KNOWLEDGE-BASED ROBOT SEQUENTIAL DECISION-MAKING UNDER PARTIAL OBSERVABILITY

BY

SAEID AMIRI

B*Sc*, Sharif University of Technology, 2013
M*Sc*, University of Houston, 2015

DISSERTATION

Submitted in partial fulfillment of the requirements for
the degree of Ph.D. in Computer Science
in the Graduate School of
Binghamton University
State University of New York
2022

Accepted in partial fulfillment of the requirements for
the degree of Ph.D. in Computer Science
in the Graduate School of
Binghamton University
State University of New York
2022

April 2022

Shiqi Zhang, Chair
Department of Computer Science, SUNY Binghamton

Lei Yu, Member
Department of Computer Science, SUNY Binghamton

Sujoy Sikdar, Member
Department of Computer Science, SUNY Binghamton

Christopher Amato, Member
Department of Computer Science, Northeastern University

Hiroki Sayama, Outside Examiner
Department of Systems Science and Industrial Engineering, SUNY Binghamton

1 ABSTRACT

Robots frequently face complex tasks that require more than one action, where Sequential Decision Making (SDM) capabilities become necessary. SDM is challenging because actions and sensor measurements of robots are imperfect, and the world state can be changed by exogenous factors that are out of the robot's control. One of the frameworks that have been used for robot SDM is Partially Observable Markov Decision Processes (POMDPs) for actively collecting information and accomplishing long-term goals under partial observability. However, POMDPs are not good at leveraging various forms of commonsense knowledge in their decision-making. Given all the advances in SDM research, there is the need of developing frameworks that can learn or reason with human knowledge to plan toward achieving long-term goals. The main contributions of this dissertation are robot SDM frameworks that leverage various forms of human knowledge to plan under uncertainty (e.g., partial observability) toward achieving long-term goals. In this dissertation, we study how learning from labeled datasets and reasoning based on commonsense knowledge can guide agents to do active information collection under uncertainty in partially observable domains. We apply our frameworks on tasks including object property retrieval using sensors of different modality, augmenting robots' knowledge in spoken dialogue systems, estimating human intention in human-robot interaction tasks, and searching for target objects. The experimental

evaluations conducted on various problems in this dissertation, demonstrate that leveraging various forms of knowledge or learning improves robot planning by reducing the information collection cost and increasing success rate in the evaluated tasks.

ACKNOWLEDGEMENTS

I am deeply indebted to numerous individuals without whose assistance this dissertation could not have been written. First and foremost, I would like to thank my supervisor, Prof. Shiqi Zhang, for his consistent support and guidance during my Ph.D. study. Furthermore, I would like to thank Prof. Yu, Prof. Sikdar, and Prof. Amato for accepting to be my committee members and for their constructive feedback. I thank Prof. Sayama for serving as the outside examiner in my committee. My research progress would not have been possible without the assistance of the aforementioned people. My gratitude extends to other researchers in our lab, including Kishan Chandan, Yohei Hayamizu, Yan Ding, Xiaohan Zhang, and David Defazio. Our research collaborations and discussions created a rewarding environment for making contributions. A special word of thanks is owed to my family, whose encouragement, moral support, and advice were extremely helpful.

Contents

List of Tables

List of Figures

List of Abbreviations

CORPP COmmonsense Reasoning and Probabilistic Planning 58

RNNs Recurrent Neural Networks 14

SARP Scene Analysis for Robot Planning viii

2 INTRODUCTION

Researchers have developed autonomous intelligent mobile robots that can provide services to people and interact with them. Such robots have been able to operate in everyday environments over extended periods of time, and travel long distances that have been impossible before, while providing services, such as escorting, guidance, and delivery [1, 2, 3, 4, 5]. Such robots require Sequential Decision Making (SDM) capabilities in order to accomplish complex tasks that cannot be executed successfully using only one single action. SDM plays a key role toward robot long-term autonomy, because real-world domains are stochastic, and a robot must repeatedly estimate the current world state and decide what to do next. Assuming domains with deterministic action outcomes, SDM can be done using classical planning [6] methods which generate a sequence of actions, enabling the agent to reach the goal state. Such systems are usually backed by plan monitoring methods. However, in practice, it frequently happens that the robot's action effects are uncertain due to noisy actuation or unexpected changes in the environment. It is important to note that in robotic systems, some components of the state may not be fully observable. For example, a mobile robot might be able to sense its orientation (e.g., using a compass sensor), but cannot tell its exact (x, y) coordinates. The robot would need to maintain a belief distribution over all possible states. Therefore, robots need to sequentially make decisions based

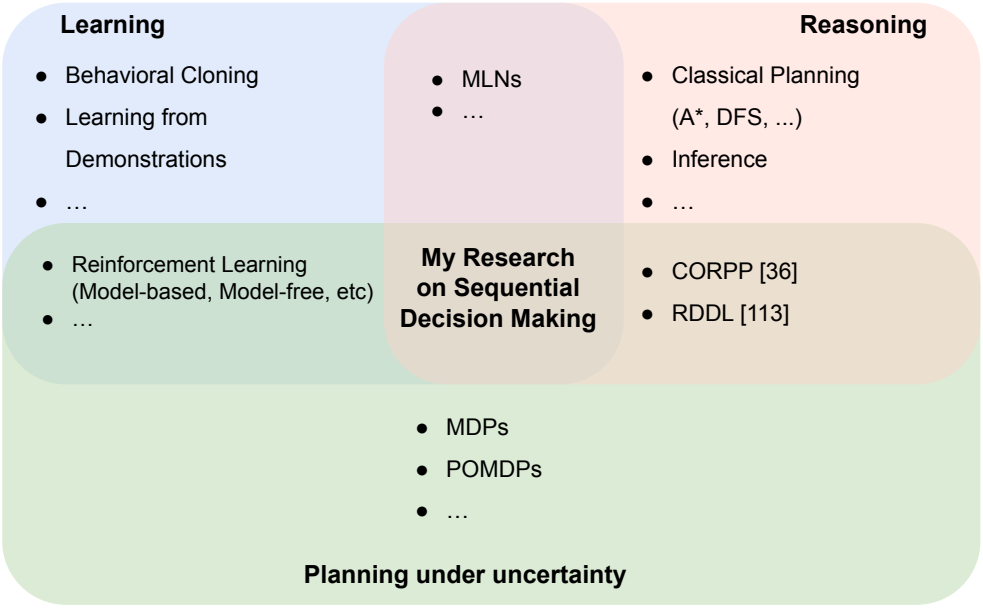

Figure 2.1: There are various ways to do sequential decision-making, but none of them individually meet the needs in the robotics context, including learning from past experience, reasoning with contextual knowledge, and planning to achieve long-term goals.

on the current state (belief) of the world. An agent faces various challenges from knowledge-based SDM such as the ones arising from multi-sensory perception, and limited, inaccurate knowledge. My research aims to address those challenges from the problem of knoweldge-based SDM under partial observability. We will discuss each challenge, and provide details of our approaches in the following chapters.

Next, we discuss some existing SDM methods, and then we discuss the challenges of SDM that have motivated our research.

2.1 Sequential Decision Making Methods

There are at least three AI paradigms, namely *supervised learning*, *automated reasoning*, and *planning under uncertainty*, that can be used for robot SDM, as illustrated in Figure 2.1. First, a robot can leverage supervised learning methods to make decisions using the previously collected (multisensory) data, e.g.,

by Learning from Demonstration (LfD) of people or other agents [7]. Second, Knowledge Representation and Reasoning (KRR) methods can be used for decision making [8]. Those methods include classical planning, inference and diagnosis, and can be logic-based, or probability-based. Third, planning under uncertainty[1] methods support active information collection for goal achievement, e.g., using decision-theoretic frameworks such as Markov Decision Processes (MDPs) [9] and Partially Observable Markov Decision Processes (POMDPs) [10].

In order for robots that operate in human-inhabited environments to have long-term autonomy, their SDM capabilities should at least support applications such as interaction with humans and identifying their intention and communicating using natural language. In fact, we use the above mentioned applications as test cases for our experimental evaluations. All the mentioned paradigms have been studied extensively in the literature, but there are no existing systematic integrations of those paradigms. Next, we discuss what our research theme is.

