDP problems whose reward function is unknown [92].

An important branch (more accurately general form) of MDP is Partially observable Markov decision processes (POMDPs) where the underlying state is not fully observable. An agent planner will have to *observe* the world to estimate the underlying state. POMDP, as an instance of probabilistic sequential decision-making, was mathematically defined by Kaelbling, Littman and Cassandra in 1998 [42] and since then many algorithms solving POMDP problems and many POMDP-based applications have been developed as will be presented below.

The first survey paper on POMDP applications was published in 1998 [18]. The paper introduced early applications of POMDP in industry such as machine maintenance, structural inspection, elevator control policies and fishery industry, early scientific applications such as autonomous robots, behavioral Ecology and Machine Vision and other applications in business, military, education and medical diagnosis in early years.

Since 2000, POMDPs have been widely used in many other applications. Hsiao, Kaelbling and Lozano-Perez developed a POMDP-based planner used for robotic manipulator in 2007 [40]. Hoey et al. focused on a specific problem—handwashing of persons with dementia [39]. The system took video as the input and provided assistance by verbal or visual prompts, or through the enlistment of a human care giver's help. Foka used POMDPs in developing a real-time autonomous-navigation robot [23]. More recently, Gobelbecker designed a sensing and navigation strategy on robots by POMDP to model the associated uncertainty [32]. Young, Gasic, Thomson and Williams published a review paper summarizing recent POMDP-based techniques for applications in Spoken Dialog Systems [103].

In 2010, The International POMDP Practitioners Workshop gathered researchers working on real-world POMDP problems in Toronto, Canada and more details can be found at [102].

The focus of this dissertation is an architecture for knowledge representation and reasoning using (ASP and) POMDP, so POMDP solvers are *not* quite related to the key contributions of this dissertation. Here I briefly review a few well-known techniques used for solving POMDP problems. Kaelbling used value iteration to construct the value function over belief space and further developed the Witness algorithm to improve the complexity of the value-iteration [42]. A policy gradient method was developed for reinforcement learning by Sutton et al. in 2000 [93]. This method paved the way for many policy gradient-based techniques solving POMDP problems. For instance, Aberdeen, Buffet and Thomas developed a POMDP solver that tracks sufficient statistics of the history instead of the history itself [1]. The statistics over history is mapped to control policies directly using policy-gradient reinforcement learning. This work including the POMDP solver named LibPG was later improved and published as [13] in 2009. Most of the POMDP problems in this dissertation was solved by LibPG.

Another branch of algorithms for POMDPs roots in Point-Based Value Iteration (PBVI) algorithm [75]. PBVI approximates an exact value iteration solution by selecting a small set of representative belief points. It is naturally strong in solving large POMDP problems, and has been popular since the first publication. A recent paper by Guy, Joelle and Robert surveyed the point-based POMDP algorithms [87]. The paper introduced the fundamentals of PBVI, explained its key concepts and ideas and included empirical analysis of the various point-based approaches as well.

Instead of aiming at global policies that can map any POMDP belief to an action, online POMDP solvers focus on online approaches that alleviate the computational complexity

by computing only local policies at each decision step during execution. Paquet presented one of the earliest online POMDP algorithms in his dissertation [71]. The effectiveness of the proposed approaches were demonstrated in the RoboCupRescue simulation environments. A comprehensive survey of online planning algorithms for POMDPs by Ross, Pineau, Paquet and Chaib-draa was published in 2008 [83].

Mixed observability frequently exists in robotic systems: even the underlying state may not be fully observable, some components of the state may still be so. Ong used a factored model to separately represent the fully and partially observable components of a robot's state and this factored representation can be combined with any point-base algorithms to compute POMDP solutions [69].

2.4 Hierarchical POMDPs

POMDP applications have been suffering the curse of dimensionality and the curse of history in POMDP algorithms [42, 75]. Although many advanced POMDP algorithms have been developed as summarized in Section 2.3, it is still a challenge to model complex real-world problems as POMDPs. Instead of solving large, complex problems directly, people have tried to decompose problems into layers of small POMDP problems that can be solved individually.

Theocharous developed a POMDP hierarchy for robot navigation in 2001 [94], which is probably the first time that *hierarchical POMDPs* appeared in literature. The authors started from hierarchical Hidden Markov Models (HMMs) and then extended to hierarchical POMDPs as a planning problem. In 2004, Theocharous further explored the advantages of representing hierarchical POMDPs as dynamic Bayesian networks (DBNs) [95]. Theocharous's approach used a topological map of the environment where the state abstraction has corresponding physical meanings. Especially, the abstract states are manually

defined so they represent places like a corridor. Different layers of POMDPs correspond to spatial maps of multiple resolutions.

In parallel of Theocharous's work [95], Foka developed the Robot Navigation-Hierarchical POMDP (RN-HPOMDP) in 2007 [23]. RN-HPOMDP focused on robot navigation problems (including localization, planning and local obstacle avoidance), took occupancy grid maps as the input and modeled environments at a much higher resolution comparing with Theocharous's work [94].

Pineau et al. developed a mobile robotic assistant to assist elderly individuals with mild cognitive and physical impairments [74]. The authors presented a hierarchical POMDP approach for high-level behavior control on the robot. The planning algorithm operates in a bottom-up manner and the the execution proceeds in a top-down manner. However, the parameters of the hierarchy need a significant amount of manual effort.

Sridharan, Wyatt and Dearden modeled a visual processing problem as hierarchical POMDPs in 2008 [90]. The goal was to plan a sequence of visual operators for regions of interest (ROIs) for reliable and efficient visual sensing and processing in tabletop scenarios. Higher-level (HL) POMDP plans a sequence of actions to process ROIs and lower-level (LL) POMDP selects specific visual operators for each individual ROI. HL-POMDP and LL-POMDP were strongly coupled. The HiPPo approach was further improved and published with more details for implementation in 2010 [89].

The construction of a POMDP hierarchy can take a lot of human effort and researchers have studied methods of automatically discovering POMDP hierarchies. Charlin, Poupart and Shioda developed a method to automatically discover a hierarchy by encoding the hierarchical structure as variables of the optimization problem [19]. Toussaint, Charlin and Poupart further improved [19] by using Expectation Maximization (EM) algorithm in

2008 [97].

2.5 Techniques of Learning from Human

Robots need learning algorithms to acquire novel skills or adapt itself to its environments with dynamic changes. This section will focus on existing techniques that help robots learn from human.

In 2001, Asoh et al. developed an office robot, Jijo-2, as a testbed for autonomous intelligent systems that interact and learn in the real world [4]. The robot could communicate with humans through a spoken-dialogue system in Japanese and navigate using models it learned by itself or through human supervision. Jijo-2 is a general-purpose robot and the intention of designing Jijo-2 was to make it physically embodied in the human world and could exhibit some generic aspects of intelligence [4]. As an early attempt of learning from human, Jijo-2 communicated with human through predefined dialog patterns. For instance, "Where am I?", by Jijo-2 and "You are in front of Dr. Asano's office." replied by human.

In addition to learning from human-robot dialogs, robots can learn to accomplish tasks from *examples* as well. Argall comprehensively surveyed the techniques of *Learning from Demonstration (LfD)* in 2008 [3]. In contrast of most of the other learning techniques that enable robots to learn a policy, i.e., a mapping from world state to actions, from experience, LfD enables robots to learn a policy from example executions. Two critical challenges within LfD are discussed separately: example gathering methods ranging from teleoperation to imitation and policy deriving (from the gathered examples) including matching functions, dynamics models and plans.

Human can provide valuable feedback to robots, but human feedback is not always available and these high-level feedbacks are hard for robots to learn from. Knox developed TAMER (Teaching Agents Manually via Evaluative Reinforcement) framework that com-

bines manual feedback with MDP reward signals for reinforcement learning [48]. People frequently have difficulties to provide step-by-step instructions for robots, but it is easier for people to tell what is good (bad). A realistic assumption is made: people can give only positive and negative feedback signals. The authors concluded that TAMER are most useful for: tasks that require much exploration before discriminatory reward is received; tasks with local maximums that make the best solution difficult to find; and tasks with a noisy MDP reward signal. Sridharan further combined human and environmental feedback while enabling the two feedback mechanisms bootstrap off of each other to continuously revise their relative contributions to the agent's policy [88]. Therefore all the available information is exploited as much as possible.

Humans are not always available to help robots and human feedbacks are not always accurate. Rosenthal introduced HOP-POMDPs (Human Observation Providers POMDPs) to learn human availability and accuracy online while the robot is executing its current task policy [82]. The HOP-POMDP incorporated availability and accuracy into the observation function when the robot performs action of "ask". The authors further focused on the situations where mobile robots need human assistance at spatially-situated locations [81]. There are many trade-offs involved in seeking help from humans including the waiting time at the help location, the time and potential interruption to find and displace people, and traveling cost to places where people are available. Rosenthal and Veloso developed a decision-theoretic algorithm to evaluate the possible choices in office domains and plan where to proactively seek help.

2.6 Existing Representation and Reasoning Architectures

Planning architectures or algorithms aim at constructing a policy that maps the current state to an action. A lot of research has been conducted on planning algorithms for tens of

years because it is one of the key capabilities for real intelligent agents. There are so many existing architectures, techniques and algorithms that there is no way to cover all of them in a single article. This section will select a tiny subset of the existing work, try to present a high-level picture over this area and make brief qualitative comparisons.

2.6.1 Cognitive Architectures

A more general research area is cognitive architecture that covers not only planning but other tasks including reasoning, solving problems, learning concepts, etc. The goal of cognitive architectures is to create intelligent agents that have cognitive abilities as humans. Many cognitive architecture have been developed since 1980s [51, 2, 9, 38].

Soar (originally stood for State, Operator and Result) is a general cognitive architecture created by John Laird, Allen Newell and Paul Rosenbloom in early 1980s [51]. It integrates knowledge-intensive reasoning, reactive execution, hierarchical reasoning, planning and learning from experience. Soar has been in development for thirty years and Laird's group added new features since the initial version. For instance, reinforcement learning, semantic memory, episodic memory, mental imagery and emotion model appeared in the most recent version [50].

ACT-R is another cognitive architecture developed by Anderson, et al. [2]. Unlike Soar, a functionality oriented cognitive architecture, ACT-R aims at Psychological plausibility. The language primitives are designed to reflect the theoretical assumptions about human cognition and these assumptions are based on facts derived from experiments in cognitive psychology and brain imaging. Similar cognitive architectures are CLARION [91] and EPIC [46]. A more recent cognitive system is CoSy published in 2010 [38]. The key competencies of CoSy are system architectures, scalable knowledge representation, adaptive embodiment, categorical perception, planning and error recovery, learning, and situated

dialog systems.

Batog et al. developed CogSys, a cognitive architecture with three layers, in 2008 [9]. The high-level symbolic planner creates abstract action plans to realized by lower levels. The mid-level stores and reasons about object-action episodes. The low-level, sensorimotor layer, connects sensory processors to motor procedures. The authors used a table cleaning task to illustrate the effectiveness of the general-purpose architecture.

2.6.2 Knowledge Representation and Reasoning with Probabilities

Probabilistic Logic was developed by Nilsson in 1986 [66]. As one of the earliest logic systems being able to model probabilities, Probabilistic Logic laid out the basic theoretic principles supporting logic with probabilities. However, no efficient inference algorithms were provided at that time. Following this path, various probabilistic first-order logics have been developed [36, 73].

A simple framework for probabilistic Horn-clause abduction (PHA) was developed by David Poole in 1993 [76]. That framework is special in each hypothesis is associated with a probability. Poole mathematically showed that any probabilistic knowledge representable in a discrete Bayesian belief network can be represented in that framework. This framework is an extension of pure Prolog to include probabilities. Poole further replaced the set of disjoint declarations in PHA by a choice space and developed Independent Choice Logic (ICL) in 1997 [77, 78]. ICL is a semantic framework that allows agents to make independent choices and at the same time provides the agents the consequence of the choices. Multiple agents can make observations in dynamic domains with uncertainty and select actions by understanding the consequences of the action selections.

Unlike PHA that is based on Horn clauses, Logic Programs with Annotated Disjunctions (LPAD) was developed based on disjunctive logic programs [99]. This representation

allowed people to directly derive a probability distribution over the set of Herbrand interpretations. Complex probabilistic knowledge in terms of a number of simple choices can be easily represented using LPAD comparing with previous work, e.g., [77].

Kersting and Raedt developed Bayesian Logic Programs by combining Bayesian networks with definite clause logic to enable the representation of objects and relations in 2000 [45]. The main idea was to establish a one-to-one mapping between true logical ground atoms and random variables. Stochastic Logic Programs (SLP), as a generalization of stochastic grammars, aimed to provide a structured definition of the probability distribution over a set of logical formulas by associating a probability to each clause [62].

As an early work on learning algorithms for probabilistic logic programs, Sato and Kameya developed PRISM (PRogramming In Statistical Modeling) in 1997 [85]. PRISM integrated logic programming with the general learning algorithms (specifically Hidden Markov Models or HMM) to enable programs to change their behaviors by learning from positive/negative examples. Typical applications of HMM could benefit from PRISM by the capability of describing statistical correlations between syntactic and semantic structures.

Markov Logic Network (MLN) was developed by Richardson and Domingos in 2006 [80]. MLN is a simple approach to combining first-order logic and probabilistic graphical models in a single representation. Each formula in the knowledge base is associated with a weight. Inference in MLNs is performed by MCMC (Markov Chain Monte Carlo) algorithms over the minimal subset of the ground network required for corresponding query. First-order logic as the knowledge representation engine cannot perform non-monotonic logical reasoning that limits MLN's application in commonsense reasoning. The conclusions cannot be drawn by first-order logic program will not be inferred by MLN. Another weakness of

MLN is the computational complexity caused by the grounding of all first-order formula.

BLOG (Bayesian LOGic) is a language focusing on reasoning about an unbounded number of unknown objects with unknown relations [61], e.g., tracking multiple people in a video sequence or identifying repeated mentions of people in a set of text documents. A BLOG model specifies a probability distribution over model structures of a first-order logic language and these model structures can include different sets of objects. BLOG has a well-defined syntax as a probabilistic programming language and recent description about the syntax can be found in [52].

Another attempt of combining knowledge representation and probabilistic reasoning is PLOG [8, 108]. PLOG is a declarative language that combines logical and probabilistic arguments in reasoning. The authors used Answer Set Programming (ASP) [29] and causal Bayes nets as the logical and probabilistic foundations respectively. As claimed by Baral in [8], the difference between first-order MDPs (e.g., MLN) and PLOG is that actions, rewards and utilities are inherent part of the former; one may encode them in PLOG though.

From a quite different view, Göbelbecker, Gretton and Dearden developed a Switching Planner that combines probabilistic planning with logical reasoning [32]. The Switching Planner switches between using a fast satisficing "classical" planner called DTPDDL (Decision Theoretic PDDL) to decide on the overall strategy and a POMDP-based planner to solve small abstract subproblems where probabilities need elaborate consideration. The overall performance of this Switching Planner is sensitive to the threshold deciding on which planner to use and the complementary features of the "classical" planner and POMDPs were not fully exploited. Another recent work focusing on combining logical and probabilistic reasoning for task and motion planning on robots is [43] by Kaelbling and Lozano-Perez.

Qualitative reasoning is another research area in AI. For example, frequently it may not be easy to tell the exact value of velocity of a car but it can be easy to tell whether the car is moving fast or not. Golińska-Pilarek and Muñoz-Velasco developed an approach to reason with qualitative velocity [33]. The use of logic provided a general framework that improved the capacity of reasoning and the paper presented an approach toward dealing with qualitative velocity.

2.7 Multirobot Collaboration

Many algorithms have been developed for multirobot and multiagent collaboration. Panait and Luke created a comprehensive survey paper on the "current" state-of-the-art in cooperative multiagent systems and the paper also pointed a few directions for future research [70]. More recently, Parker surveyed algorithms for distributed robotics [72]. Since the multirobot collaboration strategy of this dissertation builds on hierarchical POMDPs, this section will discuss only a few POMDP-related collaboration strategies.

Researchers have developed algorithms for decentralized information fusion, e.g., the decentralized delayed-state information filter enables heterogeneous agents to fuse information [16], and a decentralized information-gathering algorithm has been able to provide scalability, robustness and modularity [59]. DDSIF (Decentralized Delayed-State Information Filter) is a decentralized data fusion approach aimed to perform cooperative perception with data gathered from heterogeneous sensors [16]. This approach can provide estimation that is as good as that provided by a centralized system and reduce the impact of communication delays and latency at the same time. Especially, heterogeneous sensors can benefit greatly from this data fusion approach. However, these algorithms are not well suited to model the partial observability of robots deployed in dynamic domains.

Decentralized POMDPs (Dec-POMDPs) are being used for multiagent collaboration [49,

86]. Dec-POMDP model offers a rich framework for cooperative sequential decision making under uncertainty. Each agent has its own action capabilities and observation set. After taking an action, each agent receives a local observation and a joint immediate reward. Each agent makes decisions solely on its local information but the decisions can affect the global reward (state transitions and observations that are similar to naive POMDPs). However, the computational complexity of solving Dec-POMDPs is higher than that of default POMDPs [11].

Gymtrasiewicz and Doshi developed interactive POMDPs (I-POMDPs) [35] that enables agents to model the behavior (and preferences) of other agents as interactive beliefs with arbitrary levels of nesting. The state space for each agent is greatly inflated when the notion of agent models are incorporated into the state space. Therefore, I-POMDPs have high computational complexity and require significant domain knowledge.

An important factor in multirobot collaboration is the unreliability of communication between robots. Research has shown that complex communication strategies do not necessarily benefit robot teams engaged in collaborative tasks [84]. The POMDP hierarchy described in this dissertation supports automatic belief propagation and model generation, enabling robots to adapt sensing and information processing to the task at hand. The hierarchy is augmented with a communication layer for belief sharing and multirobot collaboration.

2.8 Other Related Work

Robotics is a interdisciplinary subject where many research areas highly rely on each other. Robotic knowledge representation and reasoning, as a subfield of robotics, is not extraordinary. The general-purpose KRR architecture developed in this dissertation will be grounded onto an indoor target localization problem. The implementation of this architec-

ture on physical robots tasked with localizing targets needs many existing techniques that are beyond the discussion topic here. This section will focus on the existing techniques used in this dissertation but not discussed in previous sections. Generally, the techniques below are not related to the key contributions of this dissertation and the descriptions would be very high-level.

2.8.1 Vision Techniques

SIFT (Scale-invariant feature transform) is a visual feature detector and descriptor developed by David Lowe in 1999 [58, 57]. SIFT feature descriptor is invariant to uniform scaling, orientation, and partially invariant to affine distortion and illumination changes. Calonder, Lepetit, Strecha and Fua later developed BRIEF (Binary Robust Independent Elementary Features) in 2010 [15]. The authors made use of binary strings as an efficient feature point descriptor. As a result, BRIEF is fast in both feature detection and matching.

The vision module of this dissertation builds on the existing work conducted in SEARL (Stochastic Estimation and Autonomous Robotics Lab) at the Department of Computer Science, Texas Tech University. Specifically, Li and Sridharan developed a novel algorithm to enable a mobile robot to better utilize the visual input to navigate safely in dynamic environments [53]. This algorithm used local image gradient cues (combination of MSER [60] and SIFT [57]) to characterize target objects reliably and efficiently; and used temporal correspondence of visual cues for robust localization and tracking of environmental obstacles. In 2011, Li et al. developed an approach that enables mobile robots to autonomously learn layered models for environmental objects using temporal, local and global visual cues [56]. Most recently, context and appearance cues have been added into the model to enable robots to learn the spatial arrangement of gradient features, parts-based models of image segments, color distributions and mixture models of local context [55].

2.8.2 Robot Operating System (ROS)

Robot Operating System (ROS) is a software framework providing operating system-like functionality for software development on physical robot platforms [79]. The key competencies are hardware abstraction, low-level device control, implementation of commonly-used functionality, message-passing between processes, and package management. In this dissertation, the algorithm implementation on physical robots (Erratic robot [100]) builds on ROS.

2.8.3 Sphinx

CMU Sphinx is an open source toolkit for speech recognition [101]. It is flexible, modular and pluggable, so new innovations in the core research of hidden Markov model (HMM) can be fostered the least effort. The Speech Recognition module in this dissertation builds on Sphinx-4. Since speech recognition is not the focus of this dissertation, the size of vocabulary used in the speech recognition module was trimmed greatly to keep a high recognition accuracy.

2.8.4 Clingo and LibPG

Clingo [25] stands for clasp on Gringo. Gringo is an Answer Set Programming (ASP) grounder [26], while clasp is an ASP solver [27]. The ASP programs in this dissertation will be solved using Clingo unless specified otherwise. Similarly the POMDP programs will be solved using LibPG unless specified otherwise [13].

2.9 Summary

This chapter presented the existing work related to this dissertation and specifically focused on modern applications of robots, ASP, (hierarchical) POMDPs, learning-from-

human techniques, multirobot collaboration and existing probabilistic logic programs.

Although ASP is strong in knowledge representation and logical reasoning, ASP in its default form can not do quantitative reasoning with probabilities. In contrast, POMDP is good at probabilistic reasoning under uncertainty, but the rapid increase in state space dimensions makes POMDP impractical for complex problems, even if advanced POMDP solvers (Section 2.3) and/or hierarchical decompositions (Section 2.4) are used. Although there are attempts to do logical reasoning while modeling probabilities at the same time (Section 2.6), they either lack the property of non-monotonicity, that is important for representing and reasoning with common sense, or cannot reason in real time in knowledge-intensive domains. To address the challenge of knowledge representation and reasoning under uncertainty in knowledge-intensive domains, this dissertation develops an architecture that integrates the knowledge representation and reasoning capabilities of ASP with the probabilistic uncertainty modeling capabilities of hierarchical POMDPs.

CHAPTER 3

OVERVIEW OF THE ASP+POMDP ARCHITECTURE

Autonomous operation is a key challenge to the deployment of mobile robots in real-world domains such as homes and offices. The partial observability, non-determinism and unforeseen dynamic changes of these domains frequently make it difficult for robots to operate without sufficient domain knowledge or human inputs. It is however infeasible to provide robots with all relevant domain knowledge (in advance), and humans are unlikely to have the time and expertise to provide elaborate and reliable feedback in complex domains. Hierarchical POMDPs (as an important component) enabled a team of robots to visually localize target objects using hierarchical partially observable Markov decision processes (POMDPs), where complex problems are decomposed into layers of POMDP problems that are individually solvable. Although POMDPs elegantly model the uncertainty in sensing and navigation, it is difficult to represent common sense knowledge or perform high-level reasoning with human inputs. This section presents an overview of the architecture combining Answer Set Programming (ASP), a non-monotonic logic programming paradigm, with hierarchical POMDPs. Domain knowledge is represented as rules and facts that capture the relationships between object classes, and ASP reasons with the available knowledge to initialize or revise the POMDP belief distributions. Sensory observations and human inputs cause POMDP belief updates and augment (or revise) the current knowledge modeled by ASP.

This dissertation focuses on a novel knowledge representation and reasoning architecture for autonomous robots as depicted in Figure 6.1. The *Knowledge Base* (KB) in ASP contains causal rules and domain facts. Currently, rules are hand-coded and facts are learned

from sensor inputs, human feedback and online repositories. The KB contains all domain knowledge that may or may not be related to the current tasks. For any specific query (or task), reasoning in the KB results in an *Answer Set* containing a set of grounded literals (Section 4). The uncertainty in sensing and navigation is modeled using POMDP belief distributions (Section 5). The answer sets from ASP initialize or revise POMDP belief distributions through a Dirichlet distribution (Section 6.1).

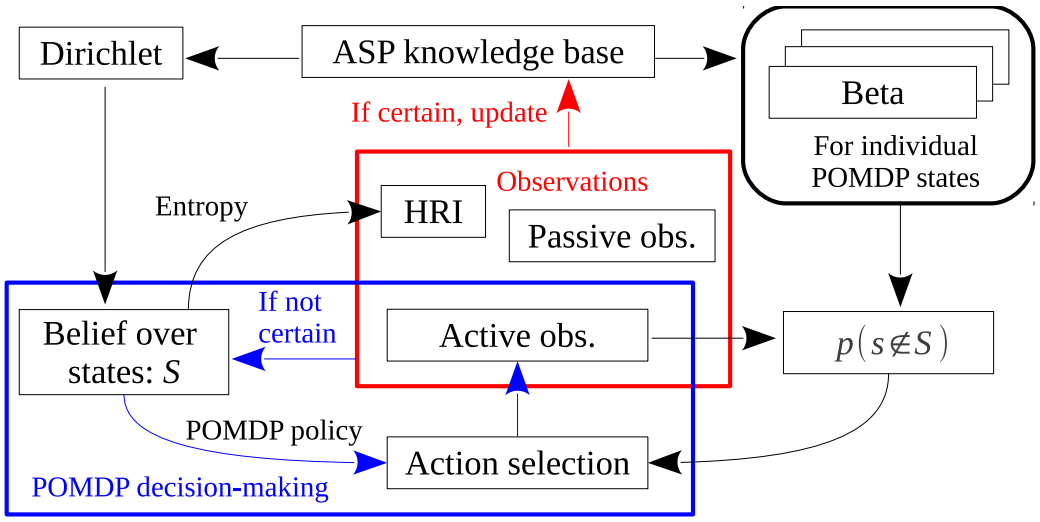

Figure 3.1: Overview of the architecture.
Architecture integrates probabilistic planning, non-monotonic logical reasoning and human-robot interaction.

The red box in Figure 6.1 includes three types of observations obtained from the environments. Active observations are directly relevant to the current task(s). The algorithms of active observations may not be real-time and can be computationally expensive. On the other hand, passive observations are not closely related to the current tasks and the corresponding algorithms are typically low-cost and real-time. For instance, passive observations enable robots to be capable of localizing itself, avoid running into obstacles and monitoring battery status all the time. The last type of observations is human-robot interac-

tion (HRI). Robots use speech recognition techniques to understand the feedback humans provide. Observations are classified based on the certainty level: observations with high certainty are used to update ASP knowledge base while the others update POMDP belief. The POMDP belief update is based on Bayesian rules.

Since human feedback is a valuable resource that is unreliable and not always available, human-robot interaction (HRI) is used when needed, e.g., if an object's location is known with considerable certainty, there is not much to gain by soliciting human help to locate the object. Robots therefore determine the need for human feedback based on entropy of POMDP belief distributions (Section 6.4). In this dissertation, the HRI strategy is passive (robot does not move physically to "seek" help): whenever the robot detects a human nearby, it computes the need and decides to ask or not.

Probabilities are modeled using hierarchical POMDPs (Section 5). The POMDP hierarchy includes two layers: Visual Sensing (VS) POMDP and Scene Processing (SP) POMDP. The blue box in Figure 6.1 shows the key components of the VS-POMDP: the VS-POMDP policy maps the belief state to an action; the selected action triggers an active observation that extracts related information from the environment; and most of the observations are then used to update the POMDP beliefs using Bayesian rules unless they are of a high certainty. The action selection of VS-POMDP determines which SP-POMDP to execute and SP-POMDP contributes to the active observations of VS-POMDP. SP-POMDP is not presented in this overview figure for simplicity.

It is possible that the underlying state does not belongs to the POMDP state set. For instance, the predefined state set includes all the possible positions of the target in the domain. However, what if the underlying state is that the target does not exist in this domain at all? The robot will keep searching until it is terminated by external actions, e.g.,

running out of energy. From this objective, this dissertation would develop a novel method that models a belief distribution for each POMDP state independently using a set of Beta distributions. These belief distributions are updated by positive and negative observations that are learned from standard POMDP observations. The probability that the underlying state is not included in the state set can be derived using the set of Beta distributions.

This novel architecture opens a window to integrate declarative languages with probabilistic reasoning tools by combining ASP and POMDPs. This architecture can be improved in many ways and Section 9 discusses some possible extensions.

CHAPTER 4

KNOWLEDGE REPRESENTATION AND LOGICAL INFERENCE USING ASP

Answer Set Programming (ASP)[1] is a declarative programming paradigm that can represent recursive definitions, defaults, causal relations, special forms of self-reference, and other language constructs that occur frequently in various non-mathematical domains, and are difficult to express in classical logic formalisms [7]. ASP is based on the stable model (i.e., answer set) semantics of logic programs [30], and has its roots in the research on non-monotonic logics.

As a knowledge representation (KR) language, ASP is defined by its signature, a tuple of sets: $\Sigma = \langle \mathscr{O}, \mathscr{F}, \mathscr{P}, \mathscr{V} \rangle$ [29]. These sets define the names of objects ($\mathscr{O}$), functions($\mathscr{F}$), predicates ($\mathscr{P}$) and variables ($\mathscr{V}$) available for use in the program. For instance, in a program about family relations, we can define `michael` and `julia` as objects and `father` and `mother` as predicates. If `michael` is the father of `julia`, `father(michael, julia)` is true and `mother(michael, julia)` is false. The use of functions and variables is similar to that in procedural programming languages. In addition, the following definitions will be used in the subsequent discussion of ASP [29]:

1. *Terms*: variable and object constants are terms, and a function of a set of terms is a term. Terms containing no symbols (for arithmetic functions) and no variables are called *ground*.

2. *Atom*: an atom is an expression of the form $p(t_1, \cdots, t_n)$ where p is a predicate and $t_1, \cdots, t_n$ are terms. An atom is called *ground* if every term t_i is ground.

[1] Answer Set Programming is also referred to as Answer Set Prolog.

3. *Literal*: a literal is an atom or its negation. Ground atoms and their negations are *ground literals*.

4. *Program*: a program Π consists of signature Σ and a collection of rules: l_0 or $\cdots$ or $l_i \leftarrow l_{i+1}, \cdots, l_m, \text{not } l_{m+1}, \cdots, \text{not } l_n$. The ls in the rules are literals of Σ.

An ASP program is thus a collection of statements describing domain objects and relations between them. An *answer set* is a set of ground literals that represent beliefs of an agent associated with the program. Program consequences are statements that are true in all such belief sets. The agent does not believe anything that it is not forced to believe. ASP supports *default reasoning*, i.e., conclusions can be drawn due to lack of evidence to the contrary, using concepts such as *default negation* (i.e., negation by failure) and *epistemic disjunction*. For instance, unlike "`¬ a`", which implies that "*a is believed to be false*", "`not a`" only implies that "*a is not believed to be true*"; and unlike "`p ∨ ¬p`" in propositional logic, "`p or ¬p`" is not a tautology. ASP also supports non-monotonic reasoning—adding a new fact can reduce the set of inferred consequences—causal reasoning, efficient reasoning in large KBs, and reasoning with quantifiers. ASP is thus well-suited for commonsense reasoning.

Knowledge Representation and Reasoning with ASP

In the illustrative target localization example, a robot learns and revises the domain map using range data and acquires semantic labels for rooms. The ASP domain description expands the signature by including symbols called *sort names*, which correspond to *types* in a procedural language, e.g.,:

1. `room`: a connected space bounded by walls and doors that can be occupied by objects and the robot. The predicate `room/1` is used to define a room, e.g., `room(hallway)`.

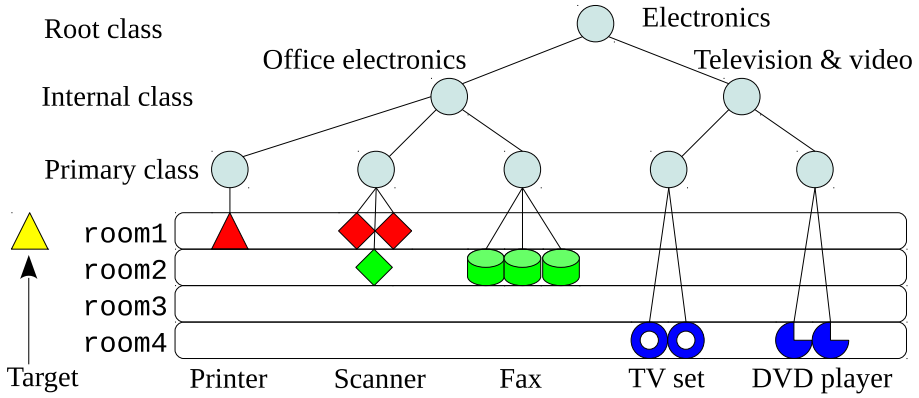

Figure 4.1: Illustrative example of object classes stored in the knowledge base.

2. `object`: a visually identifiable element in a room. The predicate `object/1` defines an object, e.g., `object(fridge1)`[2].

3. `class`: a set of specific objects or subclasses.

4. `step`: refers to a timestep (more details below).

The ASP KB encodes a *tree* of object classes as shown in Figure 4.1 for electronics items; leaf nodes are specific objects in specific rooms, while other nodes are object classes. Classes with specific objects as children are *primary classes*. Information is extracted automatically from online repositories [37] to identify some relationships between object classes. These relationships are used to create a subset of the tree, e.g., some of the nodes and links from root node to primary classes in Figure 4.1. Robots use information learned from sensor inputs and human feedback to add instances of objects to the KB and revise existing domain information. In this paper, objects are assumed to be visually distinguishable.

[2]This sort name is different from the set "Objects" ($\mathcal{O}$) in signature Σ.

Since certain aspects of the domain change dynamically, e.g., a room that was accessible may now be inaccessible, the concept of a *timestep* is introduced and modeled as a natural number (`I`). The robot uses predicates to model and reason about aspects of the domain that do not change (*statics*) and can change over time (*fluents*). Predicates are typically defined in terms of the sorts they take as arguments, and predicates dealing with fluents include the timestep as a parameter:

1. `is(object, class)` describes the class membership of a specific object, e.g., `is(tv1, tv)`.

2. `subclass(class, class)`, e.g., `subclass(C1, C2)` implies that class `C1` is a subclass of `C2`.

3. `in(object, room)` describes the room location of a specific object, e.g., `in(O, R)`.

4. `exists(class, room)`, e.g., `exists(C, R)` implies that an object of class `C` exists in room `R`.

5. `accessible(room)` is used to specify if a specific room is accessible, e.g., `accessible(R1)` implies `R1` is accessible.

6. `holds(fluent, step)` implies that a particular fluent holds true at a particular timestep.

Predicates are applied recursively when appropriate. In addition, the following rules are used for reasoning:

1. If object `O` of class `C` is in room `R`, it is believed that an object of class `C` exists in `R`;

```
holds(exists(C,R),I) :-holds(in(O,R),I), is(O,C).
```

2. If an object of class `C` exists in room `R`, it is believed that an object of the parent class (and all ancestor classes) of `C` exists in `R`;

```
holds(exists(C1,R),I) :-holds(exists(C2,R),I),
                        subclass(C2,C1).
```

3. An object can exist in only one location;

```
¬holds(in(O,R2),I) :- holds(in(O,R1),I), R1 != R2.
```

4. Rules of inertia: an object retains its location (i.e., it is in a room) until it is known to be elsewhere and a room remains accessible (inaccessible) until it is known to inaccessible (accessible).

```
holds(in(O,R1),I+1) :- holds(in(O,R1),I),
                       not holds(in(O,R2),I+1),
                       R1 != R2.


holds(accessible(R),I+1) :- holds(accessible(R),I),
                            not ¬holds(accessible(R),I+1).
```

Consider the following illustrative example of non-monotonic reasoning in ASP:

- *Test-case 1* has the following facts:

```
step(1..2).
holds(in(printer1, lab), 1).
is(printer1, printer).
```

  Reasoning in ASP produces the following answer set (existing facts are not repeated):

```
holds(in(printer1, lab), 2).
holds(exists(printer, lab), 2).
```

- Now consider *Test-case 2* that has a new fact about an object's current location:

```
step(1..2).
holds(in(printer1, lab), 1).
is(printer1, printer).
holds(in(printer1, office), 2).  %new info
```

  Reasoning in ASP now produces the following new answer set (existing facts not repeated):

```
¬ holds(in(printer1, lab), 2).
holds(exists(printer, office), 2).
```

Adding a new fact has thus revised the outcome of the previous logical inference step.

Next, consider modeling the default: "normally robots cannot climb stairs" with humanoids encoded elegantly as a *weak exception*:

$\neg$clmbstair(X) $\leftarrow$ robot(X), not ab($\mathrm{d}_{clmbstair}$(X)). % Default rule

robot(X) $\leftarrow$ wheeled(X).

robot(X) $\leftarrow$ humanoid(X).

ab($\mathrm{d}_{clmbstair}$(X)) $\leftarrow$ humanoid(X).

wheeled(peoplebot). humanoid(nao). % Specific data

where ab(d(X)) implies "X is abnormal with respect to d". The result of inference in this example is: $\neg$clmbstair(peoplebot) *without making any claims about* nao, i.e., it is unknown if humanoid robot nao can climb stairs or not. These capabilities are important for real-world human-robot collaboration. Inconsistencies caused by incorrect information being added to the KB are identified and corrected by processing sensor inputs or by posing queries to humans. The architecture (currently) only uses the inference capabilities of ASP to better explore the merging of qualitative and quantitative beliefs; future work will also consider ASP-based planning.

The architecture described in Figure 6.1 may cause incorrect information may be added to the KB. However, ASP is (by design) capable of commonsense reasoning in the presence of incomplete information. It can check for inconsistencies, which are corrected by subsequent sensor inputs or by (actively) posing relevant queries. Further details regarding ASP can be found in [7]. We use Clingo [25] to solve ASP programs.

This dissertation focuses on the logical inference and knowledge representation using ASP. Future work will use ASP for planning and diagnostic reasoning as well.

In knowledge-intensive domains, knowledge representation and logical reasoning are achieved using ASP. However, real-world environments are full of quantitative uncertainties, e.g., imperfect action outcomes and unreliable observations. ASP in default is not good at processing probabilities, so another strong tool, POMDPs, would be deployed in the next chapter to model these probabilities.

CHAPTER 5

PROBABILISTIC PLANNING USING HIERARCHICAL POMDPS

The architecture aims to achieve this objective by enabling robots to automatically learn environmental models (*Bootstrap Learning*), tailor sensing and processing to the task at hand (*Hierarchical Planning*). Furthermore, the individual components inform and guide each other to achieve the desired reliable, efficient and autonomous operation. This section primarily focuses on hierarchical planning and the corresponding algorithms are illustrated using a challenge task where robots have to locate objects (i.e., "targets") in dynamic indoor domains such as offices. The robot can then compute the 3D scene likely to contain the chosen target, move to a suitable location to capture images of the scene, analyze suitable regions in the images using relevant processing algorithms, and repeat these steps based on updated beliefs.

In the illustrative example of target localization in an office with multiple rooms, ASP-based reasoning will provide candidate (room) locations for each target object. To accurately localize a specific target object, the robot has to move and analyze a sequence of images of a sequence of scenes. This objective is posed as a planning task and addressed using our prior work on hierarchical POMDPs for (visual) sensor input processing—see Figure 5.1. For a specific target object, the 3D area is represented as a discrete 2D *occupancy grid*, each grid storing the probability of target existence. The visual sensing (VS)-POMDP plans an action sequence that maximizes information gain by analyzing a suitable sequence of scenes (Section 5.2). For each scene, the scene processing (SP)-POMDP plans the processing of specific regions of images of the scene using relevant algorithms (Section 5.3). Automatic belief propagation and model creation in the hierarchy are described

in [89, 104, 107]. For historical reasons, VS-POMDP is also referred to as higher-level (HL)-POMDP and when SP-POMDP is a two-layer POMDP planner, the two individual POMDPs are referred to as higher-level (HL)-POMDP and lower-level (LL)-POMDP.

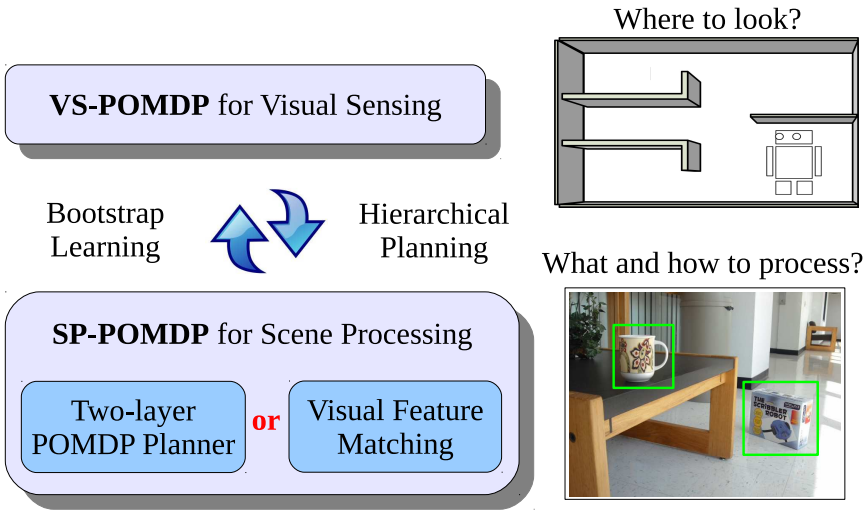

Figure 5.1: POMDP hierarchy and scenario

5.1 Partially Observable Markov Decision Process (POMDP)

Markov assumption (first order) indicates that the next state of the world only relies on the current state, and is independent from the previous states. Problems can be simplified greatly by Markov assumption because considering the states in history is not required for predicting the next state. Markov decision process (MDP) is a mathematical model used for decision-making where the underlying state transition is a Markov process. In MDP, the state of the world is assumed to be fully observable. However, for problems like robot planning, although the Markov property still holds, the actuations and sensing are unreliable. Partially observable Markov decision process (POMDP) is a generalized form of a Markov decision process, where the underlying state transitions are Markov processes

but the agent cannot fully observe the current state[42]. Hence, a probability distribution is maintained over the set of possible states to represent the current status of the world and this distribution is named as a belief state, belief distribution or a belief interchangeably. The belief distribution is updated based on the observations and the pre-learned observation functions.

A standard POMDP model is a tuple (S, A, Z, T, Ω, R), where

- S is a set of states of the world;
- A is a set of actions;
- Z is a set of observations the agent can experience of its world (Z reflects the sensor capabilities in case of the agent to be a robot) and;
- T is the state-transition function, giving for each world state and agent action, a probability distribution over world states. $T(s, a, s')$ is the probability of ending in state s', given that the agent starts in state s and takes action a;
- O: $S \times A \rightarrow \Pi(Z)$ is the observation function, which gives, for each action and resulting state, a probability distribution over possible observations. $O(s', a, o)$ for the probability of making observation o given that the agent took action a and landed in state s';
- R: $S \times A \rightarrow \mathbb{R}$ is the reward function, giving the expected immediate reward gained by the agent for taking each action in each state. $R(s, a)$ is the expected reward for taking action a in state s.

5.2 VS-POMDP for Visual Sensing

In real-world domains (e.g., an office or a home), target objects can exist in different locations in a room or in different rooms. A robot hence has to move to analyze different scenes and locate the objects. Without loss of generality, assume that the robot has learned a map of its world [22] (with walls and other static obstacles) using range information obtained from a laser range finder, and has to locate a specific target. The 3D area shown in the top right of Figure 5.1 is represented as a discrete 2D *occupancy grid*. Each grid-cell stores a probability that represents the likelihood of occurrence of the target object in that grid-cell. The size of each grid-cell is based on the field of view of the sensor (i.e., camera) being considered. The VS-POMDP poses sensing as the task of determining a sequence of actions that maximizes information gain, or equivalently reduces the entropy of the probability distribution over possible target locations. The VS-POMDP tuple $\langle S^{VS}, A^{VS}, T^{VS}, Z^{VS}, O^{VS}, R^{VS} \rangle$ for locating the desired target object in a domain discretized into N grid-cells, is then defined as follows:

- S^{VS} : s_i, $i \in [1, N]$ is the state vector; s_i corresponds to the event that the target is in grid-cell i.
- A^{VS} : $a_i, i \in [1, N]$ is the set of actions. Executing a_i causes the robot to move to and analyze grid-cell i.
- $T^{VS} : S^H \times A^{VS} \times S^{VS} \rightarrow [0, 1]$ is the state transition function. For actions that do not change the physical state of the system, it is an identity matrix.
- Z^{VS} : {present, absent} is the set of observations that indicates the presence or absence of the target object in a specific grid-cell.
- $O^{VS} : S^{VS} \times A^{VS} \times Z^{VS} \rightarrow [0, 1]$ is the observation function that is learned automatically (see below).

- $R^{VS} : S^{VS} \times A^{VS} \rightarrow \Re$ is the reward specification that is based on belief entropy (see below).

Since the robot cannot observe the true state of the world, it maintains a *belief state*, a probability distribution over the underlying set of states. The *entropy* of belief distribution B_t is then given by:

$$\mathscr{H}(B_t) = -\sum_{i=1}^{N} b_t^i log(b_t^i) \tag{5.1}$$

where b^i is the i^{th} entry of the belief distribution over the N grid-cells of the learned map of the domain. The reward of action a_t is then defined as the reduction in entropy between belief state B_{t-1} and the (expected) resultant belief state B_t after action execution:

$$\begin{aligned} R^{VS}(a_t) :=& \mathscr{H}(B_{t-1}) - \mathscr{H}(B_t) \\ =& \sum_k b_t^k log(b_t^k) - \sum_j b_{t-1}^j log(b_{t-1}^j) \end{aligned} \tag{5.2}$$

When nothing is known about the target object's location, the belief is uniformly distributed and entropy is maximum. As the belief distribution converges to states likely to be target locations, the entropy progressively reduces.

Similar to the reward specification, the observation function is specific to the domain and the characteristics of sensors mounted on the robot. As stated in Section 2, computing suitable observation functions can take a significant amount of prior knowledge or manual input. In this work, the observation function is defined adaptively based on the locations of

obstacles and the expected performance of the lower levels of the POMDP hierarchy:

$$O(z_i = present, s_j, a_k) = \begin{cases} \beta & \textbf{if } isBlocked(s_j, a_k) \\ \eta \cdot e^{-\frac{\lambda\mu^2}{2\sigma^2}} & \textbf{otherwise} \end{cases} \tag{5.3}$$

where $O(z_i = present, s_j, a_k) = p(z_i|s_j, a_k)$ is the probability of a particular observation in cell i given that the target is in cell j and the focus is on cell k. This probability is modeled as a Gaussian whose mean depends on target location, the grid cell being examined and the camera's field of view: $\mu = f_\mu(s_j, a_k)$. The variance is based on the observation functions of the lower levels of the hierarchy: $\sigma^2 = f_{\sigma^2}(O, O^I|s_j, a_k)$. When there is an obstacle between cell j and cell k, i.e., $isBlocked(s_j, a_k)$ is true, β is a small probability that the target can be observed while it is blocked by the obstacle(s). This observation function also models the fact that the probability of a robot detecting a target is inversely proportional to distance, i.e., a closer target is more likely to be found.

Given the model parameters, the belief update in a POMDP proceeds as follows:

$$B_{t+1}(s') = \frac{O^H(s', a_{t+1}, o_{t+1}) \sum_s T^H(s, a_{t+1}, s') \cdot B_t(s)}{p(o_{t+1}|a_{t+1}, b_t)} \tag{5.4}$$

POMDP solvers take such a model and compute a *policy*: $\pi^H : B_t \mapsto a_{t+1}$ that maps belief states to actions. In the VS-POMDP, the computed policy has to minimize the entropy in B_t over a planning horizon. The focus of this work is not on developing a POMDP solver but to provide a hierarchical planning scheme for achieving reliable and efficient visual sensing and processing using operators that are individually unreliable. Existing implementations of policy gradient algorithms are hence used to compute the high-level (HL) policy in the form of stochastic action choices, i.e., "weights" that are used to probabilistically choose

the *best* action for a belief state [13].

5.3 SP-POMDP for Scene Processing

For any chosen scene, the SP-POMDP plans the sequence of visual input processing (i.e., image processing) algorithms to be applied on a sequence of salient regions of interest (ROIs) of images of the scene. The SP-POMDP has one or two layers depending on scene complexity, i.e., the number of ROIs and types of features extracted from images of the scene. For instance, ROIs are extracted from each image of the scene and each ROI is modeled as a lower-level (LL) POMDP. Each LL policy provides the sequence of operators (i.e., algorithms) to apply on a specific ROI to detect the desired target object. These algorithms could, for instance, determine the dominant color or shape in the ROI. LL policies of all image ROIs are used to automatically create an HL-level (HL) POMDP. Executing an action in the HL policy directs attention to a specific ROI. Executing the corresponding LL policy (until termination) provides an observation that causes an HL belief update and action choice until presence or absence of the target in the image is determined. This provides an observation in the VS-POMDP followed by a belief update, with the robot choosing a scene for subsequent analysis. This process continues until the object is found or the belief does not converge over a period of time. The entire hierarchy adapts automatically to the task at hand. These two layers have been described in detail in prior work that focused on visual processing in a tabletop scenario [90, 89]. The description below is provided primarily for completeness.