2.2 Research Theme

Each of the three SDM paradigms, namely supervised learning, automated reasoning, and planning under uncertainty has a long history with rich literature. Given all the advancements in the mentioned SDM methods, still there are challenges that need to be addressed:

1. **SDM with multisensory perception:** When robots execute tasks, they constantly have to perceive the environment for state estimation. Robots are often equipped with sensors of different modality such as audio, vision,

[1]Here we consider the broader definition of planning under uncertainty that also includes reinforcement learning.

and proprioception. One challenge for such robots is to incorporate the information from various modalities toward state (belief) estimation. Learning multisensory object representations through manipulation has enabled robots to improve their object recognition skills as well as language grounding by representing predicates that describe the object. For instance, a robot can *look* to figure out if an object is "*red*" using computer vision methods. However, vision is not sufficient to answer if an opaque bottle is "*full*" or not, and behaviors that support other sensory modalities, such as *lift* and *shake*, become necessary. Given the sensing capabilities of robots and the perceivable properties of objects, it is important to develop algorithms to enable robots to use multimodal exploratory behaviors to identify object properties.

In Chapter 5, we investigate this problem where service robots receive tasks from humans in which they have to identify object properties in a query. Understanding object properties require robots not only to learn what objects look like but also to discover how they feel, sound, and move as a result of the robot's interaction with them. We developed SDM frameworks to solve the Multi-modal Predicate Identification (MPI) problem, where a robot selects actions for multi-modal perception in object exploration tasks under action and perception uncertainty. Our approach could dynamically construct a planning model given an object description from a human user (e.g., "*a blue heavy bottle*"), compute a high-quality policy for this model, and use the policy to guide robot behaviors (such as "look" and "shake") toward maximizing information gain. The dynamically built controllers en-

abled the robot to focus on a minimum set of domain variables that are relevant to the current object and query. The planning perception models were learned using two existing datasets collected with robots interacting with objects in the real world. This contribution primarily focused on a robot exploring objects in a tabletop scenario. We present the details of this work in Chapter 5.

2. **SDM with limited knowledge:** Robots need a Knowledge Base (KB) in order to reason about and plan high-level tasks. However, the KB may not be comprehensive most of the times, which results in the failure of the task completion or suboptimal solutions. Another challenge for robot SDM is how robots can reason about their knowledge deficiency and understand when it is necessary to augment their knowledge. We investigate this research problem on robot spoken dialogue systems in Chapter 6. SDM is necessary for spoken dialog systems as the robot needs to decide what to utter at each turn given all the challenges and uncertainties from language understanding and generation. In this contribution, we introduced a dialog agent that simultaneously supports human intention identification and knowledge augmentation through multi-turn dialog. We developed a dual-track POMDP controller that enabled the agent to conduct dialog and knowledge management at the same time. The knowledge management controller reasoned over the need of augmenting the knowledge base. We present the details of this work in Chapter 6.

3. **Leveraging human knowledge and past experiences for SDM:** Another challenge for robot SDM is to leverage the complementary features

of different existing SDM paradigms. For instance, supervised learning methods can help an agent decide what to do based on labels of precollected data, and automated reasoning methods are good at following rules to make decisions. In this dissertation, we consider the paradigms of data-driven supervised learning and automated common sense reasoning, as well as how they can be combined with planning under uncertainty methods for SDM under partial observability. We investigate this challenge in Chapter 7 where a robot needs to identify human intentions to find out if they are interested to interact with the robot or not. In particular, we utilized a labeled dataset of human trajectories, some commonsense knowledge to reason with, and robots' predefined skills. In this contribution, we developed a robot SDM framework that integrates supervised learning for passive state estimation, automated reasoning for incorporating declarative contextual knowledge, and planning under uncertainty for active perception and task completions. Compared to the previously mentioned contributions, this contribution leverages both reasoning and learning to guide the POMDPs agent for SDM under partial observability.

4. **Avoiding the curse of dimensionality:** Constructing SDM frameworks requires that the robot has a complete world model, which tends to be infeasible in practice and strains the limits of SDM solvers. In particular, real-world environments frequently include many objects, making it troublesome to use any of the SDM frameworks to have a universal representation of all objects. Also, the complexity of reasoning about these objects and their relationships grows exponentially as more objects are considered. The

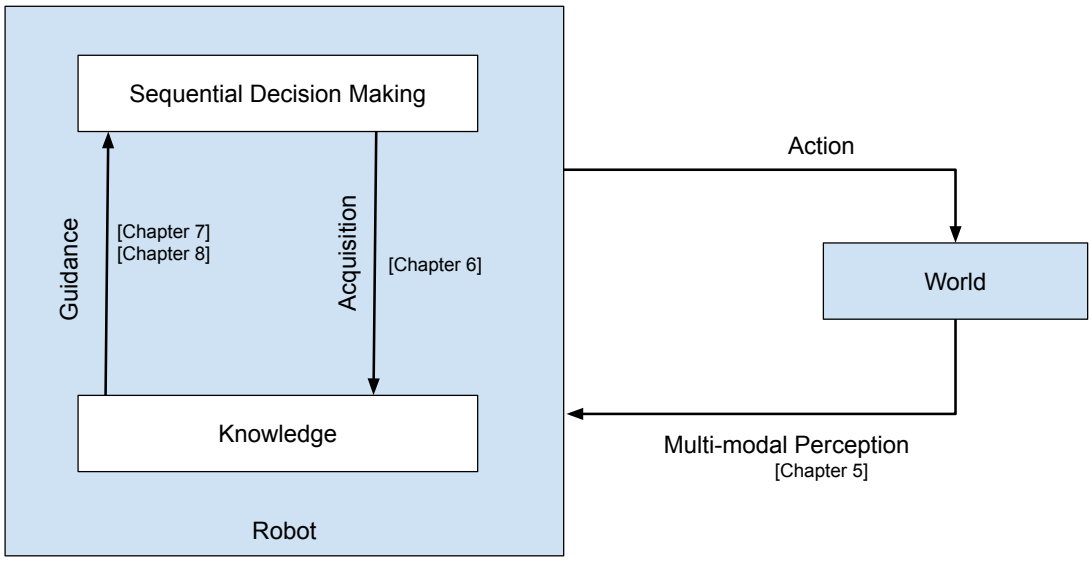

Figure 2.2: The illustration of how each of our contributions improves the interconnection of SDM, knowledge, and the environment

problem that we are concerned with is how to represent and reason with the knowledge about many objects and their relationships, and how to leverage the representation and reasoning to guide robot planning. In Chapter 8, we develop an approach that reasons with contextual information for scene analysis to enable POMDP-based robot planning. Our method called SARP is used for object search tasks while avoiding modeling all the objects inside the task planners.

2.3 Dissertation Organization

In Chapter 3, we provide background information about SDM paradigms and then in Chapter 4, discuss the works that are related to my dissertation. Afterwards, we explain each of my contributions in Chapters 5, 6, 7, and 8 . Finally, we briefly summarize the contributions in this dissertation in Chapter 9 and the future work in Chapter 10. Figure 2.2 presents how each of our contributions

improves knowledge-based SDM under partial observability.

3 BACKGROUND

In this section, we provide the background information about this dissertation. The information is about some SDM frameworks including Planning under uncertainty, Answer Set Programming (ASP), MDPs and POMDPs, as well as, information about the robotic platforms used in our research experiments.

3.1 Planning under Uncertainty

When world models are provided, agents can use planning methods to reach a goal state from an initial state. Unlike classical planning methods [6] that output a sequence of actions, planning under uncertainty methods generate a policy, which is a mapping from the current state (belief state) to an action toward maximizing long-term utilities. Planning under uncertainty can be formalized using MDPs and POMDPs:

3.1.1 Markov Decision Processes

MDP is a tuple (S, A, T, R, γ) where S is the state-space, A is the action space, T is the transition function, R is the reward function, and γ is the discount factor. One can use value iteration methods to calculate the state (V values) and state-action values (Q values) to find the policy π.