Consider the situation where the robot has moved to a chosen scene and has captured an image of the scene. Assume that some initial processing has been performed to identify salient regions of interest (ROI) in the image. The goal of determining the presence or absence of the target in this image is posed in the form of a query. It is infeasible to apply

all possible processing algorithms on the entire image. The approach described in [90, 89] hence plans a suitable sequence of processing operators to be applied on a suitable subset of image ROIs.

First consider the POMDP model for a single image ROI, which determines the subset of available processing operators to be applied on the ROI for a specific query/task. For ease of explanation (and without loss of generality) assume that the robot has two visual processing operators: a $color$ operator that classifies the dominant color of the ROI it is applied on; and a *shape* operator that classifies the dominant shape within the ROI. Other visual operators can be added as needed, including operators that handle overlapping objects in the image [89]. Each action considers the true underlying state to be composed of the class labels (e.g., *red(R), green(G), blue(B)* for color; *circle(C), triangle(T), square(S)* for shape); a label to denote the absence of any valid object—*empty* (ϕ); and a label to denote the presence of *multiple* classes (M). The corresponding observation function is a probability distribution over the set of possible action outcomes. The set of action outcomes consists of the class labels; the label *empty* (ϕ) which implies that the match probability corresponding to the class labels is very low; and *unknown* (U) which implies that multiple classes are equally likely and the ROI may therefore contain multiple objects. Since the visual processing operators only update belief state, the action set also includes "special actions" that cause a transition to a terminal state by stating the presence or absence of the target object. The LL-POMDP for the ROI is then formulated as the tuple $\langle S^L, A^L, T^L, Z^L, O^L, R^L \rangle$:

- $S^L : S_c \times S_s \cup term$, the set of states, is a Cartesian product of the state spaces of the

individual actions. It also includes a *terminal* state.

$$S_c : \{\phi_c^a, R_c^a, G_c^a, B_c^a, M_c\}$$

$$S_s : \{\phi_s^a, C_s^a, T_s^a, S_s^a, M_s\}$$

- $A^L : \{color, shape, A_{sp}\}$ is the set of actions. The first two entries are the processing actions. The rest are special actions that represent responses to the query and lead to *term*. For instance, $A_{sp} = \{sFound, sNotFound\}$ while searching for a specific object.
- $T^L : S^L \times A^L \times S'^L \rightarrow [0,1]$ represents the state transition function. For visual processing actions it is an identity matrix, since the underlying state of the world does not change when they are executed.
- $Z^L : \{\phi_c^o, R_c^o, G_c^o, B_c^o, U_c, \phi_s^o, C_s^o, T_s^o, S_s^o, U_s\}$ is the set of observations, a concatenation of the observations obtained from all actions.
- $O^L : S^L \times A^L \times Z^L \rightarrow [0,1]$ is the observation function, a matrix of size $|S^L| \times |Z^L|$ for each action under consideration. It is learned by the robot for the visual actions, and it is a uniform distribution for the special actions.
- $R^L : S^L \times A^L \rightarrow \Re$ is the reward specification. The visual processing actions have negative rewards (i.e., costs) based on their relative computational complexity and the size of the ROI.

The reward and observation functions are learned by the robot in a semi-supervised manner, by repeatedly applying the available operators on known objects in the scene—see [89] for more details. Visual planning now involves solving this POMDP to find a policy that maximizes reward over a range of belief states. Plan execution corresponds to using the policy to repeatedly choose the action with the highest value at the current belief state, and updating the belief state after executing that action and receiving an observation.

Real-world scenes contain multiple objects, resulting in multiple ROIs in images of such scenes. The state space of the joint POMDP over all image ROIs grows exponentially. For a single ROI with m features (e.g., color, shape) each with n values (e.g., R_c^a, G_c^a, B_c^a, ϕ_c^a and M_c^a for color), the POMDP has an underlying space of size $n^m + 1$. For k ROIs we have $n^{mk} + 1$ states. Solving a POMDP in the joint space of all ROIs soon becomes intractable even for a small set of ROIs and actions—with three ROIs and two actions with six outcomes each, we get ≈ 50000 states. The hierarchy therefore models each ROI with a LL-POMDP, and uses the HL-POMDP to maintain a belief over the image and select (at each step) the ROI that is to be analyzed further using its LL policy. Such a decomposition changes the $O(n^{mk})$ problem for k image ROIs into one HL-POMDP with state space $2^k + 1$, and k LL-POMDPs with state space $n^m + 1$. Without loss of generality, consider an image with two ROIs—the corresponding HL-POMDP is given by the tuple $\langle S^H, A^H, T^H, Z^H, O^H, R^H \rangle$:

- $S^H = \{R_1 \wedge \neg R_2, \neg R_1 \wedge R_2, \neg R_1 \wedge \neg R_2, R_1 \wedge R_2\} \cup term^I$ is the set of states. It represents the presence or absence of the object in one or more of the ROIs, i.e. $R_1 \wedge \neg R_2$ means the object exists in R_1 but not in R_2. The set of states also includes a terminal state ($term^I$).
- $A^H = \{u_1, u_2, A_S\}$ are the actions. The sensing actions (u_i) denote the choice of executing one of the LL ROIs' policy trees. The special actions (A_S) represent the fact of "saying" that one of the entries of S^I is the answer, and they lead to $term^H$.
- T^H is the state transition function, which leads to $term^H$ for special actions and is an identity matrix otherwise.
- $Z^H = \{FR_1, \neg FR_1, FR_2, \neg FR_2\}$ is the set of observations. It represents the observation of finding or not-finding the object when each ROI's LL policy tree is executed.
- $O^H : S^H \times A^H \times Z^H \rightarrow [0, 1]$, the observation function of size $|S^H| \times |Z^H|$, is an uniform

matrix for special actions. For sensing actions, it is obtained from the LL policies of the corresponding ROIs.

- R^H is the reward specification. It is a small "cost" for each sensing action. And for the termination action, it is a large positive value, if it predicts the true underlying state correctly, and a large negative value, if not.

An important aspect of the decomposition is the automatic belief propagation—the HL observation function and reward specification are learned automatically from the LL policies of the image ROIs. For instance, the probability of FR_2 is computed by finding the leaf nodes that represents *sFound* in R_2's policy tree and summing the probabilities of traversing a path from each such leaf node to the root node. Similarly, the cost (i.e., reward) of action u_2 is the average cost of executing the actions represented by the corresponding LL-POMDP's policy tree [89].

The overall operation of the POMDP hierarchy is as follows: the map of the world is used to generate the HL-POMDP model, which is solved to obtain the HL policy for the desired target object. The HL policy is used to choose a 3D scene to analyze next. The robot moves to this scene and captures images. Each salient region of interest (ROI) in an image is modeled as an LL-POMDP, where actions are information processing operators (e.g., detect color, object class). The corresponding LL policy provides the best sequence of operators to apply on a specific ROI to detect the target object. The LL policies of all image ROIs are used to create an HL-POMDP during run-time, and executing an action in the corresponding HL policy directs the robot's attention to a specific ROI. The result from executing the corresponding LL policy causes a belief update in the HL and a subsequent action choice—the process is repeated until the presence or absence of the target is determined in the image being analyzed. The HL outcome causes an HL belief update and the

subsequent action choice causes the analysis of another grid-cell until the target is found. Changes in the domain map cause an automatic change in the HL policy, while changes in the perceptual state space (e.g., disambiguating between overlapping objects in the image) is addressed by re-planning in the HL-POMDP and LL-POMDPs [89]. The entire operation hence occurs reliably and efficiently due to the adaptive constrained convolutional policies and automatic belief propagation.

5.4 Convolutional Policy of POMDP

One key challenge in POMDP formulations of complex domains is that as the state space dimensions increase, obtaining real-time solutions is difficult even with sophisticated solvers. In the context of VS-POMDP, we addressed this challenge based on the observation that if the robot is analyzing a specific (grid) cell, the information gained by the robot is mainly a function of (and can influence) only a small number of surrounding cells. More generally, the policies of similar POMDP problems of different sizes have similar patterns. If the underlying patterns and its parameters can be discovered, it would be possible to avoid learning policies directly for complex problems and instead the policies could be derived using the parameters of underlying patterns.

Specifically, the robot learns a *policy kernel* from a baseline policy for a local region with a small number of grid cells, and generates the policy for a larger area with a larger number of grid cells by an inexpensive convolution operation—see Figure 5.2. For instance, given the baseline policy generated for a 5×5 map (with 25 states), the matrix of weights is reorganized into layers corresponding to each state and a 3×3 mask is convolved with the layers (and normalized) to generate the 3×3 kernel. This policy kernel can then be convolved with (say) a 10×10 map to generate the corresponding policy (one layer at a time). Although it may take some time for the robot to learn a baseline policy, the extracted

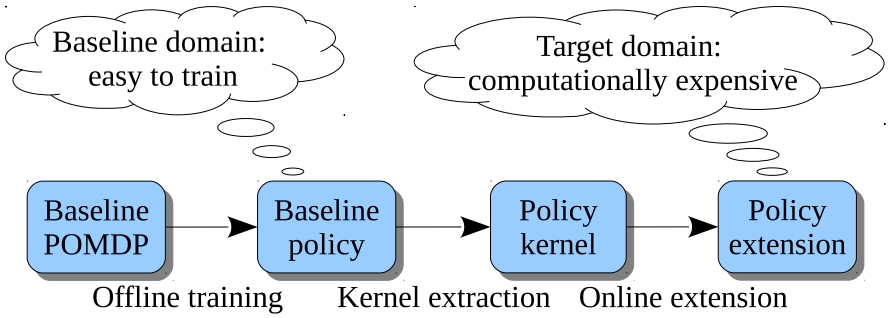

Figure 5.2: Motivation for convolutional policies.

kernel is a function of the sensors—it is typically computed once and used to create policies (in real-time) for larger-sized domain maps.

Executing an action using the VS policy causes the robot to move to and analyze a specific scene, creating and solving the corresponding SP-POMDP (with one or more levels) automatically. Executing the SP policy causes the robot to apply a sequence of visual processing algorithms on salient regions in images of the scene, resulting in a VS belief update and subsequent action selection. For more details on efficient and automatic belief propagation in the POMDP hierarchy, see [104].

5.4.1 Kernel Extraction of Convolutional Policy

Practical domains can change dynamically, have different shapes and sizes, and the number of grid-cells can be arbitrarily large. As a result, using POMDP formulations can be a formidable challenge. The hierarchy addresses this challenge by learning a convolutional policy kernel that exploits the rotation and shift-invariance of visual search. The hypothesis is that observations obtained by the robot at any location are primarily influenced by (and modify beliefs about) the neighboring locations [14].

A stochastic policy kernel is hence learned from the baseline policy for a small local

region:

$$\bar{K}(s) = (\pi^V \otimes C_m^K)(s) = \int \pi^V(\tilde{s}) C_m^K(s-\tilde{s}) d\tilde{s}, \tag{5.5}$$

$$K = (\sum_{states} \bar{K}) \cdot /W$$

where π^V is the (baseline) VS policy, C_m^K is a mask of the same size as the kernel being learned, $\bar{K}$ is the un-normalized kernel, W is the count of accumulated weights for each action and K is the normalized kernel. The kernel's size is chosen to be small (based on field of view of camera) and the baseline policy's size is chosen based on computational considerations. No other (e.g., shape) constraints are imposed on the kernel or the map. The policy for a larger area is then obtained by an inexpensive convolution of the map with the policy kernel.

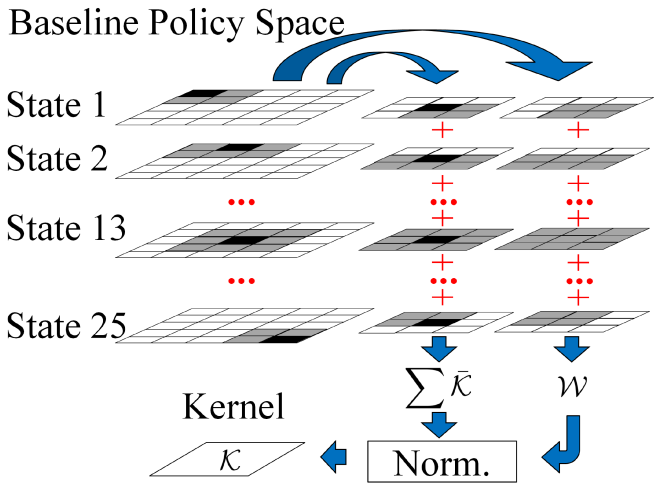

Figure 5.3: Illustration of extracting 3×3 policy kernel from a 5×5 baseline policy

Consider Figure 5.3, where a 3×3 policy kernel is extracted from a 5×5 baseline policy, a 2D matrix whose rows denote actions weights for specific states. Each row is re-arranged

to obtain a 2D matrix (of the same size as the map) that stores action weights when focusing on a specific state, decomposing the policy into layers as shown in the left column of Figure 5.3. A 3×3 mask C_m^K is convolved with the policy layers and the weights in the region covered by the mask are considered, as shown in the middle column of Figure 5.3. Since weights of the grid-cells outside the masked region are not considered, the resultant kernel is normalized (using matrix W) to obtain K, as shown in the right column of Figure 5.3.

The learned policy kernel does not assign action weights to the grid cells further away from the center of the mask. Since these weights are usually much smaller than values in the kernel, they are set to a small value:

$$W^B = \frac{\sum_{actions}\sum_{states}\pi^V - \sum_{states}\bar{K}}{N_{actions} \times N_{states} - sz(W)} \tag{5.6}$$

where the default weight value is a function of the number of actions: $N_{actions}$, the number of states: N_{states} and the size of the weight matrix: $sz(W)$.

5.4.2 Policy Extension of Convolutional Policy

The learned policy kernel has the most important parameters of the corresponding baseline policy, so it can be used to compute the policy for larger maps using an efficient convolution operation as below.

$$\pi_C^V(s) = (K \otimes C_m^E)(s) = \int K(\tilde{s}) C_m^E(s - \tilde{s}) d\tilde{s} \tag{5.7}$$

where K is the policy kernel, C_m^E is the mask of the same size as the target map and π_C^V is the convolutional policy. Consider Figure 5.4, where a 3×3 kernel is convolved with a 7×7 mask to generate the policy for a 7×7 map. This policy is generated one layer at a

time, by centering the kernel on the state represented by the layer, e.g., there are 49 layers for the 7×7 map. Since the kernel covers (at most) nine grid-cells, other cells are assigned the weight computed in Equation 5.9 and the policy is normalized. The robot can thus use the policy kernel to generate policies for larger areas in real-time.

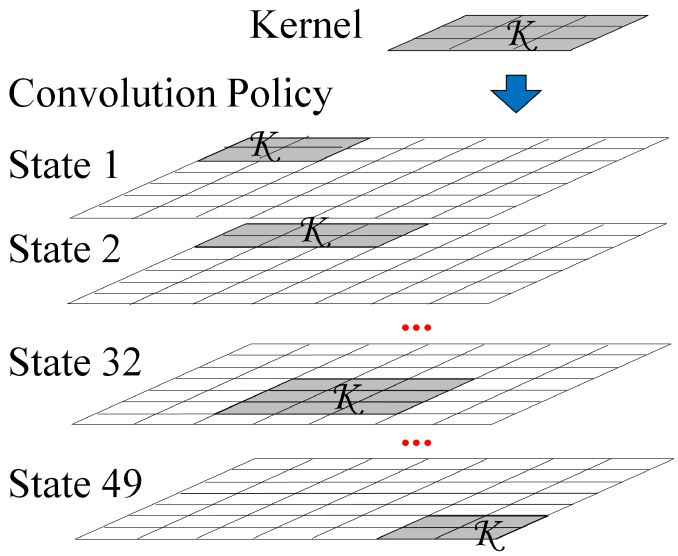

Figure 5.4: Illustration of using 3×3 policy kernel to generate 7×7 conv. policy.

However, when the policy kernel is used to generate policies for larger maps, the number of states covered by the kernel remains unchanged. Since policy weights over the map are normalized, the kernel's effect will be different on maps of different sizes (it will be much smaller when the size of the map grows larger). This is not a severe problem when the size difference between the baseline and the goal problems is not significant. Since the size of state space for baseline problem is typically small, the difference mostly relies on the size of the goal problem. Empirically, when the number of states of the goal problem is about 4 times larger than the baseline one, $\hat{W}^B$ needs to be recalculated as below.

Since parameters of POMDP policy range in $(-\infty, \infty)$, we want to project the parameters

onto a range of $[0,1]$ (as probabilities). The standard *softmax* activation function is hence used to convert weights to action probabilities (with a unit temperature value) [92].

$$p_t(a_j) = \frac{e^{w_t(a_j)}}{\sum_{i=1}^{N_{actions}} e^{w_t(a_i)}} \tag{5.8}$$

where, $p_t(a_j)$ is the probability of choosing action a_j at time t, and the individual action weights, i.e., the $w_t(a_i)$ values, are obtained from the computed HL policy.

A heuristic function is hence used to revise the value of W^B such that the ratio of importance assigned to the area covered and left uncovered by the kernel is similar over maps of different sizes:

$$\hat{W}^B = W^B - ln(\frac{N^E_{states} - sz(W)}{N^B_{states} - sz(W)}) \tag{5.9}$$

where N^E_{states} and N^B_{states} are the number of states in the large map and baseline kernel respectively. The natural logarithm function (ln) is used because the conversion of weight values to probabilities is based on a *softmax*-like activation function. Although it may take some time to learn a baseline policy for a small area and extract a policy kernel, this is a one-time computation that does not need to be repeated unless the properties of the robot's sensors change significantly.

5.5 Directed Re-weighting and Full-Path Planning

Directed re-weighting models the orientations of robots in addition to their positions. It is informative to robot planning, specifically for a target localization problem, because observing an object implies the object existing in front of the robot (beliefs on the states behind the robot should not change). During evaluation, the robot computes the relative

distance and bearing to the detected targets. However, including the computed orientation as a parameter in the observation set will destroy the shift and rotation invariance in policy space. As a consequence, the constrained convolutional policy will not be applicable. The belief update in Equation 5.4 is therefore modified as follows:

$$\begin{aligned}
&\textbf{if } \neg target \\
&\qquad B(s') = \frac{O(s',a,o)\sum_{s\in S} T(s,a,s')b(s)}{Pr(o|a,b)} = \frac{O(s',a,o)b(s)}{Pr(o|a,b)} \\
&\textbf{else} \\
&\qquad B(s') = \frac{O(s',\hat{a},o)\sum_{s\in S} T(s,\hat{a},s')b(s)}{Pr(o|\hat{a},b)} = \frac{O(s',\hat{a},o)b(s)}{Pr(o|\hat{a},b)} \\
&\textbf{end}
\end{aligned} \tag{5.10}$$

where, $B(s')$ is the updated belief for state s' after action a. Since the transition functions are identity matrices, the update equation can be simplified as shown.

The intuitive idea is straightforward: when a target is detected, the beliefs near the "target" increase (instead of the ones around the robot); when a target is not detected, the beliefs around the robot decrease. The relative distance and bearing are used to find the global location of the target in the grid map (based on robot's localization) and the belief proceeds as if the action corresponding to this global location had been executed: $\hat{a}$. This special belief update, known as *directed re-weighting*, is used in all experiments conducted on the simulated and physical robot platforms.

Mobile robots have to physically move between grid-cells in the map to search for target objects. Sensing, information processing and actuation on robots are non-deterministic and any movement by the robot takes time and expends energy. More importantly, the goal is to complete tasks with the least time. A cost is hence assigned to the robot's motion by

revising each action's (policy) weights during policy execution, based on the distance to be traveled and the robot's speed:

$$\hat{w}(a_j) = f(w, d_{A^*}) = w(a_j)\frac{1}{1+\frac{d_{A^*}(a_i,a_j)}{\text{speed}}} \tag{5.11}$$

where $d_{A^*}(a_i, a_j)$ is the distance between the current grid cell and the candidate grid cell. The A^* algorithm is used to compute the shortest path from the current grid-cell to a candidate grid-cell. The modified policy trades off likelihood of localizing the target against the cost of traveling to that location. The robot hence does not choose to travel a long distance between two sensing actions unless it expects to obtain a significant amount of knowledge about the location of the target object when it reaches the candidate grid-cell. When the domain map changes (e.g., doors are closed or some obstacles are moved), the robot uses this policy re-weighting approach to quickly recompute the distances between grid-cells and revise the action weights before making the subsequent action choices. Different speeds can be used in Equation 5.11 for different robots or situations.

While the revision of action weights captures motion-based costs, hill-climbing is used to make the search more efficient in large maps. The problem arises from designing reward in HL-POMDP as a function of entropy reduction [106]. This design forces the robot to move to places with potential to reduce belief entropy in the highest degree. However, a robot can hardly finish a task within a single step, so it is necessary to consider the potential of several steps after the next one. Consider Figure 5.5, which shows a domain map discretized into grid cells. The green grid is the current position of the robot after executing the most recent action. There are three grid cells in the map with significantly higher weights than the other cells: the orange and pink grids have $w = 0.3$ (not $\hat{w}$) and the blue grid has $w = 0.2$. Since the robot's current position is equidistant from the pink and orange grids, these grids

have an equal chance of being the next grid cell visited by the robot. However, given other valid candidates of similar relevance, it makes sense to visit the pink cell first because it is also close to the blue grid cell. We therefore enable the robot to consider the entire path of candidate grid cells, i.e., the path with the largest summation of $\hat{w}$ values instead of just looking for a target grid cell with the largest $\hat{w}$. It is however infeasible to estimate an optimal path by considering all possible paths through all grid cells in a large map. Our approach therefore detects "hot-spots", i.e., grid cells with beliefs substantially larger than the immediate neighborhood, and evaluates paths through them.

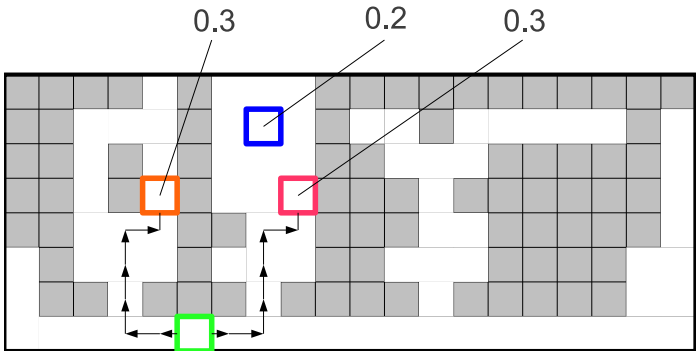

Figure 5.5: Hot-spot detection for motion planning.

To compute hot-spots, N^{hs} seeds are randomly selected and then refined based on hill-climbing to arrive at local maxima, e.g., the orange, blue and pink grids in Figure 5.5. These hot-spots are considered to be the interesting areas for further analysis. The robot then evaluates paths w^{path} through combinations of these hot-spots:

$$w^{path}([h_0, h_1, \ldots, h_{N^{hs}}]) = \sum_{i=1}^{N^{hs}} f(w(h_i), \sum_{j=1}^{i} d_{A^*}(h_{j-1}, h_j)) \tag{5.12}$$

where h_i is the ith hot-spot, h_0 is the robot's current position and $w(h_i)$ is the (action) weight at the grid-cell corresponding to hot-spot h_i. The function f is defined in Equation

5.11. In Figure 5.5, the values of the *pink-blue-orange* and *orange-pink-blue* paths are 0.0672 and 0.0591 respectively, making the pink grid cell the most likely choice for being analyzed next. This path planning does *not* imply that the robot will move through the entire path—once a robot arrives at a grid cell, the corresponding observation revises the belief distribution and planned path. The path planning ensures that the robot's attention is directed towards the most interesting areas instead of individual grid cells.

5.6 Summary

This novel POMDP hierarchy is one of the key contributions of this dissertation. *Bootstrap Learning* and *Hierarchical Planning* enable robots to automatically learn environmental models, tailor sensing and processing to the given tasks.