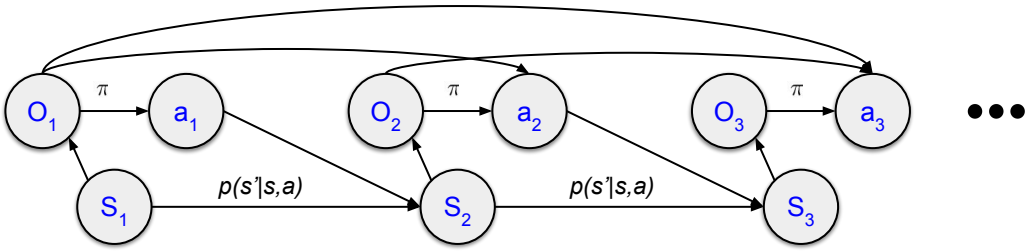

Figure 3.1: A dynamic Bayesian network (DBN) of POMDPs [15].

3.1.2 Partially Observable Markov Decision Processes

Partially observable MDPs (aka POMDPs) [10] generalize MDPs by assuming partial observability of the current state. A POMDP model is represented as a tuple $(S, A, T, R, Z, O, \gamma)$ where S is the state-space, A is the action set, T is the state-transition function, R is the reward function, O is the observation function, Z is the observation set and γ is the discount factor that determines the planning horizon (3.1). An agent maintains a belief state distribution b with observations $(z \in Z)$ using the Bayes update rule:

$$b'(s') = \frac{O(s', a, z) \sum_{s \in S} T(s, a, s') b(s)}{pr(z|a, b)}$$

where s is the state, a is the action, $pr(z|a, b)$ is a normalizer, and z is an observation.

One can use the available off-the-shelf solvers [11, 12, 13, 14] to build and compute a policy.

3.2 Knowledge Representation and Reasoning

Intelligent reasoning is some variety of formal calculation, typically deduction using first-order logic rules [16]. Representations, function as surrogates for

abstract notions such as actions, processes, beliefs, causality, and categories, allowing them to be described inside an entity, so that it can reason about them [16]. Knowledge Representation and Reasoning (KRR) is a capability that robots can use for SDM. It includes classical planning [6], inference, and diagnostic, and can be logic-based or probability-based. Graphs are another form of knowledge representation in the form of binary relations connecting objects.

Next, we describe ASP, P-log, and scene graphs which are the tools used for KRR:

3.2.1 Answer Set Programming

ASP is a declarative non-monotonic KRR formalism [17, 18, 19]. ASP supports recursive definitions, defaults, and causal relations. Its syntax is defined in terms of *atoms*, *literals*, and *rules*. An atom is an elementary proposition, such as l, while a literal is an atom with its negation, such as l and $-l$. Literal can also be the default negation of an atom represented by *not*. For instance, *not* l means it is not believed that l is true. A rule consists of *head* and *body*. Head is true when the body is true. A rule is expressed as:

$$l_0, \cdots, l_k :- l_{k+1}, \cdots, l_m, not\ l_{m+1}, \cdots, not\ l_n$$

where l_i is a literal. A rule without the head is a *constraint*, and a rule without the body is a *fact*. Solving an ASP program results in a number of answer sets.

3.2.2 P-log

P-log extends ASP by allowing probabilistic rules for quantitative reasoning. A P-log program consists of the logical and probabilistic parts. The logical part inherits the syntax and semantics of ASP. The probabilistic part contains *pr-atoms* in the form of:

$$pr_r(G(\eta) = w|B) = v$$

where $G(\eta)$ is a random variable, B is a set of literals and $v \in [0, 1]$. The pr-atom states that, if B holds and experiment r is fixed, the probability of $G(\eta) = w$ is v.

3.2.3 Scene Graphs

Scene graph G is a representation of the semantic content of an image, consisting of a set of bounding boxes $B = \{b_1, ..., b_n\}$, a set of objects $V = \{o_0, o_1, \cdots\}$, and a set of binary predicates $E = \{r_0, r_1, \cdots\}$ [20], where $b_i \in \mathbb{R}^{1\times4}$, and assigning a class label $o_i \in V$ to each b_i. A triplet of object-predicate-object is called a relationship. Given an image I, the probability distribution of the scene graph $pr(G|I)$ is decomposed into three components:

$$pr(G|I) = pr(B|I)pr(V|B, I)pr(E|V, B, I)$$

where the bounding box component $pr(B|I)$ generates a set of candidate regions from the input image, $pr(V|B, I)$ predicts the class label for each predicted region, and $pr(E|V, B, I)$ predicts the predicate among objects, which is conditioned on the predicted labels.

We will use scene graphs in Chapter 8 to guide the robot POMDP-based

planner.

3.3 Supervised Learning

Given a labeled dataset of size m, consisting of features x_i and labels y_i where i is the index of ith instance, supervised learning methods aim at estimating $\hat{y}$ that minimizes the cost function J_θ where θ is the model parameters vector. For instance, one choice for the cost function would be $J_\theta = \sum_1^m \frac{1}{m}(\hat{y} - y)^2$. Next, we describe two of the neural network types of supervised learning.

3.3.1 Convolutional Neural Networks

A Convolutional Neural Network (CNN) comprises convolutional layers followed by fully connected layers, as in a standard multilayer neural network. The basic building blocks of CNN consist of convolutional, pooling, activation, and fully-connected layers. In a convolutional layer, a filter is passed over an image, viewing a few pixels at a time. The convolution operation is a dot product of the original pixel values with weights defined in the filter. Pooling layers are used for downsampling, and fully-connected layers output a list of probabilities for different possible labels. The activation layers introduce non-linearity. The architecture of a CNN is designed to take advantage of the 2D structure of an input image (or other 2D input such as a speech signal). This is achieved with local connections and tied weights followed by some form of pooling which results in translation-invariant features. A benefit of CNNs is that they are easier to train and have many fewer parameters than fully connected networks with the same number of hidden units.

3.3.2 Long Short-Term Memory

Recurrent Neural Networks (RNNs) [21] are a kind of neural networks that use their internal state (memory) to process sequences of inputs. Given the input sequence vector $(x_0, x_1, ..., x_n)$, at each time-step, a hidden state is produced that results in the hidden sequence of $(h_0, h_1, ..., h_n)$. Long Short-Term Memory (LSTM) [22] network is a type of RNN that includes LSTM units. Each memory unit in the LSTM hidden layer has three gates for maintaining the unit state: input gate defines what information is added to the memory unit; output gate specifies what information is used as output, and forget gate defines what information can be removed (Eq. 3.3.2). LSTMs use memory cells to resolve the problem of vanishing gradients, and are widely used in problems that require the use of long-term contextual information, e.g., speech recognition [23] and caption generation [24].

$$\begin{cases} i_t = \sigma(w_i[h_{t-1}, x_t] + b_i) \\ f_t = \sigma(w_f[h_{t-1}, x_t] + b_f) \\ o_t = \sigma(w_f[h_{t-1}, x_t] + b_o) \end{cases}$$

where x_t refers to the input at current timestep and h_{t-1} is the output of the previous hidden unit. i_t, f_t, and o_t denote to input, forget, and output gates, respectively. The terms of w_t, w_f, and w_o represent weight, and b_t, b_f, and b_o represent biases.

Figure 3.2: Segway platform used in our research contributions.

3.4 Robot Platform

We use a Segway-based mobile robot platform, RMP110, for experimental trials in our research (Figure 3.2). The platform is equipped with a SICK laser sensor for mapping, localization, and navigation. Astra Orbbec RGB-D camera is used for human detection and interaction, this camera is used in Chapter 7 to detect humans using point cloud data. In addition to this monocular camera, the robot has a 360-vision camera for panaromic view of its environment. We use this camera in Chapter 8. The robot's software runs on Robot Operating System (ROS) [25]. Our implementation is established on the Building Wide Intelligence codebase that is available to the public on Github [3], where we use functionalities, such as task planning to help robot compute a sequence of actions given the initial and goal conditions. These actions include navigating to different rooms, opening the doors, and going through the doors.

4 RELATED WORKS

In this chapter, we discuss some of the previous works that leveraged various forms of knowledge for SDM or used supervised learning methods for SDM. A survey paper has summarized research on knowledge-based sequential decision making [26]. Here, we briefly discuss a few of such papers and categorize them based on the type of knowledge they leveraged in their work.

4.1 Action Knowledge for SDM

People have developed algorithms using action knowledge to guide planning and Reinforcement Learning (RL) agents. methods [27, 28, 29, 30, 31, 32]. Among them, Leonetti, Iocchi, and Stone 2016 used a declarative action knowledge to help an agent select only the reasonable actions in RL exploration. Yang et al. 2018 developed an algorithm called PEORL that integrates hierarchical RL with task planning, and this idea was applied to deep RL by Lyu et al. 2019. Another work leverages RL to incrementally learn new unknown logical rules to update the robot's knowledge base [30]. Recent research has shown that action knowledge can be used to guide a model-based RL agent by providing an optimistic, artificial interaction experience to speed up the learning process [33, 34]. Others have used action knowledge to build a hierarchical robot planner where the higher level computes a sequence of abstract actions and the lower level implements the

higher-level actions using primitive behaviors [35].

4.2 Rule-based Commonsense Knowledge to Guide SDM

Researchers have used rule-based commonsense knowledge to guide the robot planning under uncertainty [36, 37, 38, 39, 40, 41, 42, 35, 43, 44]. An example is an algorithm that uses first-order logic to construct decision tree policies for goal-oriented factored POMDPs [45]. Algorithm iCORPP enables a robot to reason with contextual knowledge to compute parameters of a planning agent's reward and transition functions [46]. Others have used commonsense knowledge to guide a classical planner to reason and plan in open worlds [47, 48].

Hybrid reasoning has been used to reason about world dynamics [46], enabling planners to generate behaviors that are adaptive to dynamic world dynamics. [35] developed a refinement-based architecture, where declarative knowledge is used for task planning and reasoning tasks, such as diagnosis and history explanation, and high-level deterministic plans are implemented via probabilistic planners. Also, researchers have studied non-stationary Markov decision processes in which policies can be updated when there is a change in the world states [43]. There is a work that has used human-provided declarative information to improve robot planning under uncertainty [38]. Researchers introduced a framework in which the agent learns world state transitions via model-based RL, and then incorporates them into the logical-probabilistic reasoning module [44]. Also, the work by Lu et al. incorporates multi-modal perception and commonsense reasoning on a POMDP-based dialog manager to better find human intentions through engaging in a dialogue [37].

4.3 Hierarchical Frameworks for Knowledge-based SDM

Researchers have been using hierarchies to construct their framework's knowledge-base or planner [35, 49]. In recent work, researchers used a visual hierarchical planning algorithm for long-horizon manipulation tasks. Their framework integrates neuro-symbolic task planning and graph-based motion generation on graph-based scene representations [49]. Their method has two-level abstractions of a manipulation scene, with geometric scene graphs and symbolic scene graphs. To enable a robot to operate in open-world domains, researchers have developed a three-layer hierarchy for reasoning about action knowledge and default knowledge [41]. In particular, the knowledge at a higher level was used for correcting lower-level knowledge to guide probabilistic planning.

4.4 Graph-based Human Knowledge for SDM

Graph-based representations have been used for reasoning with contextual information to guide robot planning [49, 50, 51]. For instance, researchers have used Conditional Random Fields [52] to construct contextual knowledge bases for robot target search by maintaining a belief over the locations of target objects and landmark objects while exploiting the knowledge of their co-appearances [50].

Other methods were developed to extract prior knowledge from data, e.g., using LSTM networks [22], to guide a planning agent in navigating new environments [51].

4.5 Knowledge-based RL

When the world model is unavailable, SDM can be realized using RL. Some works studied how human knowledge can be used to improve the performance of reinforcement learning (RL) agents [28, 53, 54, 55, 56, 57]. For instance, existing research has shown that an RL agent is able to reason with action knowledge to decompose complex tasks into smaller, tractable subtasks [28]. The concept of reward machines has been introduced for using temporal knowledge toward reusing interaction experience and creating extra feedback for learning purposes [53].

Researchers have developed algorithms to integrate model-free RL and automated planning to avoid taking unreasonable actions in exploration. Algorithm DARLING is perhaps the earliest work that leverages action preconditions and effects from human knowledge for RL agents to avoid visiting the risky or useless state, and has been applied to mobile robot navigation, and the grid world domains [34]. Researchers have integrated automated planning and Q-learning, focusing on non-stationary domains under uncertainties [43]. Those algorithms exploited the flexibility of RL approaches and the accuracy of declarative knowledge from humans. Other algorithms use action knowledge to improve model-free RL agents' performance in exploratory behaviors [58, 59].

Planning methods have been used to guide the higher level of hierarchical RL methods [53, 28, 29, 54, 60, 61]. In those methods, the agents use an action language to compute plans to decompose a complex task into a sequence of subtasks, and each subtask is then implemented by a reinforcement learner. For instance, the work of Jiang et al. (2019) showed that the introduction of a few milestone positions at the task level can improve mobile robots' performance in indoor navi-

gation tasks. The work of Icarte et al. (2018) built reward machines using temporal knowledge from domain experts to guide RL agents' learning behaviors, and also the reward machine can be learned from trial-and-error experiences [62].

The domain knowledge used in those works significantly improved the learning efficiency of RL agents. However, designing the hierarchy is frequently difficult, and many of the hierarchical methods trade optimality for learning efficiency.

Researchers have developed various algorithms for incorporating human supervision into SDM tasks [63, 64]. Among them, Taylor and Borealis surveyed a few ways of improving robots' SDM capabilities with supervision (in the form of demonstration or feedback) from people.

In another work, researchers developed a Refinement-based Acting Engine (RAE), called UPOM that is capable of learning heuristics for their planning tree search [65]. In a similar wok, Wells et al. [64] learns a classifier that finds the feasibility of a motion planner in task and motion planning.

In another work, researchers proposed the algorithm Deep Q-learning from Demonstrations (DQfD) that used a small set of demonstration data, to help an RL agent [66]. DQfD initially pretrains solely on the demonstration data using a combination of temporal difference (TD) and supervised losses. The supervised loss enables the algorithm to learn to imitate the demonstrator, while the TD loss enables it to learn a self-consistent value function from which it can continue learning with RL.

In recent years, [67] used hand-crafted POMDPs to generate data for imitation learning for vision-based grasping. Their method can learn a policy using a small amount of data.

4.6 Cognitive Architectures for SDM

Inspired by human's sequential decision making capabilities, researchers have been developing computational frameworks that can mimic human cognitive functionalities at some level. Such efforts have resulted in the development of cognitive architectures such as Prodigy [68], Icarus [69], ACT-R [70], and Soar [71]. Some of these architectures are concerned primarily with explaining human cognition, while others emphasize the construction of intelligent agents.

A cognitive architecture usually consists of: 1) memories for storing knowledge 2) processing units that extract, select, combine, and store knowledge, and 3) languages for representing the knowledge that is stored and processed. However, most of these architectures rely on complicated designs, while our research goal is to develop a general framework that has less intricate design towards better generalized applications.

5 SDM WITH MULTIMODAL PERCEPTION

In this chapter, we discuss how object-centric interactive perception [72] can be modeled as sequential decision-making (SDM) problems, and how learning from multi sensory datasets assists the robot to perform SDM under partial observability.

5.1 Introduction

Service robots are increasingly present in everyday environments, such as homes, offices, airports, and hospitals, where a common task is to retrieve an object for a user. Consider the request, "*Please fetch me the red, empty bottle.*" A key problem for the robot is to decide whether a particular candidate object matches the properties (or *predicate*) in the request, which we refer to as the Multi-modal Predicate Identification (MPI) problem.

For certain words (e.g., *heavy*, *soft*, etc.), visual classification of the object is insufficient. The robot would need to perform an action (e.g., lift the object to determine whether it is heavy or not). Multi-modal perception research has focused on combining information arising from such multiple sensory modalities.