The VS-POMDP introduces convolutional policies, entropy reduction-based reward design, directed re-weighting and a full-path planning algorithm. The entropy reduction-based reward enables the robot to seek the most efficient way to make the belief distribution converge (to the underlying state). Directed re-weighting helps model the relative orientations of detected targets while not destroying the shift and rotation invariance of POMDP policies. Full-path planning algorithm enables the robot to consider not only a single region of interest but also a set of nearby regions for path planning in grids. Convolutional operations in the policy space enable the robot to avoid directly solving POMDP problems with a large number of states. Instead a policy kernel extracted from a similar but much smaller problem (so-called baseline policy) is used to generate the goal policy in real time.

The next chapter will focus on the novel architecture that combines hierarchical POMDPs (Section 5) and ASP (Section 4).

CHAPTER 6

COMBINING ASP AND POMDPS

Previous chapters have introduced the knowledge representation and reasoning with ASP (Chapter 4) and hierarchical POMDPs (Chapter 5). While ASP enables the robot to represent, revise and reason with incomplete knowledge and POMDPs enable the robot to automatically adapt sensing and acting to the task at hand, the complementary strengths of ASP and POMDPs have not been exploited. This chapter will focus on a novel knowledge representation and reasoning architecture combining ASP and POMDPs. Figure 6.1 depicts the architecture. The ASP knowledge base (KB) represents domain knowledge that may or may not be relevant to current tasks, while POMDPs probabilistically model a subset of state space directly pointing to the task at hand. Logical inference in ASP is used to: (a) compute prior beliefs relevant to the task at hand, which are modeled as a *Dirichlet* distribution; and (b) identify eventualities not being considered by the POMDP formulation based on learned *Beta* distributions. The Dirichlet distribution satisfies the objective of Bayesian merging with POMDP beliefs, while the Beta distributions enable robots to exploit both positive and negative observations. Since the POMDP state space is intentionally kept small and focused on specific tasks, it is possible that underlying state falls out of the predefined state set. In this case, the positive and negative observations are used to terminate tasks that can no longer be accomplished (Section 6.3).

The blue box includes a POMDP decision-making steps: the robot selects actions based on a learned *policy* that maximizes long-term rewards and takes as input the merged POMDP beliefs; action execution causes the robot to actively move to specific locations and process images to obtain observations; and (active) observations as a result of POMDP actions up-

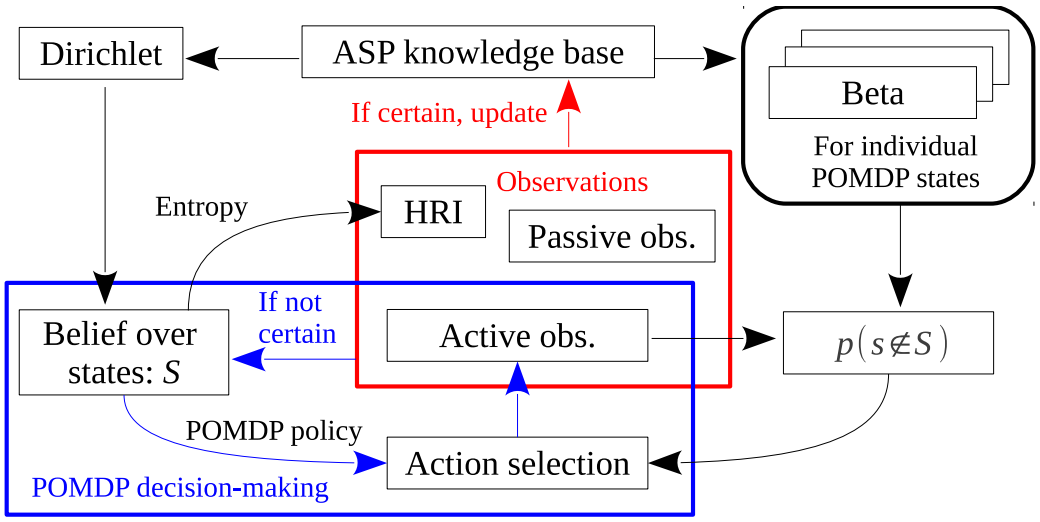

Figure 6.1: Overview of the architecture for knowledge representation and reasoning.

date POMDP belief using Bayesian rules. The red box indicates all types of observations, from which the robot learns the physical environment and creates plans on when to do what. Observations are not only activated by POMDP actions but also obtained from passive sensors (e.g., range finders, ultrasound) that keep providing sensor outputs no matter actions selected and from high-level human feedback that can have different levels of relevance to current tasks. Observations made with high certainty are (proportionately) more likely to update the KB, while other observations update POMDP beliefs. Human feedback is solicited based on need and availability, using the entropy of POMDP belief distributions to compute the need for feedback. Although this architecture is illustrated below in the context of robots localizing (i.e., computing the locations of) objects in indoor domains, the integration of knowledge representation, non-monotonic logical reasoning and probabilistic uncertainty modeling is applicable to many other domains.

Let us assume that ASP has provided candidate locations for a target object in a learned map of an office. The robot now has to move and analyze a sequence of images of a

sequence of scenes. This objective is posed as a planning task and addressed using our prior work on hierarchical POMDPs for reliable and efficient visual sensing and information processing on robots [104].

The ASP formulation (Section 4) models domain knowledge and provides an answer set that represents the result of non-monotonic logical inference. The POMDP formulation models the uncertainty in sensing and navigation to adapt sensing and processing to any given task. This section describes a principled strategy to convert answer sets to beliefs that initialize or revise POMDP beliefs. The entropy of POMDP beliefs is then used to identify the need for high-level human feedback, using information extracted from sensor inputs and human feedback to augment and revise the KB.

6.1 Bias Generation

It is a challenge to represent common sense knowledge in POMDPs, since the acquired information may have varying levels of relevance to current and future tasks. Since an answer set represents all that is currently believed to be true, This section describes our novel approach to generate a bias, i.e., a prior belief distribution over a set of possible underlying states, using the knowledge encoded in the ASP KB. For localizing objects in a set of rooms, a robot computes the prior belief of existence (or non-existence) of objects (in rooms). The robot uses knowledge of the object hierarchy—a tree of primary object classes (e.g., printer and scanner) that are subclasses of other classes (e.g., electronics)—and specific objects, and postulates that capture object co-occurrence relationships. We illustrate this approach using postulates for target localization and visual processing; some postulates (and their representation) may need to be revised when using this approach in other domains.

Postulate 1: Existence (or non-existence) of objects of a primary object class (in a room) provides support for the existence (or non-existence) of other objects of this class (in the room); level of support is proportional to logarithm of number of objects, inspired by *Fechner's law*[1]:

$$perception = ln(stimulus) + const$$

This law is applicable to visual processing, which is used in this paper to recognize objects. For a target object, support for its existence in a room is thus given by:

$$\psi_n = \begin{cases} 0 & if \;\; a_n = 0 \\ ln(a_n) + \xi & otherwise \end{cases} \tag{6.1}$$

where a_n is the number of (known) objects of the primary class (of the target) in the room, and $\xi = 1$ corresponds to *const* above. For instance, if the target is `printer1` and the robot knows that `printer2` and `printer3` are in a room, prior support for `printer1` in the room is: $ln(2) + \xi$. The probability of an object of class `C` to be located in room `R` increases with the number of objects of `C` known to be located in `R`. Certain domain objects may be exclusive, e.g., there is typically one fridge in a kitchen. Such properties can be modeled by relevant rules in the KB and by including other postulates (e.g., see below).

Postulate 2: As the number of known subclasses of a class increases, the influence that the subclasses exert on each other proportionately decreases. This computation is performed recursively in the object hierarchy from each primary object class to the lowest

[1]Fechner's law was introduced in 1860 and serves as the basis of modern Psychophysics. It states that subjective sensation is proportional to the logarithm of stimulus intensity.

common ancestor (LCA) of the primary class and the target object. Equation 6.1 is thus modified as:

$$\psi_n = \begin{cases} 0 & if \;\; a_n = 0 \\ \dfrac{ln(a_n) + \xi}{\prod_{h=1}^{H_n} W_h} & otherwise \end{cases} \tag{6.2}$$

where H_n is the height of the LCA of the candidate primary class and target. For a class node on the path from the primary class to the LCA, W_h is the number of its siblings at height h; we set $W_1 = 1$.

Postulate 3: Prior knowledge of existence of objects in different primary object classes *independently* provide support for the existence of a specific object (in a room). This postulate enables elegant merging of support from different sources, and works well in practice. For a target object, prior belief of existence in room k is thus the summation of evidence contributed by all N primary object classes:

$$\alpha_k = \sum_{n=1}^{N} \psi_{n,k} \tag{6.3}$$

The prior belief of existence of the target object in the set of rooms is modeled as a Dirichlet distribution with parameter set α, where α_k represents support for the target's existence in room k. The values of α_k are obtained as described above, using the cardinality of the set of relevant answer set statements obtained through inference in the KB. The probability density function (pdf) of this K-dimensional Dirichlet distribution is:

$$\mathscr{D}(\mu|\alpha) = \frac{\Gamma(\alpha_0)}{\Gamma(\alpha_1)\cdots\Gamma(\alpha_K)} \prod_{k=1}^{K} \mu_k^{\alpha_k - 1} \tag{6.4}$$

where Γ is the Gamma function used for normalization; $\mu_k \in [0,1]$ is a distribution of the target's existence over the rooms; and $\alpha_0 = \sum_{k=1}^{K} \alpha_k$. The expectation of this Dirichlet distribution, $\mathbb{E}(\mu_k)$, serves as the prior belief distribution that is to be merged with the POMDP belief distributions. The probability that the target is in room k based on the Dirichlet prior is given by (E_k represents the event that the target exists in room k):

$$p(E_k|\mathscr{D}) = \mathbb{E}(\mu_k) = \alpha_k/\alpha_0 \tag{6.5}$$

Although the Dirichlet distribution models the *conditional* probability of existence of the target in each room *given* its existence in the domain, it does not address the objective of learning from positive and negative observations, thus reasoning about the target's non-existence in a room or the domain. Towards this objective, parameters α also initialize Beta distributions that model the existence of the target in each room:

$$Beta(\phi_k|\alpha_k,\beta_k) = \frac{\Gamma(\alpha_k+\beta_k)}{\Gamma(\alpha_k)\Gamma(\beta_k)}\phi^{\alpha_k-1}(1-\phi^{\beta_k-1}) \tag{6.6}$$

where Γ function is used for normalization and β_k is the support for the non-existence of target in room k. The expectation of the Beta distribution for room k, $\mathbb{E}(\phi_k)$, serves as the prior belief that the target exists in room k, with E_k and E representing the existence of target in room k and the entire domain respectively:

$$p(E_k|Beta) = \mathbb{E}(\phi_k) = \frac{\alpha_k}{\alpha_k+\beta_k} \tag{6.7}$$

The probability that the target does not exist in the domain can then be derived as follows, assuming E_i and E_j are independent events $\forall i \neq j$.

$$p(\neg E) = \prod_k p(\neg E_k|\mathscr{B}) = \prod_k (1 - p(E_k|\mathscr{B})) \tag{6.8}$$

The Dirichlet distribution and Beta distributions thus elegantly model probabilities that support the desired objectives. The Dirichlet distribution models generalized conditional probabilities that directly use the information in the answer sets, while Beta distributions model specific (marginal) distributions that can be updated by positive and negative observations as described below.

6.2 Belief Merging

Section 6.1 has enabled to generate a bias distribution from answer sets using a principled strategy. The next question would be how to merge this bias distribution with a POMDP belief distribution. A straightforward method is an arithmetic or geometric combination. This section will introduce merging strategies using r-norm and Bayesian rules and these belief merging strategies would be experimentally compared in Section 8.

6.2.1 Belief Merging based on Trust Factors

The probabilistic bias distribution (for target occurrence) computed from the answer set is used to initialize or revise the POMDP belief vectors. Since the KB (and hence the answer set) can contain incomplete or outdated information, the answer set-based bias distribution and POMDP beliefs are merged using relative *trust factors*, resulting in a r-norm probability that is a generalized form of linear and logarithmic averaging methods [21],

e.g., it computes the arithmetic average for $r = 1$.

$$b_i' = \beta \left\{ (1-\Omega)(b_i)^r + \Omega(b_i^A)^r \right\}^{1/r} \tag{6.9}$$

where b_i^A is the answer set-based belief of target occurrence in room i, while b_i and b_i' are the beliefs of target occurrence in room i before and after belief merging (respectively), and β is a normalizer. The parameter $\Omega \in [0,1]$ represents the relative trust in the beliefs encoded by the answer set. The effects of Ω and r on accuracy and computational efficiency are analyzed experimentally in Section 8.3.

Consider the illustrative example in Figure 4.1. The corresponding answer set is used to compute the ASP-based bias distribution $b^A = [0.3890, 0.3361, 0.0000, 0.2749]$. The initial POMDP belief distribution (uniform in the absence of knowledge) is then revised as described in Equation 6.9, with $r = 1$ (arithmetic average) and the trust factor Ω set such that POMDP and ASP are trusted equally. The revised belief vector for the target is $[0.3195, 0.2931, 0.1250, 0.2625]$. The belief for each room is spread over grid cells in the room using a large-variance Gaussian centered in the middle of the room to induce the robot to move to a central location. Prior knowledge about likely locations of objects within rooms suitably revises the mean and variance of the Gaussian. The updated beliefs are used in the learned HL-POMDP policy to choose an action, resulting in the robot moving to analyze a specific scene.

6.2.2 Belief Merging based on Bayesian Rules

The KB contains domain knowledge, including information that may not be directly relevant to the current task. The POMDP belief summarizes all observations directly related to the current task, but these observations are obtained with varying levels of uncertainty.

Our prior work heuristically generated a belief distribution from answer sets and averaged it with POMDP beliefs. In this paper, the use of Dirichlet distribution to extract prior beliefs from answer sets supports Bayesian merging:

$$p'(E_k) = \frac{p(E_k|\mathscr{D}) \cdot p(E_k)}{\sum_i p(E_i|\mathscr{D}) \cdot p(E_i)} \tag{6.10}$$

where $p(E_k)$ is the probability that target is in room k based on POMDP beliefs.

6.3 Using Positive and Negative Observations

Although using lack of evidence for inference is a significant challenge, negative observations can be used to identify eventualities not modeled by the POMDPs, resulting in early termination of tasks that can no longer be accomplished. For instance, not observing the target or other related objects in multiple rooms can be used to reason about absence of the object in the domain; this is not computed in the standard POMDP model, and introducing a (special) terminal state in the POMDPs may invalidate the invariance properties exploited for computational efficiency.

This section describes our approach to exploit all observations. Let D be the event that the target is detected; and FoV the event that target is in the robot's field of view. The objective is to calculate $p(\neg E|D)$ and $p(\neg E|\neg D)$, and thus $p(E|D)$ and $p(E|\neg D)$, given the POMDP belief state B and action a. Towards this objective, the distribution of target existence and non-existence (in the domain) are updated in conjunction with the standard POMDP belief update. The prior beliefs computed in Section 6.1 are (re)interpreted as beliefs conditioned on the existence (or non-existence) of the target in the domain. Positive and negative observations are then handled as follows.

6.3.1 Negative Observations:

If the target is not detected, this observation should not change the probability of target's existence outside the robot's field of view, $p(\neg FoV|\neg D)$, which is the product of the probability of the target's existence in the domain, and the probability that the target exists outside the robot's view given its existence in the domain—the latter probability can be computed from the POMDP beliefs. The reduction in the probability of target's existence in the field of view will be added to the probability of non-existence of the target in the domain:

$$\begin{aligned} p(\neg E|\neg D) &= p(\neg E) + p(E)(p(FoV|E) - p'(FoV|E)) \\ &= p(\neg E) + p(E)(\sum_{s_i \in \Lambda(a)} B(s_i) - \sum_{s_i \in \Lambda(a)} B'(s_i)) \end{aligned} \tag{6.11}$$

where $B(s_i)$ and $B'(s_i)$ are the POMDP beliefs of state s_i before and after the Bayesian update using this observation; Λ, a function of action a, is the set of states that imply that the target is within the robot's field of view.

6.3.2 Positive Observations:

If the robot detects the target, the POMDP beliefs in the local field of view should grow, while beliefs outside this region and the probability of non-existence of the target (in the domain) should decrease.

The computation of the posterior probability of target's non-existence is based on the probabilities of the target being detected given that it exists or does not exist in the domain. The probability of the target being detected when it does not exist is assumed to be a fixed small probability of false-positive observations. The probability of target being detected when it exists depends on whether the target is inside or outside the robot's field of view.

If the target is in the robot's field of view, the probability of a positive observation can be computed using the POMDP observation functions; if it is outside the field of view, this probability is modeled as the probability of false-positive observations. The probability $p(\neg E|D)$ is then:

$$p(\neg E|D) = \frac{p(D|\neg E)p(\neg E)}{p(D|E)p(E) + p(D|\neg E)p(\neg E)} \tag{6.12}$$

The conditional probability that the target is detected given that it exists (or does not exist) is then given by:

$$\begin{aligned} p(D|E) &= p(D|E,FoV)p(FoV|E) + p(D|E,\neg FoV)p(\neg FoV|E) \\ &= p(D|FoV)p(FoV|E) + p(D|\neg FoV)p(\neg FoV|E) \\ &= \sum_{s_i \in \Lambda(a)} p(D|s_i,a)B(s_i) + \varepsilon \sum_{s_i \notin \Lambda(a)} B(s_i) \end{aligned} \tag{6.13}$$

$$\begin{aligned} P(D|\neg E) &= P(D|\neg E,FoV)P(FoV|\neg E) + P(D|\neg E,\neg FoV)P(\neg FoV|\neg E) \\ &= P(D|\neg E,\neg FoV) = \varepsilon \sum_{s_i \notin \Lambda(a)} B(s_i) \end{aligned} \tag{6.14}$$

where ε is the probability of false positives, i.e., detecting a target when it does not exist; $p(D|s_i,a)$ is obtained from the POMDP observation function.

6.4 Knowledge Acquisition

The final component of the architecture in Figure 6.1 is the acquisition of knowledge from sensor inputs and human feedback. To simulate high-level feedback from non-expert humans with limited time, human feedback is limited to simplistic verbal inputs. Human feedback may not be readily available and may be relevant to current and/or future tasks.

While soliciting human feedback, a robot has to consider multiple factors such as: time taken (or distance traveled) before a human is available to provide feedback; interruptibility of humans (possibly) engaged in other tasks; and expertise of human in the task currently being pursued by the robot [44, 81]. In this paper, the robot does *not* model individual humans, their expertise or their interruptibility. Instead, the focus is on enabling robots to verbally solicit human feedback when it is needed and available, and make best use of minimal (and possibly unreliable) feedback from non-expert humans.

As the robot moves in the application domain, images are processed periodically to detect humans (specific humans are *not* modeled separately). When a human is detected nearby, the robot computes the need for human feedback based on entropy of the belief distribution for the object being localized. A low entropy implies that the robot is confident of the target object's location—the human is then ignored (except for safe navigation). If the entropy is high, the robot draws the human's attention, followed by a query about a room's accessibility or the target object's location. These queries and responses are based on simplistic templates such as:

```
Robot: Where is the [object]?
Human: In [room]./I do not know.
Robot: Is [room] accessible?
Human: Yes./No./I do not know.
```

In addition to human feedback, the robot processes images at specific locations in the domain and low-resolution images as it moves between locations, detecting objects using learned object models. An object detected with high certainty is added to the knowledge base, using the detected position to form a suitable fact. This piece of information may be relevant to the current task and/or to future tasks. In addition to domain objects of interest,

robot may observe unforeseen changes in object configurations and obstacle locations, e.g., a door that was open may now be closed. The robot can confirm such changes using human feedback, and changes detected with high certainty also update the KB. These updates and additions to the KB occur incrementally and continuously, adding and eliminating areas for subsequent analysis. Furthermore, the robot may observe unforeseen changes in object configurations and the domain, e.g., a door that was open may now be closed. Such observations can also update the KB. The KB is thus augmented and revised incrementally and continuously, adding and eliminating areas for subsequent analysis.

6.5 Summary

This chapter focuses on the integration of ASP and hierarchical POMDPs. Section 6.1 presented the algorithms used to generate a bias for POMDP belief initialization or revision. When the KB changes in the running time, the new bias has to be merged with a POMDP belief that already includes previous observations. Section 6.2 discussed four belief merging strategies. It is possible that the underlying state is not included in the POMDP state set. In Section 6.3, task-level positive and negative observations are developed to identify such situations and enable the robot terminate pursuing the task early. Finally in Section 6.4, an entropy-based HRI strategy is developed to help the robot decide when to ask for human feedback.

In the next chapter, a multirobot collaboration strategy will be presented using part of the ASP+POMDP architecture, specifically the hierarchical POMDPs. The other components of this architecture would be added into this collaboration strategy later.

CHAPTER 7

MULTIROBOT COLLABORATION

Chapter 6 has introduced the novel architecture for knowledge representation and reasoning that can help a single robot create plans in real-world domains. One extension to this ASP+POMDP architecture has been made—a multirobot collaboration strategy. However, this extension builds on the hierarchical POMDPs and ASP is not used for historical reasons. This chapter will focus on the multirobot collaboration strategy that enables teamwork between multiple robots by adding a communication layer over hierarchical POMDPs.

The visual planning and collaboration algorithms are illustrated in the context of a team of robots localizing objects in large, complex and dynamic indoor domains such as offices. These algorithms represent an intuitive approach where each robot uses the current information about likely target locations to determine the object to locate next. The robot then computes the 3D scene likely to contain the chosen target, moves to a suitable location to capture images of the scene, analyzes suitable regions in the images using relevant processing algorithms, and repeats these steps based on beliefs updated by observations and information communicated by teammates.

Figure 7.1 summarizes the POMDP-based approach for sensing, processing and collaboration. Each robot uses a POMDP hierarchy to locate target objects. The top-level visual search (**VS**)**-POMDP** computes the sequence of 3D scenes to process to locate a specific target. For any chosen scene, the scene processing (**SP**)**-POMDP** uses one of two layers (depending on scene complexity) to determine the sequence of algorithms to apply on a sequence of regions of interest in images of the scene. Each robot also shares beliefs with

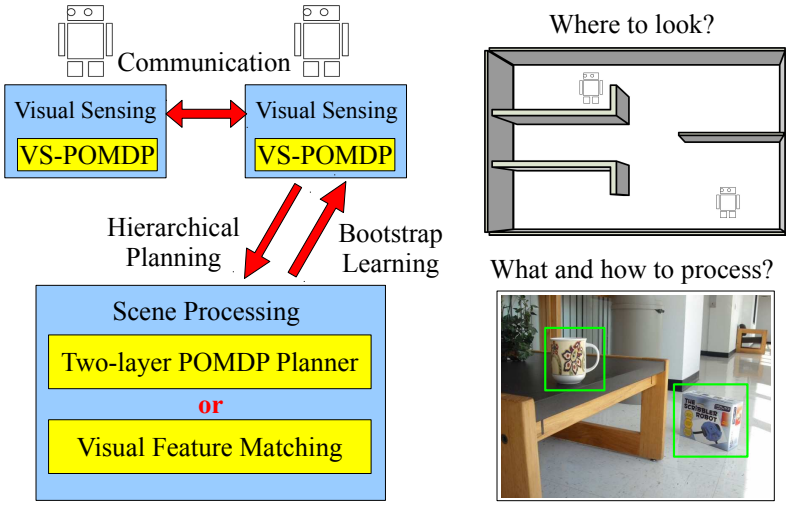

Figure 7.1: An overview of the POMDP hierarchy and scenario.

teammates to collaborate robustly despite unreliable communication.

Consider a team of X robots that is tasked with locating Y target objects in an office environment. Each robot in the team maintains a separate belief vector (over the domain map) for each target. Each robot also uses hierarchical POMDPs (as described above) to tailor visual sensing and information processing to the task and the domain. This section describes a probabilistic approach for the team of robots to share beliefs and collaborate robustly to locate the desired target objects.

For the experiments reported in this paper, the targets are assumed to be unique. It is also assumed that observations of different targets are independent of each other. In addition, in the experiments below, the targets are assumed to stay within a local area while they are being located by the team of robots. However, these simplifications are only used to study the effects of associated factors (e.g., prior knowledge and communication failures)—the POMDP hierarchy and the belief sharing approach (below) are suitable for objects that can move, as long as the movement is not very frequent. To enable multirobot collaboration,

each robot in the team stores a data structure as described below:

$$\{B_i, f_i\},\ \forall i \in [1, |TL|] \tag{7.1}$$

where B_i is the belief vector for a specific target i among the list of target objects (TL) and f_i is a binary flag that states if the target has been discovered. In addition, the robot stores an action map $\mathscr{M}$, a vector of the same size as the belief vector. Each entry in this vector stores the number of times the robot has visited the corresponding grid-cell in the domain map:

$$\mathscr{M} = \langle m_1, \cdots, m_N \rangle \tag{7.2}$$

where m_i is the normalized count of the number of times grid-cell i has been visited. The entries in the action map corresponding to locations that have not been visited in the recent past decay over time. As a robot moves to detect a specific target, it updates its action map and uses each observation to update the appropriate belief vector. After such a belief update, the robot communicates with teammates by broadcasting a UDP package that includes current belief vectors for all objects ($\forall i,\ B_i$), discovery flags ($\forall i,\ f_i$), action map ($\mathscr{M}$) and own position. If the bandwidth is limited, only changes in the data structure need to be communicated.

Since communication and sensing are unreliable, a robot cannot blindly trust information received from teammates. At the same time, the communicated estimates provide useful information about (possibly large) regions of the domain that the robot has not visited and hence has no knowledge about. In the belief merging scheme, each robot therefore assigns probabilistic weights to own beliefs and beliefs communicated by each teammate. The

intuitive idea is to assign greater importance to estimates communicated by a robot if the robot has visited the corresponding region of the map recently. Each robot therefore uses the action map entries as a probabilistic weight distribution to merge beliefs (acquired by sensing) with communicated beliefs:

$$b_i^{j,own} = \frac{m_i^{j,own} \cdot b_i^{j,own} + m_i^{j,comm} \cdot b_i^{j,comm}}{m_i^{j,own} + m_i^{j,comm}} \tag{7.3}$$
$$\forall j \in [1,N], \quad \forall i \in [1,|TL|]$$

where b_i^j is j^{th} entry of the belief vector corresponding to target i, while $m_i^{j,own}$ and $m_i^{j,comm}$ are entries of action maps of the robot and the teammate whose communicated belief is being merged. The action map entries are not merged to prevent rumor propagation among teammates. In addition, data association is achieved by matching the belief vectors (to the extent possible) based on the communicated locations of the robots. Each robot is thus able to robustly assimilate communicated estimates that may complement or contradict own beliefs, and the merged beliefs are revised as each robot's beliefs change. Although this belief merging approach can (in theory) be sensitive to the order in which the communicated beliefs are merged, it works well in practice.

Each robot also updates the vector of flags representing the discovery of target objects, i.e., f_i, by considering the efforts of all the robots in the team:

$$\mathscr{F} = \{f_i^{own} || f_i^{comm}; \forall i \in [1,|TL|]\} \tag{7.4}$$

where each target is assumed to be found when at least one robot in the team has communicated its discovery (belief in a grid above a threshold) to teammates. There may hence be times when a target object is being searched for by more than one team member. This

overlap of targets among robots in a team is allowed (intentionally) to ensure effective coverage of the desired target objects by robots in the team. However, in practice, multiple robots rarely search for the same target.

Once a target is discovered, a new target has to be chosen from the list of undiscovered objects in TL. This choice of the next target object is made as described below:

$$targetID = argmax_i\{\max_j B_i(j)\} \tag{7.5}$$

where the goal is to identify target i whose location the robot is most certain about based on the merged beliefs of all members in the team. In other words, the robot selects the target object that it is likely to discover (i.e., locate) with the least effort. This choice of a new target can also include a heuristic cost based on distance of travel and relative priority of the remaining targets (if such information is available). These costs can be incorporated as weights on the belief vectors, similar to the policy re-weighting in Equation 5.11. The key aspect of this collaboration approach is that robots in a team are able to reliably and efficiently coordinate their efforts towards a common objective. Dynamic changes in team composition are addressed automatically and the lack of communication causes each robot to smoothly transition to operating as if it were the only robot in the team. The next chapter evaluates the visual sensing, information processing and multirobot collaboration capabilities in dynamic domains.

This chapter presents a multirobot collaboration strategy where each robot shares its POMDP beliefs with teammates and each robot merges its beliefs with the communicated beliefs of teammates. As a result, a team of mobile robots is able to collaborate robustly in simulation and in the real-world. However, as an extension to the ASP+POMDP architecture presented in Section 6, this collaboration strategy does not use the capability of

non-monotonic knowledge representation of ASP. Future work will follow this path and develop a multirobot collaboration strategy using ASP+POMDP architecture.

CHAPTER 8

EXPERIMENTAL RESULTS

Previous chapters have presented the ASP+POMDP architecture for knowledge representation and reasoning for robots. This chapter will focus on evaluating the efficiency and effectiveness of this architecture. Specifically, the following hypotheses will be evaluated experimentally:

- combining ASP and POMDP improves target localization accuracy and time in comparison with the individual algorithms, including:
  - the constrained convolutional (CC) policy is more efficient than non-convolutional policy while providing similar accuracy;
  - the belief merging strategy enables a team of robots to share beliefs and collaborate robustly despite unreliable communication.
- the entropy-based strategy enables a robot to make best use of human feedback;
- using positive and negative observations helps robots identify tasks suitable for early termination.

Experiments have been comprehensively conducted in simulated and physical robot platforms on a target localization problem. Since it is a challenge to run many trials on robots, the architecture was evaluated extensively in simulation, using learned object models and observation models to realistically simulate motion and perception [55]. For instance, a simulated office domain consisted of four rooms connected by a surrounding hallway in a 15×15 grid as shown in Figure 8.1. Fifty objects in 10 primary classes (e.g., office electronics) were simulated, and (in each trial) some objects were randomly selected as targets

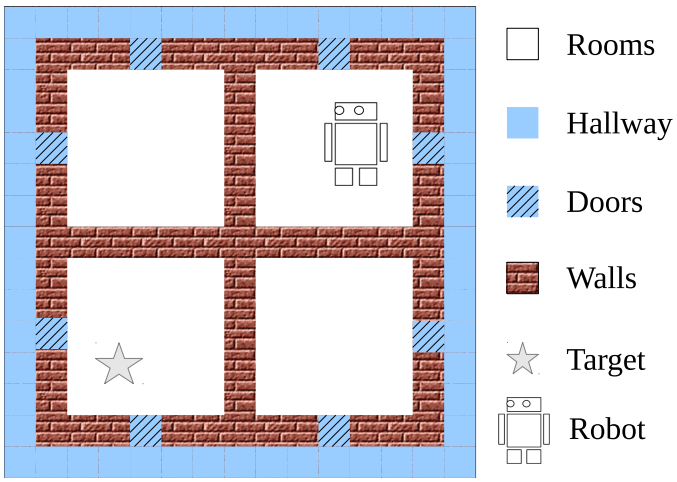

Figure 8.1: Simulated domain map.

whose positions were unknown to the robot. The robot revises the KB, including the basic object hierarchy mined from repositories, during experimental trials. Each data point in the results described below is the average of 5000 simulated trials. In each trial, the robot's location, target object(s) and location(s) of object(s) are chosen randomly. A trial ends when belief in a grid cell exceeds a threshold (e.g., 0.80); some trials included a time limit.

8.1 Experiments with KRR Using ASP

A robot was firstly tasked with using domain knowledge and Equations 6.1-6.4 to infer the target objects' locations. Note that ASP can only infer object existence in rooms and cannot infer the specific location of objects in rooms. Figure 8.2 shows that when the robot has all the domain knowledge, i.e., value= 1 along x-axis, it can (as expected) correctly infer the room containing the object. However, the accuracy decreases when the domain knowledge decreases, e.g., with 50% of domain knowledge, the robot can correctly identify the target's (room) location with only 0.7 accuracy. Also, with 50% domain knowledge, the correct room location is in the top two choices in 95% of the trials. When no domain

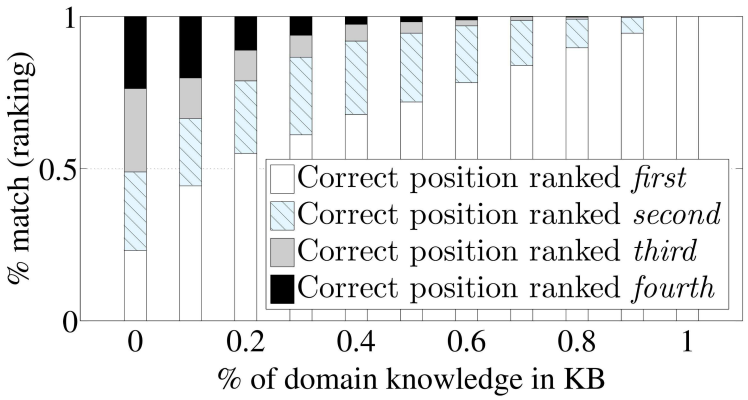

Figure 8.2: Target localization accuracy using only ASP.

knowledge is given (which corresponds to the left end in x-axis), the claim on target position would be a random guess, so it is evenly distributed over all possible rooms. One key expected outcome of using ASP-based inference (and the associated beliefs) is thus the significant reduction in target localization time.

8.2 Experiments with Hierarchical POMDPs

Experiments were designed to evaluate the following hypotheses: constrained convolutional (CC) policy provides similar detection accuracy to non-convolutional (i.e., baseline) policy but is much more efficient; CC policy significantly reduces time for reliable target localization compared with manually-tuned heuristic search strategies; and belief merging enables a team of robots to fully utilize prior knowledge and collaborate despite unreliable communication. The first hypothesis was evaluated in simulation while the second and third hypotheses were evaluated in simulation and on robots.

8.2.1 Experiment on a Simulated Robot

Simulation experiments were included because of the intractability of executing many trials on the physical robot. The simulator is realistic because it uses the same model parameters computed for the physical robot—Section 5.2. In each simulated trial, a grid map of a specific size was generated with the locations of the target object and the robot chosen randomly. Then, the ability of the robot to detect the target with different initial beliefs was analyzed.

A baseline policy was computed for a relatively small grid map: 5×5, from which the 3×3 policy kernel was derived. The kernel computation is an one-time process. Though the POMDP model for this grid has only 25 states, it takes ≈ 5 hours to get an acceptable policy with the average reward still growing. Using the convolutional policy hence provides significant benefits for larger maps.

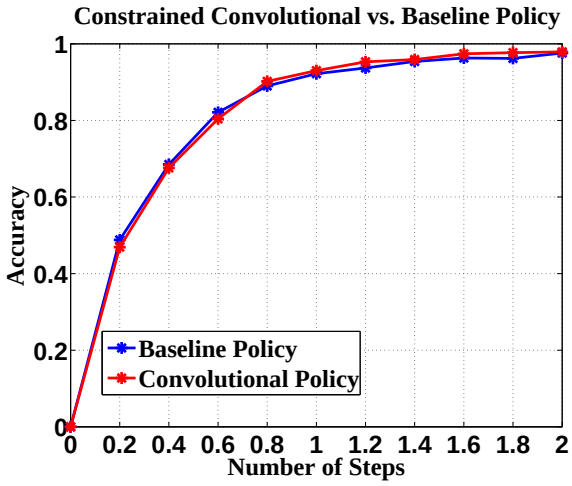

Figure 8.3: CC policies performance similarly to the baseline policy

First, the constrained convolutional (CC) policy was evaluated against the baseline (i.e., non-convolutional) policy on simulated grids. Over a set of 1000 trials (with targets at

random locations), the CC policy's detection accuracy was similar to the baseline policy. A trial was deemed successful if the target was identified in the correct grid-cell. Figure 8.3 shows the results for a 5×5 grid—the x-axis shows the number of times the policy was invoked, as a fraction of the number of states. Unlike the baseline policy, the CC policy was computed in no time from the 3×3 kernel.

The convolutional policy's performance was then compared against a policy that generates random actions, as a function of the number of actions the robot is allowed to execute (expressed as a fraction of the number of states). These experiments used a 15×15 convolutional policy generated from a 3×3 kernel. As before, the location of the robot and the target were randomly selected. In order to simulate prior knowledge of target location, 70% of the belief was uniformly distributed over all grids, and 30% of the belief was Gaussian-distributed surrounding the target. Each point in Figure 8.4 is the average of 1000 trials. At the end of each trial, the belief vector entry with the largest value was taken as the target location. The robot's performance was scored as the weighted distance between the actual target location and the detected location. The convolutional policy was observed to greatly reduce the number of steps taken to compute the target's location. The same CC policy was also used to compare directed re-weighting (Equation 5.10) against the policy execution without the special belief update scheme (Equation 5.4. Figure 8.5 shows that the directed re-weighting provides better performance despite added noise in the distance and bearing measurements.

The next experiment computed the number of actions required to achieve a high detection accuracy (0.95), as a function of the initial bias and variance. Table 8.1 reports the average number of action steps as a fraction of the total number of states. As expected, a smaller number of actions are required to find a target with a larger initial bias. In addition, if the

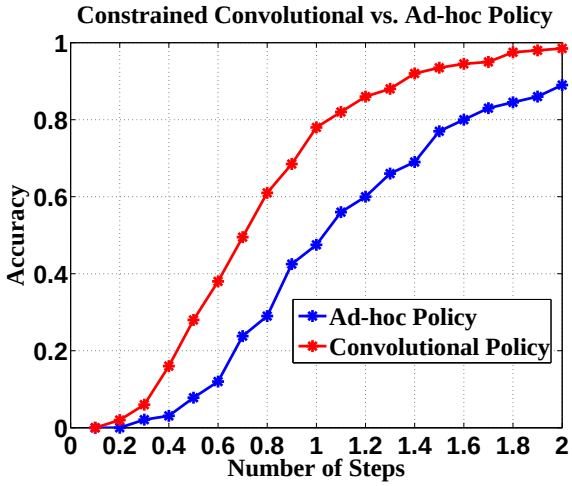

Figure 8.4: Convolutional policies performs better than a ad-hoc search strategy

initial bias has a larger variance, a larger number of actions are required to find the target.

Table 8.1: Average number of steps to achieve accuracy of 0.95

Bias		Covariance	
	0.1	0.2	0.3
10%	0.877	1.132	1.357
30%	0.521	1.019	1.274
50%	0.462	1.000	1.236

8.2.2 Experiments on a Wheeled Robot

The sensing and decision-making capabilities of a mobile robot (using hierarchical POMDPs) were evaluated on the *Erratic* robot platform shown in Figure 8.6. This robot is equipped with stereo and monocular cameras that provide 640×480 images at $30Hz$, and a laser range finder with an angular range of $\pm 135^o$ for a distance of $30m$. All processing is performed using an on-board dual-core $2.6GHz$ processor. Experimental trials were conducted in an indoor office domain and the corresponding occupancy-grid map, generated by a simultaneous localization and mapping algorithm, is shown in Figure 8.7. This map

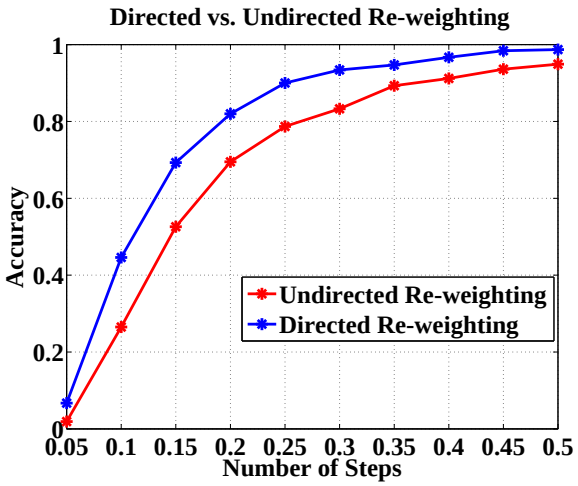

Figure 8.5: Accuracy comparison from simulated trials and directed re-weighting.

corresponds to an entire floor of the CS department at Texas Tech University—it has three research labs, 13 faculty offices, a conference room and a common area with a kitchen. The size of each grid cell in the map is $\approx 3m$.

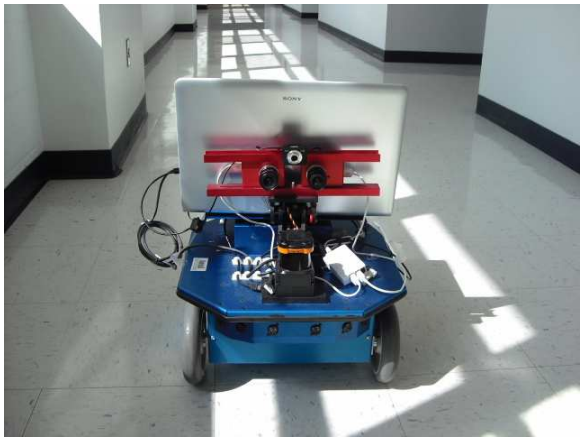

Figure 8.6: Erratic robot

The object models (learned by the robot) consist of color distributions and the Binary Robust Independent Elementary Features (BRIEF) [15], i.e., objects were characterized by color and local image gradient features. The robot classified cluttered and uncluttered

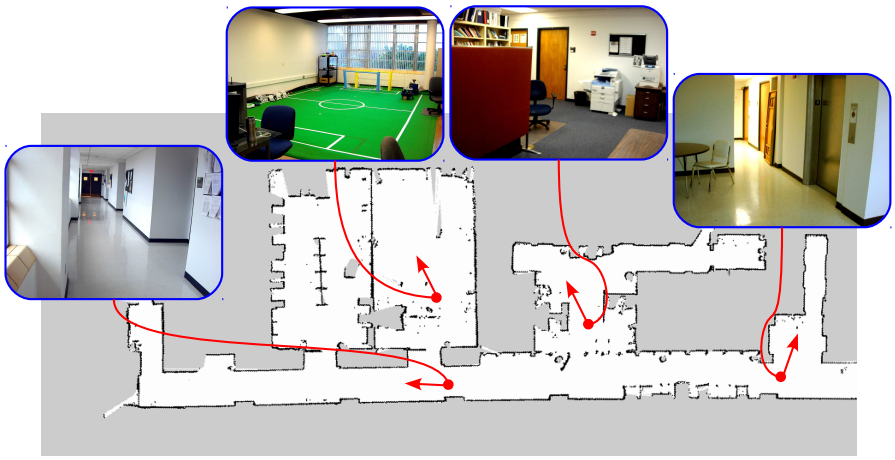

Figure 8.7: Domain map used in experiments.
Occupancy-grid map of the third floor of the CS department with 13 faculty offices, three research labs, a conference room and a common area with a kitchen.

scenes based on the number of ROIs and detected visual features. Although BRIEF features are not rotation and scale invariant, images of an object (captured during the learning phase) are automatically rotated and scaled to generate a set of images that model a range of rotations and scale changes—features extracted from these images are used to populate the object model. Figure 8.8 is a screenshot that shows the BRIEF features in a test image being matched with those in a learned model to recognize an object. Images from different viewpoints are considered at any given location. Target objects included boxes, cups, books and other robots in backgrounds with varying levels of clutter.

To enable modular software development, the algorithms were implemented within the Robot Operating System (ROS) [79] framework. Figure 8.9 presents an overview of a relevant subset of the implementation. Our planning algorithms are placed in the *vs_planner* node, which accepts messages from the *vs_vision* node that processes input images to populate the `<v_pack>` package. This package contains the ID of detected objects, distance and bearing of the objects (relative to the robot), and (probability) measures of the cer-

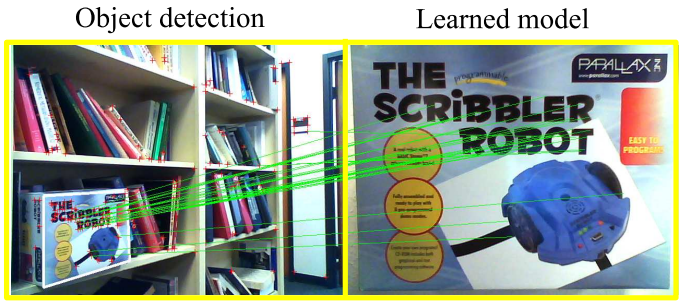

Figure 8.8: Illustrative example of the use of BRIEF descriptor.

tainty associated with the observations. Belief updates occur when the robot: (1) arrives at a desired grid cell and processes some images of the scene; or (2) processes images during navigation to a desired grid cell. The planner node sends coordinates of desired grid cells to the movement control node *move_base* and then waits for a response, e.g., `arrived`, `canceled` or `not-arrived`. The `not-arrived` response is usually caused by a change in the domain, e.g., closing a door makes an office inaccessible. The *hokuyo_node* provides laser readings to the motion control node and the localization node *amcl*. The platform driver node *erratic_base_driver* moves the robot platform based on the velocity command `cmd_vel`. The `position` and `goal` are sent and received by the *amcl* and *navigation_goals* nodes to aid in local path planning, localization and navigation.

The wheeled robot was asked to locate different objects in the domain shown in Figure 8.7—the target position and initial position of the robot were chosen randomly. The belief distributions were initialized to give the robot some prior knowledge of target locations. The left half of Table 8.6 summarizes target localization time for specific target objects (i.e., a subset of experiments). Each data point is an average of $10-15$ trials. Since the positions of robot and targets differ between trials, results for *random* and *heuristic* strategies are expressed as a multiple of the *proposed* strategy's results—e.g., the average time taken to localize the *Box* using the POMDP hierarchy is 4.08mins. Table 8.6 does not

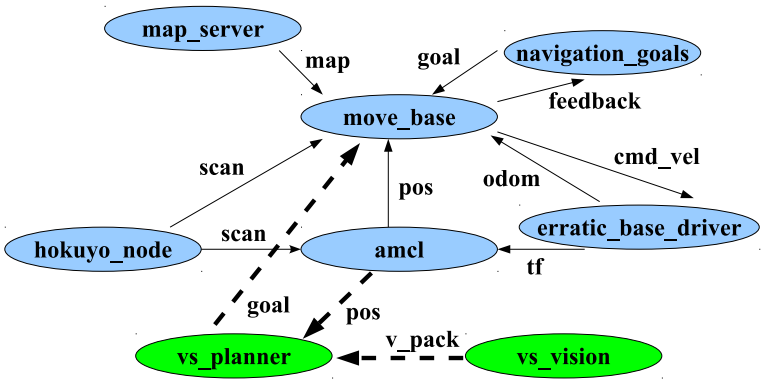

Figure 8.9: Connections between a relevant subset of nodes in the ROS implementation.

show result for the *random* strategy due to the large variance—many trials do not terminate even after 15mins. Across different target objects (boxes, robots, books), the POMDP hierarchy significantly reduces the localization time in comparison to heuristic (multiplying factor is 1.5) and random strategies. The results are more pronounced than in the simulated experiments because the real-world domain (see Figure 8.7) is more complex.

Table 8.2: Target localization time using ASP and POMDPs

Target localization time expressed as a fraction of the time taken by the proposed approach. Use of POMDP hierarchy enables the wheeled robot to identify targets reliably and efficiently. Sharing POMDP beliefs enables a team of humanoid robots to improve target localization time.

Search and Collaboration Strategies	Target localization time			
	Wheeled robot		Humanoid robots	
	Box in Fig. 8.8	Nao robot	Boxes	Balls
Random	–	–	1.93	1.64
Heuristic	1.47	1.44	1.2	1.03
Proposed	1	1	1	1

8.2.3 Experiments with Collaboration on Simulated Robots

Experiments have also been conducted on multiple robots in simulation. In each simulated trial, a grid map of a specific size was generated with the locations of the target object and the robot chosen randomly. When the belief in a grid cell exceeded 0.9, the grid cell was assumed to contain a target. Each data point in the following results is the average of 1000 simulated trials.

Assuming that all robots in a team move at the same speed, the average distance moved by robots in a team (in an episode/trial) was used as a measure of the team's performance—better collaboration will result in lower values of this measure. In each trial, robots and targets were placed randomly in a grid map, with no more than one robot or target in each grid-cell. A Gaussian bias (20%) was added to the initial belief in a 3×3 area around every target—the belief vector is then normalized. To simulate the unreliable communication, a *communication success rate* (CSR) parameter was introduced and set to 0.5, i.e., approximately every other broadcasted package was not received. Figure 8.10 shows the results for different combinations of robots and targets in a 15×15 grid map based on a real-world office scenario. The results show that the robots collaborate effectively to find the targets—similar results were obtained when experiments were conducted with maps of different sizes (4×4 to 25×25) with different amounts of initial belief and different values of CSR.