Given multi-modal perception capabilities, a robot needs to decide which actions (possibly out of many) to perform on an object, i.e., generate a behavioral

policy for a given request. For instance, to obtain an object's color, a robot could adjust the pose of its camera, whereas sensing the content of an opaque container requires two actions: grasping and shaking. The robot has to select actions in such a way that the information gain about object properties is maximized while the cost of actions is minimized. Sequential reasoning is required in this action selection process, e.g., a shaking action would make sense only if a grasping action has been successfully executed. Also, robot perception capabilities are imperfect, so the robot sometimes needs to take the same action more than once. planning under uncertainty algorithms aim at computing action policies to help select actions toward maximizing long-term utility (information gain in our case), while considering the uncertainty in non-deterministic action outcomes.

MDPs [9] and partially observable MDPs (POMDPs) [10] enable an agent to plan under uncertainty with full and partial observability respectively. However, the observability of real-world domains is frequently mixed: some components of the current state can be fully observable while others are not. A MOMDP is a special form of POMDP that accounts for both fully and partially observable components of the state [73]. In this work, we model the robot MPI problems using MOMDPs because of the mixed observability of the world that the robot interacts with (e.g., whether an object is in hand or not is fully observable, but object properties such as color and weight are not).[1]

Robot behavioral exploration policies are used for suggesting exploration actions given the current world state estimation. In this chapter, the robot learns its policies from the experience of interacting with objects in the real world [74].

[1]Referring to our model as a MOMDP (as opposed to a POMDP) is not of practical importance in this work. It is mainly for ease of describing the domain.

We use datasets that include tens of objects and nearly one hundred properties. In such domains, it frequently takes a prohibitively long time to compute effective behavioral exploration policies. To tackle this issue, we dynamically learn MOMDP-based controllers to model a minimum set of domain variables that are relevant to current user queries (e.g. "red, empty bottle"). This strategy ensures a small state set and enables us to generate high-quality robot action policies in a reasonable time (e.g., ≤ 5 seconds). Our experiments show that the policies of the learned controllers improve accuracy for recognizing new objects' properties while reducing exploration cost, in comparison to baseline strategies that deterministically or randomly use predefined sequences of actions.

5.2 Related Works

Recent research has shown that robots can learn to classify objects using computer vision methods as well as non-visual perception coupled with actions performed on the objects [75, 76, 77]. For example, a robot can learn to determine whether a container is full or not based on the sounds produced when shaking the container [78]; or learn whether an object is soft or hard based on the haptic sensations produced when pressing it [79]. Past work has shown that robots can associate (or *ground*) these sensory perceptions with human language predicates in vision space [80, 81, 82, 83] and joint visual and haptic spaces [84].

Nevertheless, there has been relatively little emphasis on enabling a robot to *efficiently* select actions at test time when it is tasked with classifying a new object. The few approaches for tackling action selection, e.g., [85, 86, 76, 87], assume that only one target property has to be identified (e.g., the object's identity in the

case of object recognition). In contrast, we address the multi-modal predicate identification (MPI) problem where a robot needs to recognize multiple properties about an object, e.g., "is the object a *red empty bottle*?".

Sequential decision-making frameworks, such as MDPs, POMDPs, and MOMDPs, can be used for planning under uncertainty toward achieving long-term goals, while accounting for non-deterministic action outcomes and different observabilities [10, 73]. As a result, these frameworks have been applied to multi-modal predicate identification (MPI) problems on physical objects in robotics. For instance, hierarchical POMDPs were used for suggesting visual operators and regions of interest for exploring multiple objects on a tabletop scenario [88]; the work of Eidenberger and Scharinger further enabled a robot to actively adjust its position to avoid occlusions [89]. More recent work used a robotic arm to move objects enabling better visual analysis [90]. Interaction with objects in these lines of research relies heavily on robot vision while other sensing modalities, such as audio and haptics, are not considered.

Behavioral policies for MPI problems have been learned in simulation using deep reinforcement learning methods [91], where *force* was directly used in the interactions with objects. The simulation environment used in that work makes it possible to run large numbers of trials, but does not establish applicability on real robots.

Behavior	Modality		
	color	**shape**	**deep**
look	64	308	4096
	audio	**haptics**	**proprioception**
grasp	100	60	20
drop, hold, lift, lower, press, push	100	60	

Table 5.1: The number of features extracted from each *context* (i.e., the combination of robot behavior and perceptual modality) for one of the datasets (**Thomason16**) used in our experiments.

5.3 MPI Framework

5.3.1 Multi-modal predicate learning

In this work, the robot learns predicate recognition models using the methodology described in [92, 77], briefly summarized here. The robot interacts with objects using behaviors (e.g., *look*, *grasp*, *lift*) coupled with sensory modalities (e.g., *color*, *haptics*, *audio*). We refer to a combination of a behavior and modality as a sensorimotor context (e.g., *look-color*, *lift-haptics*, etc.), where $\mathcal{C}$ is the set of all such contexts. Table 5.1 shows the set of sensorimotor contexts for one of the datasets used in our experiments (discussed in more detail in Section 5), along with the feature dimensionality for each context. Note that not all modalities are produced by every behavior – for example, the *lift* action does not produce *color* features while the *look* action does not produce *haptic* features.

We connect these feature representations of objects to predicates by learning discriminative classifiers on the feature spaces for each predicate $p \in \mathcal{P}$, the set of all predicates. For each predicate p, and context $c \in \mathcal{C}$, the robot learns a binary classifier using data points $[x_i^c, y_i]$, where x_i^c is the i^{th} feature vector in context c (e.g., in the *look-color* context, the feature vector encodes a color histogram of the object), and $y_i = true$ if the predicate p holds true for the object in trial i,

and $false$ otherwise. We assume that the classifiers' outputs can be mapped to probabilities, i.e., a classifier associated with context c for predicate p can estimate $\mathbf{Pr}_p^c(y_i = true|x_i^c)$.

Let $\mathcal{C}_b \subset \mathcal{C}$ be the set of sensorimotor contexts associated with behavior $b \in \mathcal{B}$. When executing action b, the robot detects a set of feature vectors, $\mathcal{X}_i$ (one vector per each context in $\mathcal{C}_b$), and uses them to query the classifiers associated with contexts $\mathcal{C}_b$. The probability estimates of the classifiers are combined using weighted combination and normalized again to compute the final prediction:

$$\mathbf{Pr}_p(y_i = true|\mathcal{X}_i) = \alpha \times \sum_{x_i^c \in \mathcal{X}_i} w_c^p \times \mathbf{Pr}_p^c(y_i = true|x_i^c)$$

where α is a normalization constant to ensure the probabilities sum up to 1.0 and $w_c^p \in [0.0, 1.0]$ is a reliability weight indicating how good the classifier associate with context c is at recognizing predicate p, as estimated by performing cross-validation on the training data. In other words, each behavior acts as a classifier ensemble where each individual classifier's output is combined using a weighted combination.

At the end of the training stage, cross-validation at the behavior level is used to compute the confusion matrix $C_p^b \in \mathbb{R}^{2\times 2}$ for predicate p and behavior b. These confusion matrices are normalized to compute the True Positive, True Negative, False Positive, and False Negative rates for each behavior-predicate pair, which are later used for dynamically constructing controllers. Example confusion matrices are shown in Figure 5.1. Next, we describe the problem of generating an action policy when identifying whether a set of predicates hold true for a novel object.