In real-world applications, it is common for the robot to have some prior knowledge of the likely location of the target object (e.g., a microwave is likely to be found in the kitchen). We therefore evaluated the ability of the collaboration approach to use prior knowledge in the form of an initial bias in the target locations. Figure 8.11 shows the performance of a team with two robots tasked with localizing two targets as a function of the bias in the

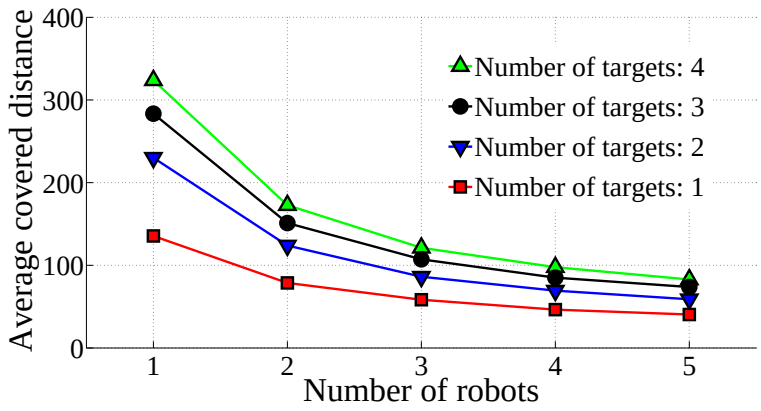

Figure 8.10: Robust multirobot collaboration in traveled distance needed

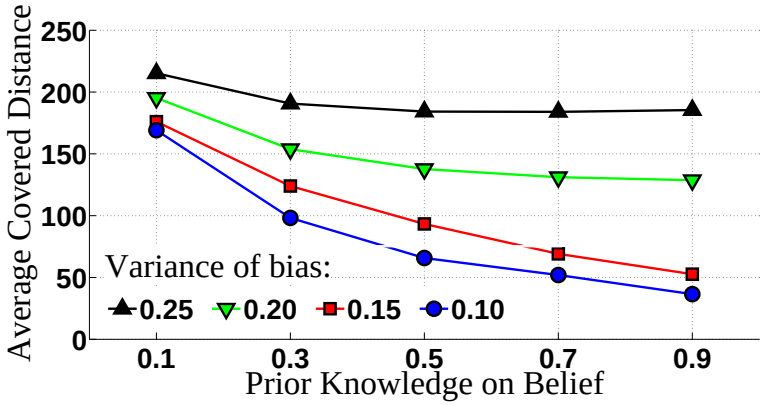

Figure 8.11: Performance improves if prior information is incorporated

initial belief. The robots are able to identify the targets faster as more information about target locations is made available or the information available about the targets' positions is more accurate (i.e., smaller variance of bias). Similar performance is observed for teams of different sizes searching for targets in domains of different sizes.

As stated in Chapter 7, communication between robots in the real-world is not reliable. The effect of communication uncertainty on multirobot collaboration was therefore evaluated next. The uncertainty in communication was simulated by changing the value of CSR.

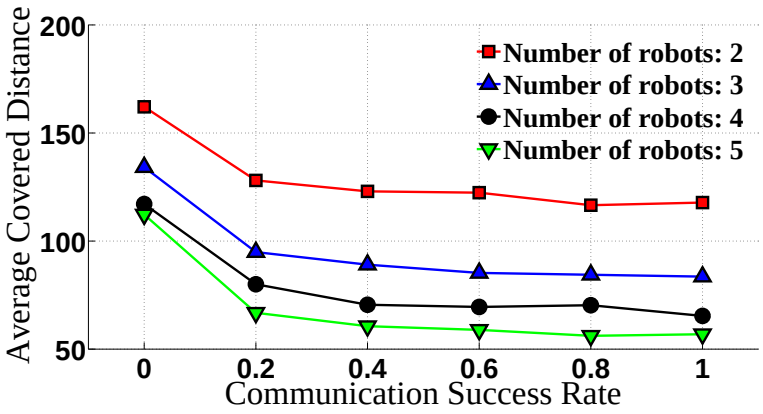

Figure 8.12: Performance is robust to dropped packages

Figure 8.12 shows results of experiments as a function of varying CSR, where robot teams were asked to locate two targets. Although a low likelihood of successful communication hurts the team's performance, results show that the time taken to localize targets stabilizes as CSR increases and is then no longer sensitive to the value of CSR. Similar performance was observed in experiments conducted with a different number of robots and targets.

Next, Table 8.3 summarizes target localization accuracy as a function of the (normalized) distance traveled by the robots, when two robots searched for targets in a 15×15 map. The initial positions of robots and targets were randomly assigned in each trial. The proposed approach (belief sharing with hierarchical POMDPs) was compared with: (a) random selection of actions and assignment of targets to robots (row labeled "random"); and (b) a heuristic policy which selects targets and actions based on the grid cell with the largest belief (row labeled "heuristic"). To simulate realistic scenarios, prior belief was assigned to multiple areas in the map (including the target location). The results show that belief sharing in hierarchical POMDPs results in the robots traveling a much smaller distance to detect targets with high accuracy. Over extensive simulation experiments (and robot trials,

Table 8.3: Target localization accuracy using hierarchical POMDPs

Target localization accuracy as a function of the (normalized) distance traveled by the robots. The proposed approach enables a team of robots to detect targets more accurately than random and heuristic search strategies.

Approach	Normalized covered distance			
	0.5	1.0	1.5	2.0
Random	0.033	0.171	0.382	0.537
Heuristic	0.079	0.334	0.549	0.817
Proposed	0.153	0.544	0.825	0.957

as described below) in different maps (3×3 to 25×25), using hierarchical POMDPs with the communication layer enables a team of robots to collaborate and localize target objects reliably and efficiently. There are some instances of more than one robot searching for the same target, but this overlap is intentional and occurs very infrequently.

Figure 8.13 is a pictorial representation of the POMDP approach for multirobot collaboration, when two robots repeatedly attempt to localize two targets in a 15×15 map with obstacles. The two figures show the number of times the robots visited each grid-cell. Intuitively, each robot trying to locate a target should first look around its starting position (in the absence of prior information of target location) and then explore other areas. Once the target is sighted, the robot should verify that the target is in the observed location. The actions taken by the robots are recorded over 100 simulated trials—a trial ends when the targets are located. In Figure 8.13, each grid cell's color changes from *blue* to *red* along the *vibgyor* spectrum based on repeated visits by the robot. Results shows that the obstacles are never visited and grid cells near the targets are visited more often than other grid cells. The radius of the yellow area reflects the largest distance of effective observation. In the absence of significant prior bias, the robots cannot move towards the targets until the first sighting of a target, which happens only when the robot is in a grid-cell near the cell with a

target. There is hence no clear path (in the figure) between the starting positions of robots and the target locations. Similar performance is observed for other simulated grid maps with different number of targets and robots. As stated earlier, there can be some instances of more than one robot searching for the same target, but this overlap is intentional and occurs very infrequently.

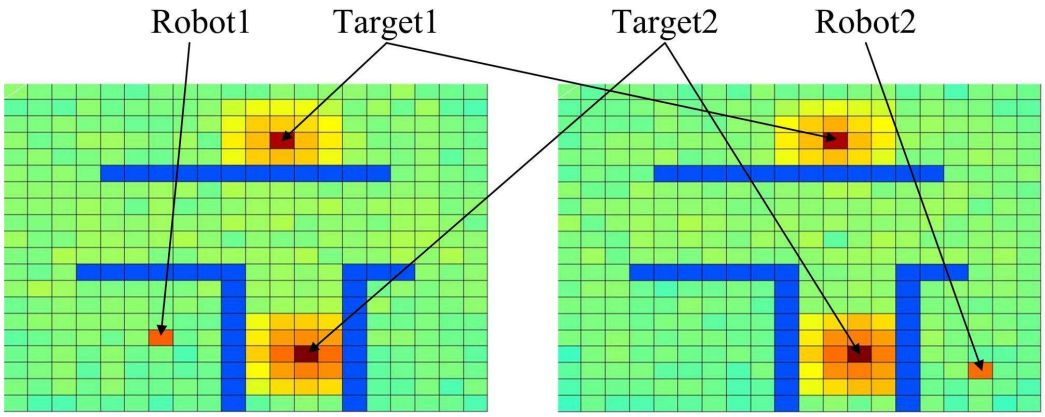

Figure 8.13: Numbers of times of cells visited in colors

Simulated trials with 2 robots and 2 targets. Obstacle locations are shown in *blue*, target locations in *dark red* and robot starting positions in *red*. Other cells show the number of times they were visited using colors ranging from *blue* to *red* along the visible spectrum.

8.2.4 Experiments with Multirobot Collaboration on Humanoid Robots

Multirobot collaboration experiments were conducted on humanoid Nao robots [65] as shown in Figure 8.14 because multiple wheeled robots were not available. The Nao is equipped with multiple monocular cameras that provide 640×480 images at $30Hz$, and ultrasound sensors for obstacle avoidance. Since stable navigation on different surfaces is a challenge on humanoids, experiments were conducted on an indoor ($4m \times 6m$) robot soccer field, which is typically used by a team of robots to play a competitive game of soccer. This moderately constrained domain captures the collaboration challenges we seek

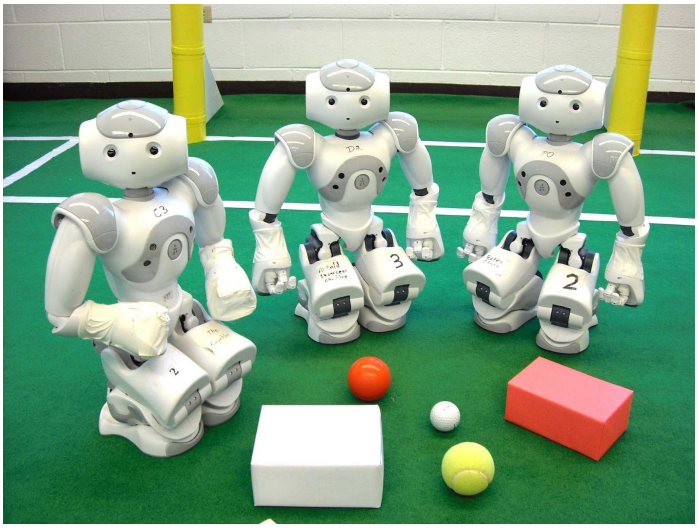

Figure 8.14: Humanoid robots

to address. Each robot has a domain map and localizes based on domain landmarks such as goals and field corners (with known map positions) detected in input images. All scenes in this domain are treated as uncluttered. Challenging scenarios were created by artificially introducing obstacles that the robot(s) had to walk around to see the desired targets (and landmarks). All computation (e.g., visual processing, localization and navigation) was performed on-board the robots using a $500MHz$ processor. The robots used UDP to broadcast packages to teammates to share information. The size of each grid cell in the map is $\approx 0.5m$.

Target objects include boxes and balls of different colors and shapes. Since objects are composed of homogeneous colors, gradient features are not used in the object models and visual processing operators consist of algorithms that detect the dominant color and shape in each ROI. Scene processing was modeled as a two-layered POMDP, with a POMDP that selects operators (i.e., algorithms) to apply on each salient ROI in an image, and a POMDP that controls the selection of image ROIs for processing. The transfer of control between

SP-POMDP and VS-POMDP is described in Section 5.3.

In all multirobot collaboration experiments, a team of $(1-4)$ Naos successfully localized one or more targets much faster than a heuristic collaboration strategy (manually) designed for this domain. The right half of Table 8.6 summarizes a subset of these experiments, where two humanoid robots localized two targets (different boxes and balls). Similar to the experiments on wheeled robots, target localization times with *heuristic* and *random* collaboration strategies are expressed as a multiple of the results with the *proposed* collaboration strategy—the average time taken by two Naos to localize two boxes is 1.01min. The target localization times are smaller (compared to experiments on wheeled robot) due to the collaborative effort and the relative simplicity of the domain (compared to Figure 8.7). Similar results are obtained for different combinations of robots and targets—the proposed strategy significantly reduces the target localization time in comparison to random and heuristic strategies. Delayed or lost packets (between robots) did not have a major effect on the target localization performance as long as the communication success ratio (CSR) was above a low threshold—the results reported in Table 8.6 correspond to a CSR of ≈ 0.5. Furthermore, the robots were able to deal with changes in team composition, e.g., adding a new robot or removing an existing team-member resulted in the smooth re-distribution of targets among robots. These experiments show that the POMDP hierarchy and belief sharing strategy enable one or more robots to adapt visual sensing and information processing to the task at hand, and collaborate robustly with teammates.

8.3 Experiments with ASP+POMDPs

Experimental trials were conducted in simulation and on wheeled robots visually identifying the locations of target objects in indoor domains. The following hypotheses were evaluated: (I) integrating ASP and POMDP enables reliable target localization while sig-

nificantly reducing target localization time in comparison with using ASP or POMDP individually; and (II) entropy-based strategy enables the robot to make best use of human feedback to localize targets.

A realistic simulated domain was designed to extensively evaluate the architecture, using learned object models and observation models to simulate motion and perception. Figure 8.15 shows an instance where four rooms are connected by a surrounding hallway in a 15×15 grid. Fifty stationary objects in 10 primary classes are simulated, and one or more of these objects are randomly selected as targets whose positions are unknown to the robot.a Table 8.4 shows the corresponding 50 objects in 10 different primary classes. The robot automatically creates the corresponding category tree from the KB. Each data point in the results described below is the average of 5000 simulated trials. In each trial, the robot's location, target object(s) and location(s) of target object(s) are chosen randomly. Unless stated otherwise, a trial ends when the belief in a grid cell exceeds a threshold (e.g., 0.90).

Table 8.4: Object classes for the simulated domain.

Classes	Root level	1									
	Internal level	1				1			1		
	Primary level	1	2	3	4	5	6	7	8	9	10
Object numbers	`room1`	3	2	3	1						
	`room2`				3	4	2	1			
	`room3`	1	1	7					6	1	2
	`room4`					2	3	2	1	3	2

8.3.1 Overall Performance using Trust Factors

Accuracy, localization time and the ratio of these values evaluated. The accuracy is maximum when reported position and ground truth position of an object are identical (e.g.,

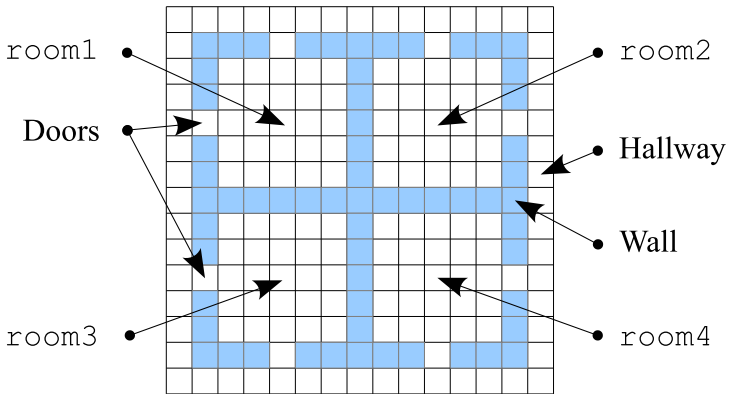

Figure 8.15: Simulated domain.

same grid cell), and drops off exponentially as the distance between reported position and ground truth position increases. Figures 8.16-8.18 summarize experimental results, with the x-axis depicting the extent to which ASP beliefs are trusted (Ω)—*all results in these figures are statistically significant*. Figure 8.16 shows that when ASP beliefs are not considered (0 along the x-axis), the accuracy is high (≈ 0.95) irrespective of the value of r (Equation 6.9). Even the few errors correspond to objects close to the edge of a grid cell being localized in one of the neighboring cells. However, the corresponding target localization time is large, as shown in Figure 8.17. As the robot starts considering ASP-based beliefs, i.e., Ω grows from 0 to 1, the target localization time decreases substantially. The effect of ASP-based beliefs on accuracy also depends on the value of r, e.g., a decrease in accuracy is observed very soon for $r = 0.05$ but not for $r = 0.2$. Target localization accuracy and time have different relative importance in different situations. The trade-off between these two measures is modeled by computing their ratio. Figure 8.18 displays the value of this third measure as a function of the value of Ω. We observe that irrespective of the value of r, the best accuracy-time balance occurs when the value of Ω (i.e., trust in ASP-based beliefs) is neither too high nor too low. We therefore conclude that combining

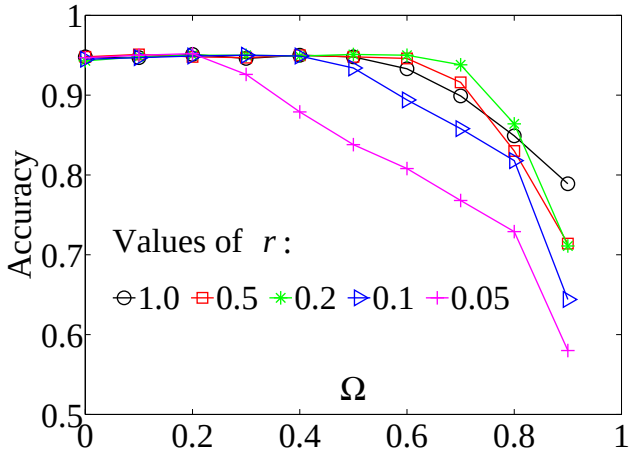

Figure 8.16: Performance measures for our architecture in accuracy.

answer sets and POMDP beliefs exploits their complementary properties, resulting in high accuracy while reducing the target localization time.

Some errors in the experimental trials are due to the incorrect organization of the classes (extracted from online repositories), and the robot not receiving sufficient observations to correct these KB errors. Another reason is that the evidence from "related" objects can sometimes overwhelm certain facts. For instance, when the `scanner` in `room2` is selected as the target in Figure 4.1, `room1` has the highest initial belief based on the answer set. It is a challenge for robots to recover from such situations if ASP-based beliefs are trusted substantially, especially when this trust is combined with false positive observations of target(s).

To investigate how the robot performs in individual trials, some data points from the plot corresponding to $r = 0.2$ in Figure 8.17 were selected and expanded into a set of cumulative density function (CDF) plots, as shown in Figure 8.19. A data point in Figure 8.19 should be interpreted as the robot completing a certain percentage of trials (values along y-axis),

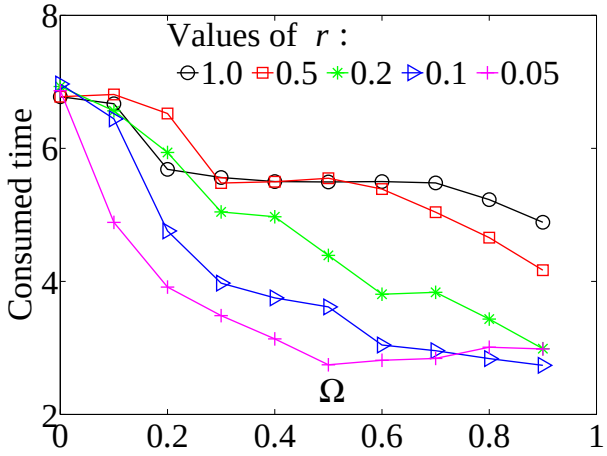

Figure 8.17: Performance measures for our architecture in time.

i.e., successfully localizing the target(s), within a certain amount of time, or equivalently, by traveling less than a certain distance (values along x-axis)—distance is measured in units of grid cells. The black line with $\Omega = 0.0$ corresponds to trials when the ASP KB is not included, i.e., the robot plans using just the POMDPs—in more than 60% of the trials, the robot localizes the desired target objects by traveling ≤ 100 grid cell units. When domain knowledge is included in the ASP KB and used for planning, the robot is able to successfully complete a larger percentage of trials within the corresponding limits on the distance traveled. All plots in Figure 8.19 corresponding to $\Omega \in [0.1, 0.9]$ also display an interesting (repeated) pattern of (almost) piecewise change in the robot's performance. For instance, the *cyan* plot shows that there are hardly any trials in which the distance traveled by the robot is in the range of $[40, 80]$ grid cell units. Each plot has more than one occurrence of this pattern (see "zoom-in area" in Figure 8.19. These piecewise changes can be explained as follows: if merging the ASP and POMDP beliefs does not provide a good initial location (e.g., room) for the robot to investigate, the robot may have to move a

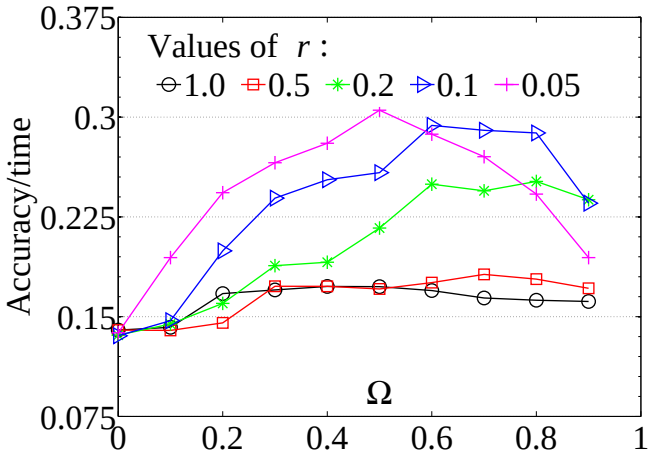

Figure 8.18: Performance measures for our architecture in accuracy-time ratio

considerable distance (e.g., to a different room) before it is likely to find the desired target. In addition, Figure 8.19 shows that as the amount of domain knowledge increases, the robot completes a larger % of trials by traveling a smaller distance. Furthermore, including a small amount of randomly selected knowledge (in the KB) can result in performance that is worse than considering no domain knowledge at all.

8.3.2 Reasoning with Different Amounts of Domain Knowledge

Once ASP identifies likely locations of target objects, POMDP beliefs can be used to focus the robot's sensing and navigation to identify the specific locations of objects in the rooms. Figure 8.20 summarizes the results of these experiments as a function of the amount of domain knowledge included in the KB and used to generate the prior beliefs—there is a time limit of 100 units in these trials. The trials corresponding to value= 0 on the x-axis represents the use of just POMDPs to localize objects. Combining prior beliefs extracted from answer sets with POMDP beliefs significantly increases the target localization accuracy (0.96) when all relevant domain knowledge (except the target) is provided;

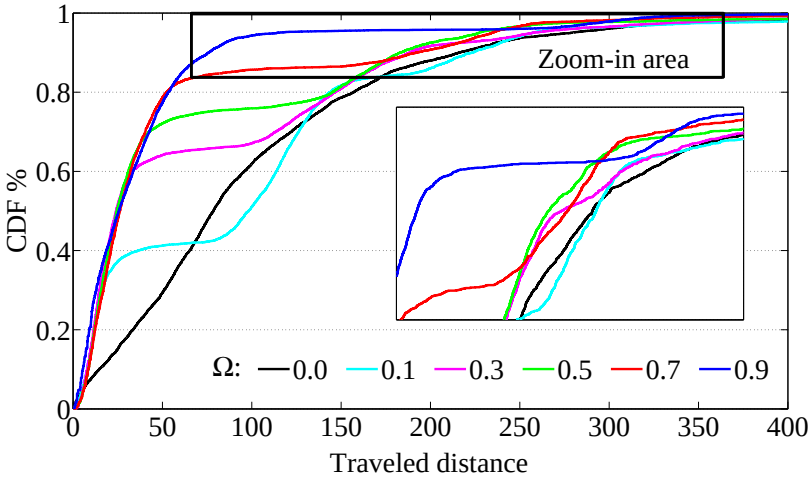

Figure 8.19: CDF plot of traveled distances in 5000 trials.

the errors correspond to trials where objects are at the edge of grid cells. The second data point in Figure 8.20 corresponds to a small amount of prior domain knowledge. Since this knowledge is selected randomly in the simulated trials, it can be incomplete and misleading. Robots can waste time exploring irrelevant locations, reducing localization accuracy especially when incorrect observations are made and/or the time provided to localize the target objects is exceeded. Even in such trials, the POMDP-based beliefs can help the robot recover from observation errors if the time limit is relaxed. As more knowledge is made available, we observe that the robot's localization accuracy improves quickly.