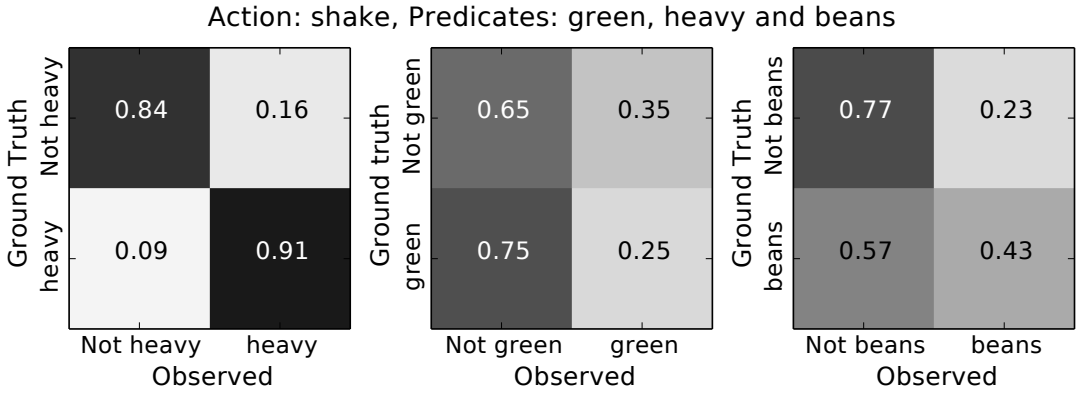

Figure 5.1: Example confusion matrices from one of the datasets (**Sinapov14**) showing the TP, FP, TN, and FN rates for three of the predicates when using the robot's *shake* action. The action is good at recognizing *heavy* due to the rich haptic feedback produced when shaking an object, somewhat good at recognizing *beans* (referring to the objects' contents) due to the sound produced by the contents, and poor at recognizing *green* as no visual input is processed when performing this action.

5.3.2 MOMDP-based SDM

Behaviors (or actions[2]), such as *look* and *drop*, have different costs and different accuracies in predicate recognition. At each step, the robot has to decide whether more exploration behaviors are needed, and, if so, select the exploration behavior that produces the most information. In order to sequence these behaviors toward maximizing information gain, subject to the cost of each behavior (e.g., the time it takes to execute it), it is necessary to further consider preconditions and non-deterministic outcomes of the actions. For instance, *shaking* and *dropping* actions make sense only if a preceding (unreliable) *grasping* action succeeds.

In this work, we assume action outcomes are fully observable and object properties are not. For instance, a robot can reliably sense whether a *grasping* action is successful, but it cannot reliably sense the color of a bottle or whether that

[2]The terms "behavior" and "action" are widely used in developmental robotics and sequential decision-making communities respectively. In this dissertation, the two terms are used interchangeably.

bottle is full. Due to this mixed observability and unreliable action outcomes, we use mixed observability MDPs (MOMDPs) [73] to model the sequential decision-making problem for object exploration.

A MOMDP is fundamentally a factored POMDP with mixed state variables. The fully observable state components are represented as a single state variable x (in our case, the *robot-object status*, e.g., the object is in hand or not), while the partially observable components are represented as state variable y (in our case, the *object properties*, e.g., the object is heavy or not). As a result, (x, y) specifies the complete system state, and the state space is factored as $S = \mathcal{X} \times \mathcal{Y}$, where $\mathcal{X}$ is the space for fully observable variables and $\mathcal{Y}$ is the space for partially observable variables.

Formally, a MOMDP model is specified as a tuple,

$$(\mathcal{X},\ \mathcal{Y},\ A,\ T_{\mathcal{X}},\ T_{\mathcal{Y}},\ R,\ Z,\ \mathcal{O},\ \gamma),$$

where A is the action set, $T_{\mathcal{X}}$ and $T_{\mathcal{Y}}$ are the transition functions for fully and partially observable variables respectively, R is the reward function, Z is the observation set, $\mathcal{O}$ is the observation function, and γ is the discount factor.

The definitions of A, R, Z, $\mathcal{O}$, and γ of a MOMDP are identical to these of POMDPs [10], except that Z and $\mathcal{O}$ are only applicable to $\mathcal{Y}$, the partially observable components of the state space. γ is the discount factor that specifies the planning horizon.

5.4 Experimental Results

Next, we describe the experiments we conducted to evaluate the proposed MOMDP-based perception strategy using MPI problems. The goal was to increase the accuracy in identifying properties of a novel object while reducing the overall action costs required in this process. In all evaluation runs, the object that needs to be identified was not part of the robot's training set when learning the predicate recognition models or the MOMDP parameters. The following baseline action strategies are used in experiments, where belief is updated using Bayes' rule except for *Random*:

- *Random*: Actions are randomly selected from both reporting and legal exploration actions. A trial is terminated by any of the reporting actions.
- *Random Plus*: Actions are randomly selected from legal exploration actions. Under an exploration budget, one selects the reporting action corresponding to y with the highest belief.
- *Predefined*: An action sequence is strictly followed: *ask, look, press, grasp, lift, lower* and *drop*.[3] Under an exploration budget or in early terminations caused by illegal actions, the robot selects the reporting action that makes the best sense.
- *Predefined Plus*: The same as *Predefined* except that unsuccessful actions are repeated until achieving the desired result(s).

[3]Action *ask* was used only in the **Thomason16** experiments, because other exploration actions are not as effective as in **Sinapov14**.

Properties	Method	Overall cost (std)	Accuracy
Two	Random	17.56 (30.00)	0.245
	Predefined Plus	37.10 (0.00)	0.583
	MOMDP (Ours)	29.85 (12.87)	0.860
Three	Random	10.12 (21.77)	0.130
	Predefined Plus	37.10 (0.00)	0.373
	MOMDP (Ours)	33.87 (8.78)	0.903

Table 5.2: Performances of MOMDP-based and two baseline planners in cost (second) and accuracy on the **Sinapov14** dataset. Numbers in parenthesis denote the Standard Deviations over 400 trials.

Sinapov14 Dataset: In each trial, we place an object that has three attributes (color, weight, and content) on a table and then generate an object description that includes the values of two or three attributes. This description matches the object in only half of the trials. When two (or three) attributes are queried, $\mathcal{Y}$ includes four (or eight) states plus *term* state, resulting in $\mathcal{S}$ that includes 25 (or 49) states. The other components of the dynamically constructed MOMDPs grow accordingly, given an increasing number of queried attributes.

Experimental results are reported in Table 5.2. Not surprisingly, randomly selecting actions produces low accuracy. The overall cost is smaller in more challenging trials (all three properties are questioned), because in these trials there are relatively fewer exploration actions (more properties produce more reporting actions), making the agent more likely to take a reporting action. Our MOMDP-based multi-modal perception strategy reduces the overall action cost while significantly improving the reporting accuracy. Our performance improvement is achieved by repeating actions as needed, selecting legal actions (e.g., *lift* is legal only if the current state is x_2) that produce the most information or have the potential of doing so in the future, and even arbitrarily reporting without "wasting" exploration actions given queries where the exploration actions are not effective.

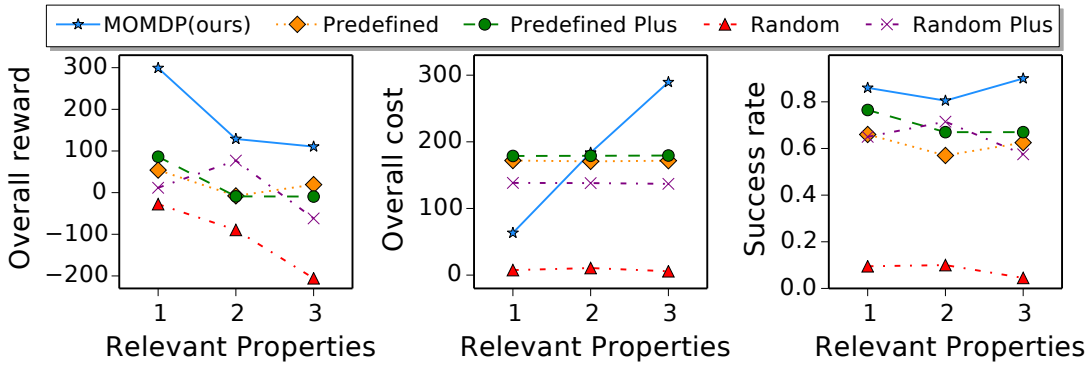

Figure 5.2: Evaluations of five actions strategies on the **Thomason16** dataset. Comparisons are made in three categories of *overall reward* (Left), *overall exploration cost* (Middle), and *success rate* (Right).