8.3.3 Comparison of Belief Merging Strategies

Next, we evaluated our approach of generating and merging beliefs. The KB is initialized with 20% domain knowledge. Periodically, information about a few randomly chosen objects is added into the KB to simulate learning from sensor inputs or human feedback. Inference in ASP produces new answer sets that are used to create new prior beliefs to

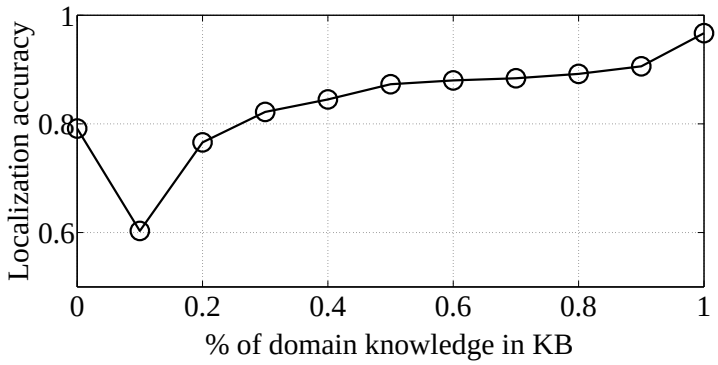

Figure 8.20: TL accuracy as a function of % of domain knowledge.
Combining ASP and POMDPs significantly increases the accuracy in comparison with just using POMDPs (left-most point on the curve).

be merged with the POMDP beliefs, thus guiding subsequent action selection. As stated earlier, the Bayesian belief merging is compared with three other strategies, and the results are summarized in Figure 8.21. The x-axis represents the localization error in units of grid cells in the simulated domain, while the y-axis represents % of trials where error is below a specific value. For instance, with Bayesian merging, more than 80% of the trials report an error of ≤ 5 units. This error is much smaller than our previous approach that used weighted averaging ("trust factors") to merge beliefs [105]. To compare with the Dirichlet-based weighting scheme, the individual beliefs were assigned weights based on the degree of correspondence with the Dirichlet distribution (i.e., beliefs extracted from answer sets may be assigned higher weights). These results indicate that Bayesian belief merging/revision performs better than other strategies (results are statistically significant). Similar results were observed with different levels of (initial) domain knowledge; assigning undue importance to ASP or POMDP beliefs can (in fact) hurt performance even when significant domain knowledge is available.

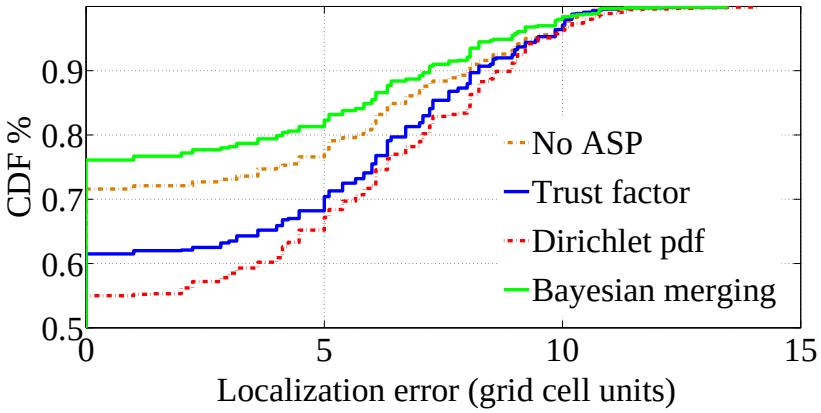

Figure 8.21: Comparison of belief merging strategies.
Using Dirichlet distribution for extracting prior beliefs from answer sets, and Bayesian belief merging result in lowest localization errors.

8.3.4 Evaluation of Entropy-based HRI Strategy

Human feedback is then considered in addition to sensor inputs. The simulator uses known ground truth to simulate human feedback that is *available* to the robot approximately once every five actions. In addition, there is a 20% likelihood of the feedback being incorrect. The results in Figure 8.22 are for the domain in Figure 8.15. Humans can help identify the room containing the target (*but not the exact location*) and comment on accessibility of rooms, as described in Section 6.4. The x-axis shows the belief entropy threshold above which the robot seeks human input. The three solid lines correspond to different costs associated with human feedback (in units of time). As a baseline for comparison, the three dashed lines (different colors correspond to different costs) represent the random acquisition of human feedback without considering the entropy. The trust factor for ASP is chosen in the range ($\approx 0.2 - 0.6$) that results in good performance in Figures 8.16–8.18 and r is 1. When the threshold equals the maximum entropy (≈ 5.4), the robot never asks for human feedback, whereas the robot always solicits human feedback (when available)

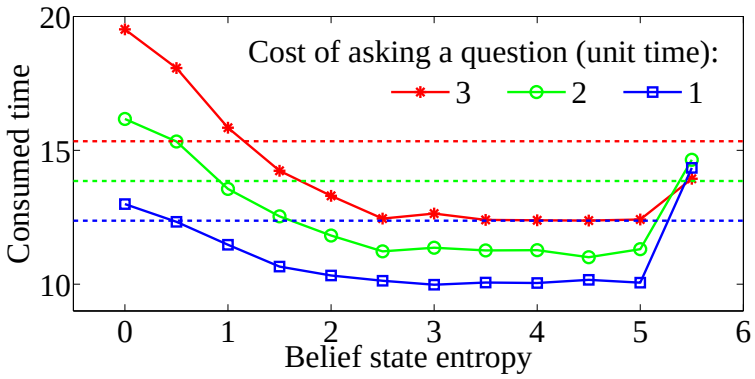

Figure 8.22: Target localization time with human feedback

when the threshold is 0. Since human feedback can be unreliable, acquiring and using a lot of human feedback increases target localization time. At the same time, if the robot rarely solicits human feedback (high entropy threshold), target localization takes more time. For any entropy threshold between $2.5 - 5.0$, time taken by the robot to localize targets is minimum. Human feedback thus helps significantly if used when needed. Furthermore, as cost of interacting with humans increases, feedback should be acquired more judiciously.

Another set of experiments were conducted by requiring the robot to localize target objects within a time limit, e.g., 100 time units. The robot uses ASP and POMDP beliefs (Bayesian merging) and starts with a fixed amount of domain knowledge in each trial. The robot can terminate the search if the POMDP beliefs converge, e.g., one of the cells has a belief larger than 0.8. Human appears with 0.5 probability after every action. When a human is detected, the robot evaluates the need for human feedback based on entropy of POMDP beliefs. Interacting with a human is modeled as taking twice as much time as (i.e., twice the cost of) a normal POMDP action. The (simulated) human responds to the robot's queries. The human may provide no useful information or even incorrect information; this is simulated by providing no answer or an incorrect answer periodically. Figure 8.23 sum-

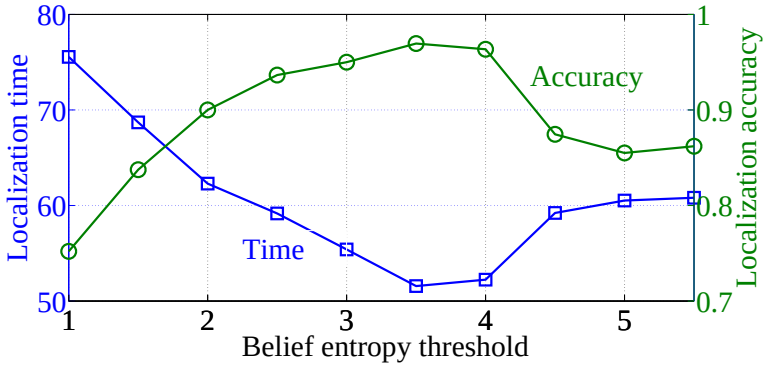

Figure 8.23: Entropy-based strategy to solicit human feedback.
Acquiring human feedback only when needed increases localization accuracy while reducing localization time.

marizes results of these experiments. When the entropy threshold is too small, the robot will always ask questions (when human is nearby), consuming a considerable amount of time in acquiring high-level information. This hurts the robot's ability to accurately localize objects within the available time. At the same time, when the entropy threshold is too high, the robot rarely asks for feedback, which also hurts localization accuracy. However, over a wide range of entropy values that represent true need for feedback (i.e., performance not very sensitive to choice of threshold), the robot localizes targets with high accuracy while minimizing localization time. Similar results were observed in trials that used different levels of (initial) domain knowledge.

8.3.5 Evaluation of Positive and Negative Observations

Experiments were then conducted to evaluate the modeling of positive and negative observations described in Section 6.3. The KB is fixed and target localization trials are conducted. In each trial, the target is randomly selected to be present or absent, i.e., the target does not exist in $\approx 50\%$ of the trials. A baseline policy (for comparison) is designed to

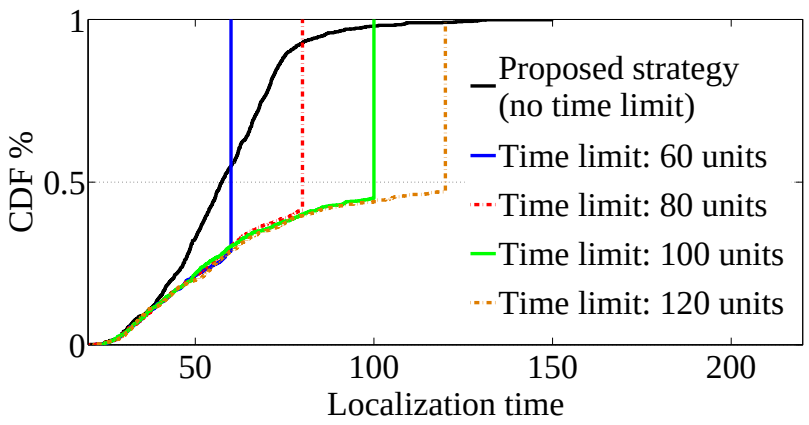

Figure 8.24: Using positive and negative observations.
Using positive and negative observations enables robot to terminate trials early when the target does not exist in the domain.

use ASP and POMDP beliefs (similar to experiments described above); this policy claims absence of target if it cannot be found within the time limit. Results are summarized in Figure 8.24 and Table 8.5. For instance, Figure 8.24 shows that exploiting positive and negative observations enables the robot to complete trials within 75 time units in 90% of the trials; trials are mostly completed much before the time limit especially when the target does not exist in the domain. However, when using the baseline policy with different time limits, the robot can never terminate when the target does not exist in the domain (obviously as predefined). Table 8.5 presents more results on the average time needed and accuracy. It shows that exploiting positive and negative observations results in much higher target localization accuracy while significantly lowering localization time compared with the baselines. To achieve a comparable accuracy to the proposed strategy, the baseline requires a large time limit (between 100 and 120). However, the proposed strategy using positive and negative observations requires an average time of 58.11 only that is roughly half of the baselines.

Table 8.5: Using positive and negative observations results in better performance.

	Proposed	Baseline (with time limit)			
		60	80	100	120
Time	58.11	54.94	67.51	79.05	90.42
Accuracy	0.952	0.795	0.896	0.949	0.972

8.3.6 Experiments on Physical Robots

Experiments were also conducted on physical robots operating on two floors of the Computer Science department at our University. The second floor, for instance, has three classrooms, a conference room, eight offices, a research lab, a kitchen and a common area—see Figure 8.25. The test platform was a wheeled robot (inset in Figure 8.25) equipped with cameras, range finder, microphones and on-board $2GHz$ processor. Algorithms were implemented on the robot using the Robot Operating System [79].

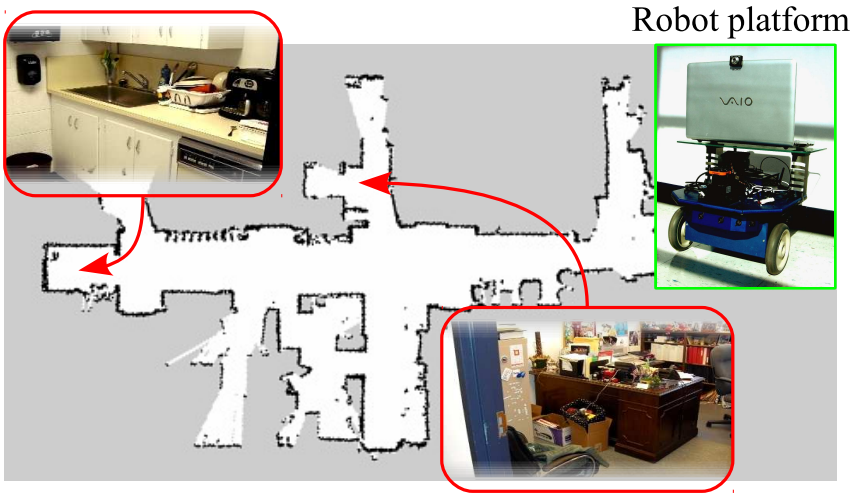

Figure 8.25: Domain map and wheeled robot platform

Figure 8.26 shows examples of target objects in this domain. Objects are characterized using visual features such as color and local image gradients. The robot uses our visual

learning algorithm to autonomously learn object models as a combination of models for these individual features [54]. Inputs from sensors and humans are processed to populate the KB. Plan execution in the lowest level of hierarchical POMDPs causes the robot to apply a sequence of actions, i.e., operators based on individual feature models in the learned object models, on input images, merging evidence to identify target objects.

Figure 8.26: Examples of target objects in domain

The robot starts with learned object models, learned domain map and some domain knowledge, which are revised incrementally. In all experimental trials, the robot successfully localized target objects in the appropriate positions. The results were similar to the simulated trials summarized in Figure 8.22 and Figure 8.18. In these trials, target localization times vary substantially depending on the initial positions of robot and targets. We therefore do not report the actual target localization times measured in the individual trials. However, using ASP-based beliefs and POMDP beliefs significantly reduces the target localization time by a factor of ≈ 0.6 (on average, with $\Omega = 0.4$) compared with just using POMDP beliefs. Trusting ASP beliefs a lot more than POMDP beliefs reduces localization accuracy—just using ASP beliefs results in trials where the robot does not find the targets even after a long period of time. Furthermore, judicious use of human feedback enables the

robot to interact with different humans and further reduce target localization time.

We summarize results of 30 experimental trials (each) with two specific target objects, a microwave oven and a humanoid—in each trial, the robot's starting location was chosen randomly. In a large subset of trials (25 out of 30), the targets were placed in expected locations, e.g., kitchen for microwave, and lab and office for humanoid—targets were placed in random locations in the remaining trials. The ground truth locations of the target objects are not provided to the robot, but it is equipped with learned object models, learned domain map and some domain knowledge. The proposed ASP+POMDP architecture was compared with two baseline strategies: (a) just using POMDP beliefs; and (b) using a heuristic policy that makes greedy action choices.

In all experimental trials, the robot successfully localized target objects. The results shown in Table 8.6 are similar to the simulated trials summarized in Figure 8.16-8.24. The actual target localization times vary substantially depending on the initial positions of robot and targets. We therefore report the target localization time of the baseline strategies as a factor of the target localization time of the proposed ASP+POMDP approach. Using ASP-based beliefs in conjunction with POMDP beliefs significantly reduces the target localization time. The target localization time with just POMDP beliefs is ≈ 1.6 times (averaged across different targets) that of the proposed architecture, while the factor is ≈ 2.4 for the heuristic policy. Similar to the results in simulation trials, trusting ASP beliefs a lot more than POMDP beliefs reduces the localization accuracy—just using ASP beliefs results in trials where the robot does not find the targets even after a long period of time. Furthermore, judicious use of human feedback enables the robot to interact with different humans and further reduce target localization time.

Consider a trial where the robot knows the presence of a refrigerator and a microwave

Table 8.6: Target localization time on physical robot platforms.

Target localization time expressed as a fraction of the time taken by the proposed approach.

Search strategies	Localization time for specific targets	
	Microwave	Humanoid
Heuristic	2.96	1.78
POMDP only	1.96	1.32
ASP+POMDP	1	1

in the "kitchen" and has to localize a coffee maker. Based on the object class tree of the current knowledge base, the robot concludes that the coffee maker is highly likely to occur in the same room with other kitchenware, resulting in high initial belief (of target occurrence) in the kitchen after merging the answer set-based bias distribution with the POMDP beliefs. As the robot moves to the kitchen, it meets a human but does not ask for input because the belief entropy is not high. In the main office outside the kitchen, the robot detects an HP printer that had recently been moved from the floor above, and the door to an instructor's office that was closed recently. These pieces of information, though not relevant to the current task, revise the KB for later use. When the robot reaches the kitchen, it processes images of different scenes and localizes the coffee maker. If the robot has to enter the instructor's office or find the (recently moved) HP printer in subsequent trials, it uses the existing knowledge to automatically generate suitable initial belief distributions and solicits human input appropriately. The video of an experimental trial is available online: `http://youtu.be/psCKXegot1c`

Figure 8.27 shows some screenshots of an experimental trial in an indoor office domain. The uses a learned map with some known semantic labels and the target object to be localized is the humanoid observed in the last row. Screenshots show external view of the robot, location of robot in learned domain map, and robot's view that lags slightly behind the

robot's actual location: (a) robot starts in learned domain map with some known semantic labels and plans path (*pink* curve) to first location ("main office"), trading off distance to be moved against likelihood of finding the target; (b) robot about to reach the first location; (c) robot processing a set of images in the specific scene; (d) belief update after target was not detected in this location; (e) robot plans path (*pink* curve) to next likely target location ("robot lab"); (f) robot avoids obstacle (human) on its way to the desired location; (g) robot analyzes images of the scene and obtains first positive observation of target; (h) robot moves closer to confirm the existence of target and accurately localize the target. Robot dynamically revises the learned map and periodically processes images as it moves between desired locations.The screenshots capture specific steps in the sequence of actions executed by the robot as it analyzes different images of a specific subset of scenes. Note that the robot dynamically revises the learned map and periodically processes images (at low resolution) as it moves between desired location. The video of such experimental trials can be viewed online:

- `http://youtu.be/EvY\_Jt-5BqM`
- `http://youtu.be/DqsR2qDayGQ`

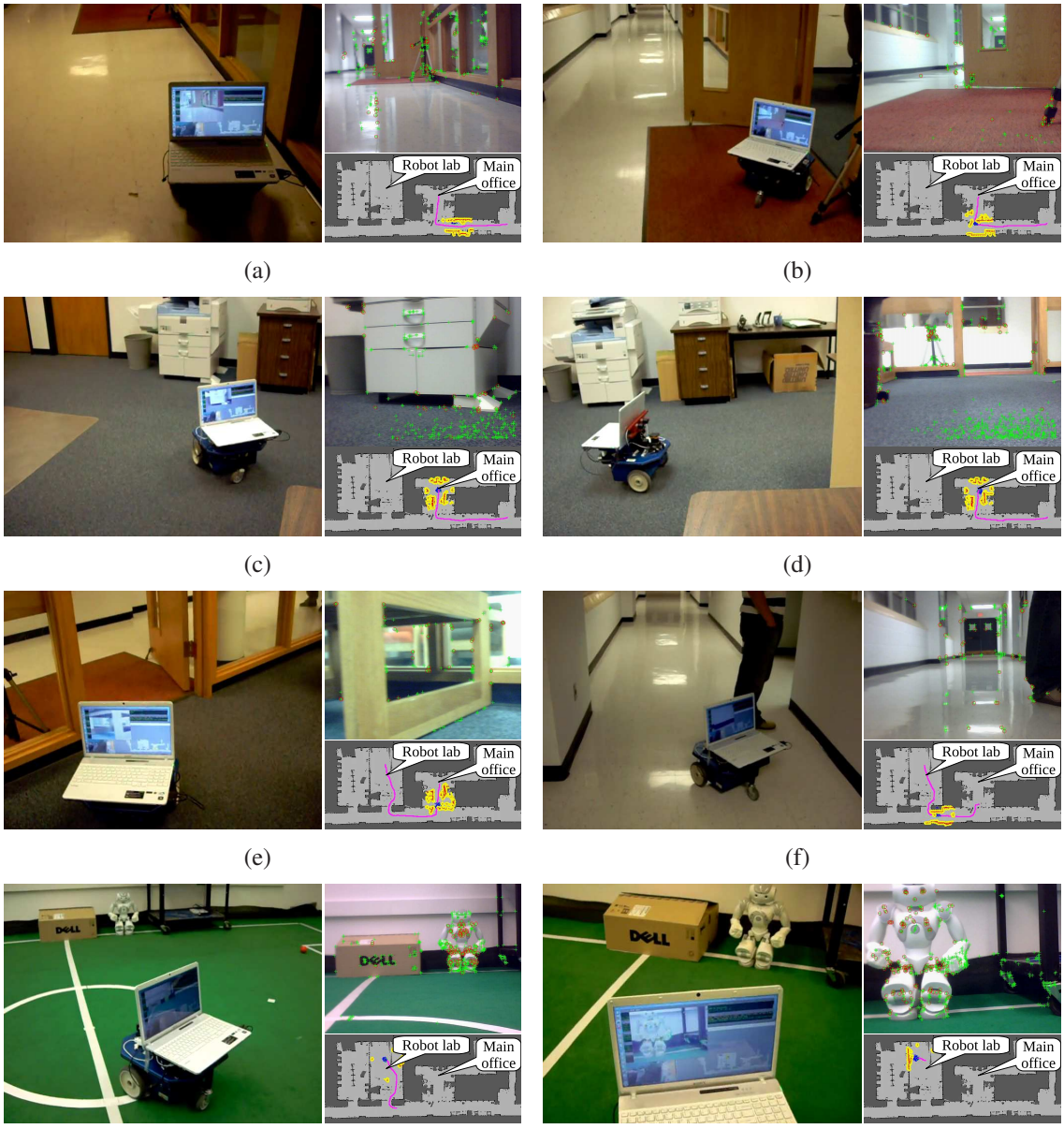

Figure 8.27: Screenshots of an experimental trial in which the target is a humanoid.

CHAPTER 9

CONCLUSIONS AND FUTURE WORK

A novel architecture is developed for knowledge representation and reasoning in robotics. This architecture integrates answer set programming and hierarchical POMDPs to enable a mobile robot to: represent, reason with and revise domain knowledge, automatically adapt sensing and information processing to the task at hand, merge non-monotonic logical inference with probabilistic beliefs, and acquire and use high-level human feedback when such feedback is available and necessary.

A hierarchical decomposition of POMDPs enables a mobile robot to automatically tailor visual sensing and information processing to each of a range of tasks at hand. Adaptive constrained convolutional policies and automatic belief propagation enables the robot to operate reliably and efficiently in complex indoor domains. Belief sharing between a team of robots is accomplished by augmenting the hierarchical POMDPs by a communication layer, enabling each robot to merge beliefs acquired by sensing the environment with the beliefs communicated by teammates. The robots are thus able to fully utilize the available information and collaborate robustly in simulated and real-world domains. Experimental results show that the architecture enables a robot to localize objects in complex indoor domains, making best use of domain knowledge, sensor inputs and human feedback.

Answer set programming enables a mobile robot to represent, reason with and revise domain knowledge obtained from sensor inputs and high-level human feedback. The prior beliefs extracted from relevant answer sets are represented by a Dirichlet distribution whose parameters are learned using a principled method. A few merging strategies are then compared including the arithmetic-geometric merging scheme and Bayesian updates. In par-

allel with the Dirichlet distribution, a set of Beta distributions are used to independently estimate whether each individual state is the underlying state by exploiting the positive and negative observations. Positive and negative observations enable a robot to identify situations in which assigned tasks can no longer be pursued. Grounding the architecture in a target localization domain shows that a mobile robot is able to use all available knowledge to reliably and efficiently localize target objects in an office domain.

The architecture opens many directions of future research, e.g., to explore a tighter coupling between logical inference and probabilistic planning for intelligent robots and agents, to investigate other algorithms for bias generation from answer sets, and consider other tasks such as information gathering and area coverage for evaluating the architecture. Another research direction is the choice of questions for human feedback to enable more realistic human-robot interaction. Regarding multirobot collaboration, one direction of further investigation is to explicitly model the sensing and actuation capabilities of different robots, and incorporate these learned models to improve the collaboration capabilities. Experiments will also be conducted using a larger number of robots and targets, and different types of robots. The ultimate goal is to enable widespread deployment of mobile robots that can interact and collaborate with humans in complex real-world domains.