Thomason16 Dataset: In this set of experiments, a user query is specified by randomly selecting one object and N properties ($1 \leq N \leq 3$), on which the robot is questioned. Each data point is an average of over 200 trials, where we conducted pairwise comparisons over the five strategies, i.e., the strategies were evaluated using the same set of user queries. A trial is successful only if the robot reports correctly on all properties. It should be noted that most of the contexts are misleading in this dataset due to the large number of object properties, so more exploration actions confuse the robot if the actions are not carefully selected. Figure 5.2 shows the experimental results. The overall reward is computed by subtracting the overall action cost from the reward yielded by the reporting action (either a big bonus or a big penalty). We do not compute standard deviations in this dataset, because the diversity of the tasks results in problems of very different difficulties.

We can see our MOMDP-based strategy consistently performs the best in terms of the overall reward and overall accuracy. When more properties are queried, the MOMDP-based controllers enable the robot to take more exploration actions

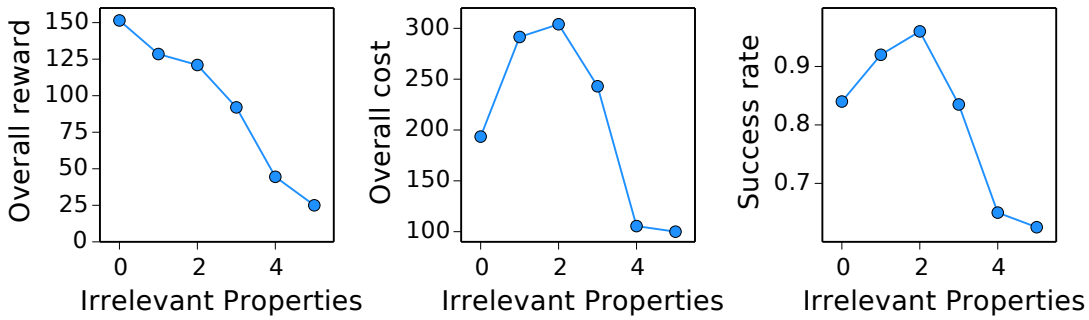

Figure 5.3: A "super" MOMDP that models two relevant plus increasing irrelevant properties, in comparison to our dynamically learned controllers that model only the relevant predicates and correspond to the left end of each curve.

(Middle subfigure), whereas the baselines could not adjust their question-asking strategy accordingly.

The last experiment aims to evaluate the need of dynamically constructed controllers, answering the question "*Can we build a 'super' controller that models all properties?*" We constructed MOMDP controllers including two relevant and an increasing number of irrelevant properties (i.e., the ones that are not queried). Our dynamically learned controllers include only the relevant properties and correspond to the curves' left ends. Results are shown in Figure 5.3. We can see, the quality of the generated action policies decreases soon, e.g., from > 150 to < 25 in reward (what we try to maximize), when more irrelevant properties are included in the controllers. The right two subfigures show that the controllers first try to achieve higher accuracy by taking more exploration actions and then "give up" due to the growing number of irrelevant properties. The results show the infeasibility of "super" controllers that model all properties and justify the need of dynamic controllers.

5.5 Summary of SDM with Multimodal Perception

We investigate using mixed observability Markov decision processes (MOMDPs) to solve the multi-modal predicate identification (MPI) problem, where a robot selects actions for multi-modal perception in object exploration tasks. Our approach can dynamically construct a MOMDP model given an object description from a human user (e.g., "*a blue heavy bottle*"), compute a high-quality policy for this model, and use the policy to guide robot behaviors (such as "look" and "shake") toward maximizing information gain. The dynamically built controllers enable the robot to focus on a minimum set of domain variables that are relevant to the current object and query. The MOMDP perception models are learned using two existing datasets collected with robots interacting with objects in the real world. Experimental results show that our object exploration approach enables the robot to identify object properties more accurately without introducing extra cost from exploration actions compared to a baseline that suggests actions following a predefined action sequence.

6 SDM WITH KNOWLEDGE ACQUISITION

In this chapter, we discuss a robot knowledge acquisition approach through robot SDM. In particular, the robot constructs a statistical dialog manager using SDM methods, and augments its knowledge base via human-robot dialog.

6.1 Introduction

Mobile robots have been extensively used to conduct tasks, such as guidance and object delivery, in the real world. Notable examples include the Amazon warehouse robots and the Relay robots from Savioke. However, these robots either work in human-forbidden environments, or have no interaction with humans except for obstacle avoidance. Researchers are developing mobile robot platforms that are able to interact with people in every day, human-inhabited environments [3, 1, 93, 2]. Some of the robot platforms can learn from the experience of human-robot interaction (HRI) to improve their language skills, e.g., learning new synonyms [94], but none of them learn entirely new entities. This work aims at a multitask dialog management problem, where a robot simultaneously identifies service requests through human-robot dialog and learns new entities from this experience to augment its internal knowledge base (KB).

A robot dialog system typically includes at least three components for lan-

Figure 6.1: Our dialog agent is implemented and deployed on a Segway-based mobile robot platform (front and back).

guage understanding: state tracking, dialog management, and language synthesis. Our dialog agent includes the four components by further supporting dialog-based knowledge augmentation. Our dialog system is *goal-oriented*, and aims at maximizing information gain. In this setting, people prefer dialog agents that are able to accurately identify human intention using fewer dialog turns.

Goal-oriented dialog systems are necessary for language-based human-robot interaction because, in most cases, people cannot *fully* and *accurately* deliver information using a single dialog turn. Consider a service request of "*Robot, please deliver a coffee to the conference room!*" It is possible that the robot does not know which conference room the speaker is referring to, in which case it is necessary to ask clarification questions such as "*Where should I deliver a coffee?*" in order to perform the correct action. To further identify the service request, the robot might want to ask about the recipient as well: "*For whom is the delivery?*" Although such goal-oriented dialog systems have been implemented on robots,

few of them can learn to improve their language capabilities or augment their KB from the experience of human-robot conversations in the real world (details in Section 6.2).[1]

This chapter focuses on dialog-based robot knowledge augmentation, where the agent must identify *when* it is necessary to augment its KB and *where* in the KB to do that, as applied to our Segway-based mobile robot shown in Fig. 6.1. In this chapter, we develop a dual-track dialog manager to help the agent maintain a confidence level of how well the current dialog is being supported by the KB, and accordingly decide to whether to augment its KB or not [95]. After the agent becomes confident that new entities are necessary so as to make progress in the dialog, it decides where in the KB to add a new entity (e.g., a new *item* or a new *person* is being referred to by the user) by analyzing the flow of the dialog. As a result, our dialog agent is able to decide both *when* and *how* to augment its KB in a semantically meaningful way.

Our dialog system has been evaluated in simulation and in the real world. Results show that our dialog system performs better in service request identification (both efficiency and accuracy), in comparison to baselines that use predefined strategies. Human-subject experiments suggest that our knowledge augmentation component improves user experience as well.

6.2 Related Works

Researchers have developed algorithms for learning to interpret natural language commands [96, 97, 98]. Recent research enabled the co-learning of syntax

[1]In comparison, there are dialog agents that aim at maximizing social engagement and prefer extended conversations, e.g., Microsoft XiaoIce, which are beyond the scope of this work.

and semantics of spatial language [99, 100]. Although the systems support the learning of language skills, they do not have a dialog management component (implicitly assuming perfect language understanding), and hence do not readily support multi-turn communications.

Algorithms have been developed for dialog policy learning [101, 9, 102]. Recent research on Deep RL has enabled dialog agents to learn complex representations for dialog management [103, 104]. The systems do not include a language parsing component. As a result, users can only communicate with their dialog agents using simple or predefined language patterns.

Mobile robot platforms have been equipped with semantic parsing and dialog management capabilities. After a task is identified in dialog, these robots are able to conduct service tasks using a task planner [93, 36, 37]. Although these works enable a robot to identify human requests via dialog, they do not enable learning from these experiences.

Dialog agents for mobile service robots have been developed for identifying service tasks such as human guidance and object delivery [94, 105]. A dialog manager suggests language actions for asking clarification questions, and the agent is able to learn from human-robot conversations. These methods focus on learning to improve an agent's language capabilities, but do not augment its knowledge base in this process (i.e., only pre-defined people, objects, and environmental locations can be reasoned about by the robot). This work builds on the dialog agent implemented by [94], and introduces a dual-track dialog-knowledge manager and a strategy for augmenting the robot's knowledge base.