LIST OF TABLES

8.1 Average number of steps to achieve accuracy of 0.95 82

8.2 Target localization time using ASP and POMDPs 86

8.3 Target localization accuracy using hierarchical POMDPs 90

8.4 Object classes for the simulated domain. 94

8.5 Using positive and negative observations results in better performance. . . . 105

8.6 Target localization time on physical robot platforms. 108

LIST OF FIGURES

1.1 Overview process diagram for autonomous robots. 1
3.1 Overview of the architecture. 26
4.1 Illustrative example of object classes stored in the knowledge base. 31
5.1 POMDP hierarchy and scenario 38
5.2 Motivation for convolutional policies. 49
5.3 Illustration of extracting 3×3 policy kernel from a 5×5 baseline policy . . 50
5.4 Illustration of using 3×3 policy kernel to generate 7×7 conv. policy. . . . 52
5.5 Hot-spot detection for motion planning. 56
6.1 Overview of the architecture for knowledge representation and reasoning. . 59
7.1 An overview of the POMDP hierarchy and scenario. 72
8.1 Simulated domain map. 78
8.2 Target localization accuracy using only ASP. 79
8.3 CC policies performance similarly to the baseline policy 80
8.4 Convolutional policies performs better than a ad-hoc search strategy 82
8.5 Accuracy comparison from simulated trials and directed re-weighting. . . . 83
8.6 Erratic robot 83
8.7 Domain map used in experiments. 84
8.8 Illustrative example of the use of BRIEF descriptor. 85
8.9 Connections between a relevant subset of nodes in the ROS implementation. 86
8.10 Robust multirobot collaboration in traveled distance needed 88
8.11 Performance improves if prior information is incorporated 88
8.12 Performance is robust to dropped packages 89

8.13 Numbers of times of cells visited in colors 91
8.14 Humanoid robots 92
8.15 Simulated domain. 95
8.16 Performance measures for our architecture in accuracy. 96
8.17 Performance measures for our architecture in time. 97
8.18 Performance measures for our architecture in accuracy-time ratio 98
8.19 CDF plot of traveled distances in 5000 trials. 99
8.20 TL accuracy as a function of % of domain knowledge. 100
8.21 Comparison of belief merging strategies. 101
8.22 Target localization time with human feedback 102
8.23 Entropy-based strategy to solicit human feedback. 103
8.24 Using positive and negative observations. 104
8.25 Domain map and wheeled robot platform 105
8.26 Examples of target objects in domain 106
8.27 Screenshots of an experimental trial in which the target is a humanoid. . . 110

BIBLIOGRAPHY

[1] D. Aberdeen, O. Buffet, and O. Thomas. Policy Gradients for PSRs and POMDPs. In *The International Conference on Artificial Intelligence and Statistics*, 2007.

[2] John R. Anderson, Daniel Bothell, Michael D. Byrne, Scott Douglass, Christian Lebiere, and Yulin Qin. An integrated theory of the mind. *PSYCHOLOGICAL REVIEW*, 111:1036–1060, 2004. Unlike Soar, a functionality oriented architecture, ACT-R aims at Psychological plausibility. Similar ones are CLARION and EPIC.

[3] Brenna Argall, Sonia Chernova, Manuela Veloso, and Brett Browning. A Survey of Robot Learning from Demonstration. *Robotics and Autonomous Systems*, 57(5):469–483, 2008.

[4] H. Asoh, Y. Motomura, F. Asano, I. Hara, S. Hayamizu, K. Itou, T. Kurita, T. Matsui, N. Vlassis, R. Bunschoten, and B. Krose. Jijo-2: an office robot that communicates and learns. *Intelligent Systems, IEEE*, 16(5):46–55, 2001.

[5] Alper Aydemir, Moritz Göbelbecker, Andrzej Pronobis, Kristoffer Sjöö, and Patric Jensfelt. Plan-based Object Search and Exploration Using Semantic Spatial Knowledge in the Real World. In *European Conference on Mobile Robots*, 2011.

[6] Alper Aydemir, Andrzej Pronobis, and Patric Jensfelt. Active visual object search in unknown environments using uncertain semantics. *Transactions in Robotics*, 2012. Extended version to the auhtors' ECMR paper.

[7] Chitta Baral. *Knowledge Representation, Reasoning and Declarative Problem Solving*. Cambridge University Press, 2003.

[8] Chitta Baral, Michael Gelfond, and Nelson Rushton. Probabilistic Reasoning with Answer Sets. *Theory and Practice of Logic Programming*, 9(1):57–144, 2009.

[9] A. M. Batog, A. Kjr-Nielsen, N. Kr, R. Petrick, C. Geib, N. Pugeault, M. Steedman, T. Asfour, R. Dillmann, B. Hommel, R. Detry, and J. Piater. Exploration and Planning in a Three-Level Cognitive Architecture. In *Cognitive Systems*, 2008.

[10] Richard Bellman. *Dynamic Programming*. Princeton University Press, Princeton, NJ, USA, 1 edition, 1957.

[11] Daniel S. Bernstein, Robert Givan, Neil Immerman, and Shlomo Zilberstein. The Complexity of Decentralized Control of Markov Decision Processes. *Mathematics of Operations Research*, 27(4), November 2002.

[12] Gerhard Brewka, Thomas Eiter, and Miroslaw Truszczynski. Answer set programming at a glance. *Commun. ACM*, 54(12):92–103, December 2011.

[13] O. Buffet and D. Aberdeen. The Factored Policy-Gradient Planner. *Artificial Intelligence*, 173(5-6):722–747, 2009.

[14] N. J. Butko and J. R. Movellan. I-POMDP: An Infomax Model of Eye Movement. In *International Conference on Development and Learning*, Monterey, USA, August 2008.

[15] M. Calonder, V. Lepetit, C. Strecha, and P. Fua. Brief: Binary robust independent elementary features. In *European Conference on Computer Vision*, 9 2010.

[16] J. Capitán, L. Merino, F. Caballero, and A. Ollero. Decentralized Delayed-State Information Filter (DDSIF): A new approach for cooperative decentralized tracking. *Robotics and Autonomous Systems*, 59(6):376–388, June 2011.

[17] J. Casper and R. R. Murphy. Human-robot interactions during urban search and rescue at the wtc. In *Transactions on SMC*, 2003.

[18] A. R. Cassandra. A Survey of POMDP Applications. In *AAAI Symposium: Planning with Partially Observable Markov Processes*, 1998.

[19] Laurent Charlin, Pascal Poupart, and Romy Shioda. Automated hierarchy discovery for planning in partially observable environments. In *Advances in Neural Information Processing Systems (NIPS)*, pages 225–232, 2006.

[20] Xiaoping Chen, Jianmin Ji, Jiehui Jiang, Guoqiang Jin, Feng Wang, and Jiongkun Xie. Developing High-Level Cognitive Functions for Service Robots. In *International Conference on Autonomous Agents and Multiagent Systems*, May 2010.

[21] Roger M. Cooke. *Experts in Uncertainty: Opinion and Subjective Probability in Science*. Oxford University Press, USA, 1991.

[22] G. Dissanayake, P. Newman, and S. Clark. A Solution to the Simultaneous Localization and Map Building (SLAM) Problem. *Transactions on Robotics and Automation*, 17(3):229–241, 2001.

[23] Amalia Foka and Panos Trahanias. Real-time hierarchical pomdps for autonomous robot navigation. *Robotics and Autonomous Systems*, 55(7):561–571, 2007.

[24] C. Galindo, J. A. Fernandez-Madrigal, J. Gonzalez, and A. Saffiotti. Robot task planning using semantic maps. *Robotics and Autonomous Systems*, 56(11):955–966, 2008.

[25] M. Gebser, R. Kaminski, B. Kaufmann, M. Ostrowski, T. Schaub, and M. Schneider. Potassco: The Potsdam Answer Set Solving Collection. *AI Communications*, 24(2):105–124, 2011.

[26] M. Gebser, R. Kaminski, A. König, and T. Schaub. Advances in *gringo* series 3. In *Proceedings of the Eleventh International Conference on Logic Programming and Nonmonotonic Reasoning (LPNMR'11)*, pages 345–351.

[27] M. Gebser, B. Kaufmann, A. Neumann, and T. Schaub. Conflict-Driven Answer Set Solving. In *International Joint Conference on Artificial Intelligence (IJCAI)*, 2007.

[28] Michael Gelfond. Answer Sets. In Frank van Harmelen and Vladimir Lifschitz and Bruce Porter, editor, *Handbook of Knowledge Representation*, pages 285–316. Elsevier Science, 2008.

[29] Michael Gelfond and Yulia Kahl. *Knowledge Representation, Reasoning and the Design of Intelligent Agents*. `http://www.cs.ttu.edu/~mgelfond/ai/book.pdf`, 2012.

[30] Michael Gelfond and Vladimir Lifschitz. The Stable Model Semantics for Logic Programming. In *International Conference on Logic Programming*, pages 1070–1080, 1988.

[31] Malik Ghallab, Dana Nau, and Paolo Traverso. *Automated Planning: Theory and Practice*. Morgan Kaufmann, San Francisco, USA, 2004.

[32] Moritz Göbelbecker, Charles Gretton, and Richard Dearden. A Switching Planner for Combined Task and Observation Planning. In *National Conference on Artificial Intelligence (AAAI)*, 2011.

[33] J. Golińska-Pilarek and E. Muñoz Velasco. Reasoning with qualitative velocity: Towards a hybrid approach. In *Proceedings of the 7th International Conference on Hybrid Artificial Intelligent Systems - Volume Part I*, HAIS'12, pages 635–646, Berlin, Heidelberg, 2012. Springer-Verlag.

[34] Michael A. Goodrich and Alan C. Schultz. Human-Robot Interaction: A Survey. *Foundations and Trends in Human-Computer Interaction*, 1(3):203–275, 2007.

[35] Piotr J. Gymtrasiewicz and Prashant Doshi. A Framework for Sequential Planning in Multi-Agent Settings. *Journal of Artificial Intelligence Research*, 24:49–79, 2005.

[36] Joseph Y Halpern. Reasoning about uncertainty. 2003.

[37] Marc Hanheide, Charles Gretton, Richard Dearden, Nick Hawes, Jeremy Wyatt, Andrzej Pronobis, Alper Aydemir, Moritz Gobelbecker, and Hendrik Zender. Exploiting Probabilistic Knowledge under Uncertain Sensing for Efficient Robot Behaviour. In *International Joint Conference on Artificial Intelligence (IJCAI)*, 2011.

[38] Nick Hawes, JeremyL. Wyatt, Mohan Sridharan, Henrik Jacobsson, Richard Dearden, Aaron Sloman, and Geert-Jan Kruijff. Architecture and representations. In HenrikIskov Christensen, Geert-JanM. Kruijff, and JeremyL. Wyatt, editors, *Cognitive Systems*, volume 8 of *Cognitive Systems Monographs*, pages 51–93. Springer Berlin Heidelberg, 2010.

[39] Jesse Hoey, Pascal Poupart, Axel Bertoldi, Tammy Craig, Craig Boutilier, and Alex Mihailidis. Automated Handwashing Assistance for Persons with Dementia using Video and a Partially Observable Markov Decision Process. *Computer Vision and Image Understanding*, 114(5):503–519, 2010.

[40] Kaijen Hsiao, Leslie Pack Kaelbling, and Tomas Lozano-Perez. Grasping POMDPs. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2007.

[41] iRobot Corporation. Autonomous Home Vacuum Cleaner: Roomba, 2012. `http://www.irobot.com/us/`.

[42] Leslie Kaelbling, Michael Littman, and Anthony Cassandra. Planning and Acting in Partially Observable Stochastic Domains. *Artificial Intelligence*, 101:99–134, 1998.

[43] Leslie Kaelbling and Tomas Lozano-Perez. Domain and Plan Representation for Task and Motion Planning in Uncertain Domains. In *IROS Knowledge Representation for Autonomous Robots Workshop*, 2011.

[44] Keith S. Jones and Elizabeth A. Schmidlin. Human-Robot Interaction: Toward Usable Personal Robots. In *Reviews of Human Factors and Ergonomics*, pages 100–148. Human Factors and Ergonomics Society, 2011.

[45] Kristian Kersting and Luc De Raedt. Bayesian logic programs. In *Proceedings of the Work-in-Progress Track at the 10th International Conference on Inductive Logic Programming*. Citeseer, 2000.

[46] David E. Kieras and David E. Meyer. An overview of the epic architecture for cognition and performance with application to human-computer interaction. *Human-Computer Interaction*, 12(4):391–438, 1997.

[47] Hiroaki Kitano, Minoru Asada, Yasuo Kuniyoshi, Itsuki Noda, and Eiichi Osawa. Robocup: The robot world cup initiative. In *Proceedings of the first international*

conference on Autonomous agents, AGENTS '97, pages 340–347, New York, NY, USA, 1997. ACM.

[48] W. Bradley Knox and Peter Stone. Combining manual feedback with subsequent MDP reward signals for reinforcement learning. In *International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, May 2010.

[49] JunYoung Kwak, Rong Yang, Zhengyu Yin, Matthew Taylor, and Milind Tambe. Teamwork and Coordination under Model Uncertainty in DEC-POMDPs. In *The AAAI Workshop on Interactive Decision Theory and Game Theory*, 2010.

[50] John E Laird. *The Soar cognitive architecture*. MIT Press, 2012. Similar ones (function oriented) are Companions, ICARUS and LIDA.

[51] John E. Laird, Allen Newell, and Paul S. Rosenbloom. Soar: An architecture for general intelligence. *Artificial Intelligence*, 33(1):1 – 64, 1987. First ever publication of SOAR.

[52] Lei Li and Stuart J. Russell. The blog language reference. Technical Report UCB/EECS-2013-51, EECS Department, University of California, Berkeley, May 2013.

[53] Xiang Li and Mohan Sridharan. Safe Navigation on a Mobile Robot using Local and Temporal Visual Cues. In *International Conference on Intelligent Autonomous Systems*, 2010.

[54] Xiang Li and Mohan Sridharan. Vision-based Autonomous Learning of Object Models on a Mobile Robot. In *AAMAS-2012 Workshop on Autonomous Robots and Multirobot Systems*, Valencia, Spain, 2012.

[55] Xiang Li, Mohan Sridharan, and Catie Meador. Autonomous Learning of Visual Object Models on a Robot Using Context and Appearance Cues. In *International Conference on Autonomous Agents and Multiagent Systems*, May 2013.

[56] Xiang Li, Mohan Sridharan, and Shiqi Zhang. Autonomous Learning of Vision-based Layered Object Models on Mobile Robots. In *International Conference on Robotics and Automation*, Shanghai, China, May 9-13 2011.

[57] D. Lowe. Distinctive Image Features from Scale-Invariant Keypoints. *IJCV*, 60(2):91–110, 2004.

[58] David G Lowe. Object recognition from local scale-invariant features. *International Conference on Computer Vision, 1999*, pages 1150–1157, 1999.

[59] Alexei Makarenko, Alex Brooks, Stefan Williams, and Hugh Durrant-whyte. A decentralized architecture for active sensor networks. In *Proc. of Int. Conf. on Robotics and Automation (ICRA)*, 2004.

[60] Jiri Matas, Ondrej Chum, Martin Urban, and Tomás Pajdla. Robust wide-baseline stereo from maximally stable extremal regions. *Image and vision computing*, 22(10):761–767, 2004.

[61] Brian Milch, Bhaskara Marthi, Stuart Russell, David Sontag, Daniel L Ong, and Andrey Kolobov. Blog: Probabilistic models with unknown objects. *Statistical relational learning*, page 373, 2007.

[62] Stephen Muggleton. Stochastic logic programs. *Advances in inductive logic programming*, 32:254–264, 1996.

[63] R.R. Murphy. A decade of rescue robots. In *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*, pages 5448–5449, 2012.

[64] J. Najemnik and W. S. Geisler. Optimal Eye Movement Strategies in Visual Search. *Nature*, 434:387–391, March 2005.

[65] Nao. The Aldebaran Nao Robots, 2008. `http://www.aldebaran-robotics.com/`.

[66] Nils J Nilsson. Probabilistic logic. *Artificial intelligence*, 28(1):71–87, 1986. probably the first probabilistic reasoning publication.

[67] James R Norris. *Markov chains*. Number 2008. Cambridge university press, 1998.

[68] Jose Nunez-Varela, B. Ravindran, and Jeremy L. Wyatt. Where do i look now? gaze allocation during visually guided manipulation. In *International Conference on Robotics and Automation (ICRA)*, pages 4444–4449, 2012.

[69] S. C. Ong, S. W. Png, D. Hsu, and W. S. Lee. Planning Under Uncertainty for Robotic Tasks with Mixed Observability. *International Journal of Robotics Research*, 29(8):1053–1068, July 2010.

[70] Liviu Panait and Sean Luke. Cooperative Multi-Agent Learning: The State of the Art. *Autonomous Agents and Multiagent Systems*, 11(3):387–434, November 2005.

[71] Sebastien Paquet. *Distributed decision-making and task coordination in dynamic, uncertain and real-time multiagent environments*. PhD thesis, Université Laval, 2005.

[72] L. E. Parker. Distributed Intelligence: Overview of the Field and its Application in Multi-Robot Systems. *Journal of Physical Agents (invited article) Special Issue on Multi-Robot Systems*, 2(2):5–14, 2008.

[73] Hanna Pasula and Stuart Russell. Approximate inference for first-order probabilistic languages. In *IJCAI*, volume 1, pages 741–748. Citeseer, 2001.

[74] J. Pineau, M. Montemerlo, M. Pollack, N. Roy, and S. Thrun. Towards Robotic Assistants in Nursing Homes: Challenges and Results. In *RAS Special Issue on Socially Interactive Robots*, 2003.

[75] Joelle Pineau, Geoff Gordon, and Sebastian Thrun. Point-based Value Iteration: An Anytime Algorithm for POMDPs. In *The International Joint Conference on Artificial Intelligence*, 2003.

[76] David Poole. Probabilistic horn abduction and bayesian networks. *Artificial intelligence*, 64(1):81–129, 1993.

[77] David Poole. The independent choice logic for modelling multiple agents under uncertainty. *Artificial Intelligence*, 94(1):7–56, 1997.

[78] David Poole. Abducing through negation as failure: Stable models within the independent choice logic. *The Journal of Logic Programming*, 44(1):5–35, 2000.

[79] Morgan Quigley, Ken Conley, Brian P. Gerkey, Josh Faust, Tully Foote, Jeremy Leibs, Rob Wheeler, and Andrew Y. Ng. ROS: An Open-Source Robot Operating System. In *Open Source Software Workshop*, 2009.

[80] Matthew Richardson and Pedro Domingos. Markov logic networks. *Machine Learning*, 62(1-2):107–136, 2006.

[81] Stephanie Rosenthal and Manuela Veloso. Mobile Robot Planning to Seek Help with Spatially Situated Tasks. In *National Conference on Artificial Intelligence*, Toronto, Canada, July 2012.

[82] Stephanie Rosenthal, Manuela Veloso, and Anind Dey. Learning Accuracy and Availability of Humans who Help Mobile Robots. In *National Conference on Artificial Intelligence*, San Francisco, 2011.

[83] Stéphane Ross, Joelle Pineau, Sébastien Paquet, and Brahim Chaib-draa. Online planning algorithms for pomdps. *Journal of Artificial Intelligence Research*, 32(1):663–704, July 2008.

[84] P. E. Rybski, A. Larson, H. Veeraraghavan, M. LaPoint, and M. Gini. Communication strategies in Multi-Robot Search and Retrieval: Experiences with MinDART. In *International Symposium on Distributed Autonomous Robotic Systems*, 2004.

[85] Taisuke Sato and Yoshitaka Kameya. Prism: a language for symbolic-statistical modeling. In *IJCAI*, volume 97, pages 1330–1339. Citeseer, 1997.

[86] Sven Seuken and Shlomo Zilberstein. Improved Memory-Bounded Dynamic Programming for Decentralized POMDPs. In *Twenty-Third International Conference on Uncertainty in Artificial Intelligence (UAI)*, Vancouver, Canada, July 2007.

[87] Guy Shani, Joelle Pineau, and Robert Kaplow. A survey of point-based pomdp solvers. *Autonomous Agents and Multi-Agent Systems*, 27(1):1–51, 2013.

[88] Mohan Sridharan. Augmented Reinforcement Learning for Interaction with Non-Expert Humans in Agent Domains. In *International Conference on Machine Learning Applications (ICMLA)*, 2011.

[89] Mohan Sridharan, Jeremy Wyatt, and Richard Dearden. Planning to See: A Hierarchical Aprroach to Planning Visual Actions on a Robot using POMDPs. *Artificial Intelligence*, 174:704–725, July 2010.

[90] Mohan Sridharan, Jeremy L. Wyatt, and Richard Dearden. HiPPo: Hierarchical POMDPs for Planning Information Processing and Sensing Actions on a Robot. In *International Conference on Automated Planning and Scheduling (ICAPS)*, pages 346–354, 2008.

[91] Ron Sun. The clarion cognitive architecture: Extending cognitive modeling to social simulation. *Cognition and multi-agent interaction*, pages 79–99, 2006.

[92] R. L. Sutton and A. G. Barto. *Reinforcement Learning: An Introduction*. MIT Press, Cambridge, MA, USA, 1998.

[93] R. S. Sutton, D. McAllester, and S. Singh. Policy Gradient Methods for Reinforcement Learning with Function Approximation. In *Advances in Neural Information Processing Systems (NIPS)*, 2000.

[94] G. Theocharous, K. Rohanimanesh, and S. Maharlevan. Learning hierarchical observable markov decision process models for robot navigation. In *IEEE International Conference on Robotics and Automation (ICRA)*, volume 1, pages 511–516 vol.1, 2001.

[95] Georgios Theocharous, Kevin Murphy, and Leslie Pack Kaelbling. Representing hierarchical pomdps as dbns for multi-scale robot localization. In *IEEE International Conference on Robotics and Automation (ICRA)*, volume 1, pages 1045–1051 Vol.1, 2004.

[96] S. Thrun. Stanley: The Robot that Won the DARPA Grand Challenge. *Journal of Field Robotics*, 23(9):661–692, 2006.

[97] Marc Toussaint, Laurent Charlin, and Pascal Poupart. Hierarchical pomdp controller optimization by likelihood maximization, 2008.

[98] Karthik Varadarajan and Marcus Vincze. Ontological Knowledge Management Framework for Grasping and Manipulation. In *IROS Workshop on Knowledge Representation for Autonomous Robots*, 2011.

[99] Joost Vennekens, Sofie Verbaeten, and Maurice Bruynooghe. Logic programs with annotated disjunctions. In *Logic Programming*, pages 431–445. Springer, 2004.

[100] Videre-Design. Videre Design Robot and Sensors, 2010. `http://www.videredesign.com/index.php?id=21`.

[101] Willie Walker, Paul Lamere, Philip Kwok, Bhiksha Raj, Rita Singh, Evandro Gouvea, Peter Wolf, and Joe Woelfel. Sphinx-4: A Flexible Open Source Framework for Speech Recognition. Technical report, Sun Microsystems SMLI TR-2004-139, November 2004.

[102] The International POMDP Practitioners Workshop. The international pomdp practitioners workshop, 2010. `http://users.isr.ist.utl.pt/~mtjspaan/POMDPPractioners/`.

[103] Steve Young, Milica Gasic, Blaise Thomson, and Jason D. Williams. Pomdp-based statistical spoken dialog systems: A review. *Proceedings of the IEEE*, 101(5):1160–1179, 2013.

[104] Shiqi Zhang and Mohan Sridharan. Active Visual Sensing and Collaboration on Mobile Robots using Hierarchical POMDPs. In *International Conference on Autonomous Agents and Multiagent Systems*, 2012.

[105] Shiqi Zhang, Mohan Sridharan, and Forrest Sheng Bao. ASP+POMDP: Integrating Non-monotonic Logical Reasoning and Probabilistic Planning on Robots. In *International Conference on Development and Learning and Epigenetic Robotics*, 2012.

[106] Shiqi Zhang, Mohan Sridharan, and Xiang Li. To Look or Not to Look: A Hierarchical Representation for Visual Planning on Mobile Robots. In *International Conference on Robotics and Automation (ICRA)*, 2011.

[107] Shiqi Zhang, Mohan Sridharan, and Christian Washington. Active visual planning for mobile robot teams using hierarchical pomdps. In *Robotics, IEEE Transactions on*, volume PP, pages 1–1, 2013.

[108] Weijun Zhu. *PLOG: Its Algorithms and Applications*. PhD thesis, Texas Tech University, USA, May 2012.