There are other dialog agents that specifically aim at knowledge augmentation

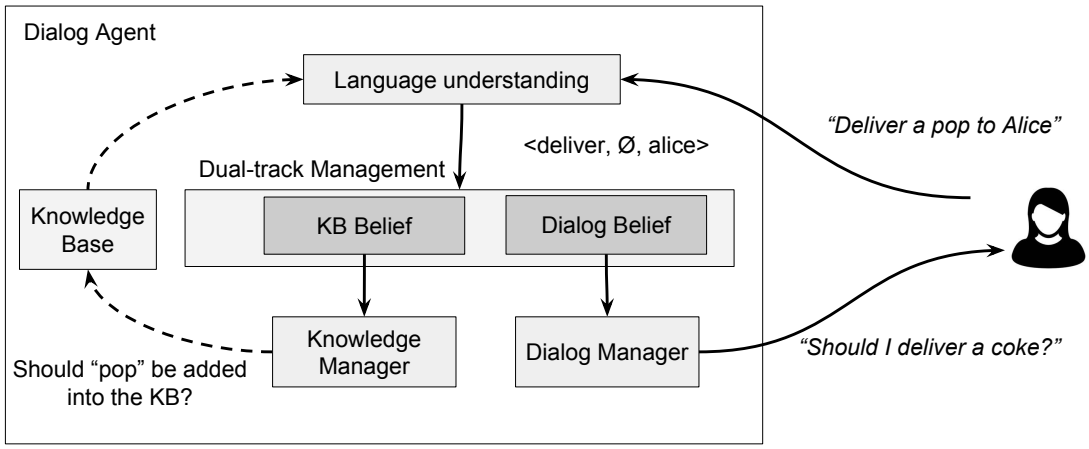

Figure 6.2: A pictorial overview of our dialog system, including a hybrid parser for language understanding, and two management tracks for knowledge base (KB) and dialog respectively.

through human-robot dialog [106, 107]. An instructable agent is able to learn new concepts and new procedural knowledge through human-robot dialog [108]. Recent work enabled a mobile robot to ground new concepts using visual-linguistic observations, e.g., to ground new word "box" given a command of "move to the box" by exploring the environment and hypothesizing potential new objects [109]. These agents are able to augment their knowledge bases through language-based interactions with humans. However, their dialog management components (if any) do not model the noise in language understanding. Researchers developed a robot dialog system that focuses on situated verb semantics learning [110]. Their dialog agent uses RL to learn a dialog management policy, and uses a semantic parser to process natural language inputs. A recent paper surveyed research on robot learning new tasks through natural language and action demonstration [111]. These works focused on learning the semantics of verbs, limiting the applicability of their knowledge augmentation approach. A recent work focuses on planning with open-world knowledge by reasoning about hypothetical objects while we focus

more on modeling the language uncertainty [47].

Our dialog agent is the first that together: 1) processes language inputs using a semantic parser to understand users' service requests, 2) leverages a dialog manager to account for the unreliability from the parser, and 3) augments its knowledge base using a knowledge manager.

6.3 Dialog-based Knowledge Augmentation Framework

In this section, we present our dialog agent that integrates a decision-theoretic dialog manager, and an information-theoretic knowledge manager, as illustrated in Fig. 6.2.

6.3.1 Dialog and Knowledge Management

Markov decision process (MDP) is a general sequential decision-making framework that can be used for planning under uncertainty [9]. POMDPs [10] generalize MDPs to situations where ground truth world knowledge is fuzzy. POMDPs have been used for dialog management [112], where the intentions of the interlocutors are latent. There are two interleaved control loops in our dialog agent, resulting in a *dual-track controller*. One track focuses on maintaining the dialog belief state, and suggests language actions to the agent. The other focuses on maintaining the belief of the current knowledge being (in)sufficient to complete the task, and suggests knowledge augmentation.

Our dialog agent is implemented on a mobile service robot that communicates with human users using natural language to identify service tasks in the form of

$$< task, item, recipient >$$

where the agent must efficiently and accurately identify the service request (with unreliable language understanding capabilities) while augmenting KB on an as-needed basis.

Dialog Management Track

The dialog management POMDP includes the following components:

- $S : \{S^T \times S^I \times S^R\} \cup term$, where S^T is the set of task *types* (delivery and guidance in our case), S^I is the set of *items* used in the task, S^R is the set of recipients of the delivery, and *term* is the terminal state.

- $A : A^W \cup A^C \cup A^R$ is the action set. A^W consists of general "wh" questions, such as "*Whom is this delivery for?*" and "*What item should I deliver?*". A^C includes confirming questions that expect yes/no answers. Reporting actions A^R return the estimated human requests.

- $T : S \times A \times S' \rightarrow [0, 1]$ is the state-transition function. In our case, the dialog remains in the same state after question-asking actions, and reporting actions lead transitions to *term* deterministically.

- $R : S \times A \rightarrow \mathbb{R}$ is the reward function. The reward values are assigned as:

$$R(s,a) = \begin{cases} r^C, & \text{if } s \in S, a \in A^C \\ r^W, & \text{if } s \in S, a \in A^W \\ r^+, & \text{if } s \in S, a \in A^R, s \odot a \\ r^-, & \text{if } s \in S, a \in A^R, s \otimes a \end{cases}$$

where r^C and r^W are the costs of confirming and general questions, in the form of negative, relatively small values; r^+ is a big bonus for correct reports; and r^- is a big penalty (negative) for incorrect reports.

- $Z : Z^T \cup Z^I \cup Z^R \cup \{z^+, z^-\}$ is the set of observations, where Z^T, Z^I and Z^R include observations of *task type*, *item*, and *recipient* respectively. z^+ and z^- correspond to "*yes*" and " *no*". Our dialog agent takes in observations as semantic parses that have correspondence to elements in Z. Other parses, including the malformed ones, produce random observations (detailed shortly).

- $O : S \times A \times Z \cup$ *inapplicable* is the observation function that specifies the probability of observing $z \in Z$ in state $s \in S$, after taking action $a \in A$. Reporting actions yield the *inapplicable* observations. Our observation function models the noise in language understanding, e.g., the probability of correctly recognizing z^+ ("*yes*") is 0.8. The noise model is heuristically designed in this work, though it can be learned from real conversations.

Solving this POMDP generates a policy π, which maps a belief to a language action ($a \in A$) that maximizes long-term information gain.

Knowledge Management Track

In addition to the dialog management POMDP, we have a knowledge management POMDP that monitors whether the agent's knowledge is sufficient to support estimating human intentions. The knowledge management POMDP formulation is similar to that for dialog management but includes entities for unknown items and recipients. The components of the knowledge management POMDP are:

- $S^+ : S \cup \{(s^T, s^I, \hat{s}^R) \mid \forall s^T \in S^T, \forall s^I \in S^I\} \cup \{(s^T, \hat{s}^I, s^R) \mid \forall s^T \in S^T, \forall s^R \in S^R\}$ the set of states. It includes all states in S along with the states corresponding to new entities that correspond to an unknown item $\hat{s}^I$ and an unknown recipient $\hat{s}^R$;

- $A^+ : A \cup \{\hat{a}^I, \hat{a}^R\} \cup \hat{A}^R$ the set of actions including the actions in A, two actions ($\hat{a}^I$ and $\hat{a}^R$) for confirming the unknown *item* and *recipient*, and $\hat{A}^R$, reporting actions that correspond to the states in S^+;

- $Z^+ = Z \cup \{\hat{z}^I, \hat{z}^R\}$ the augmented observation set, including $\hat{z}^I$ and $\hat{z}^R$ for unknown item and recipient.

Transition and observation functions are generated accordingly and hence not listed.

At runtime, we maintain belief distributions for both tracks of POMDPs. Belief b of dialog POMDP is used for sequential decision making and dialog management, and belief b^+ of knowledge POMDP is only used for language augmentation purposes, i.e., determining when it is necessary to augment the KB. When observations are made (observation $z \in Z$), both beliefs are updated using the Bayes update rule [10]. In our dialog system, observations are made based on the language understanding using a semantic parser.

The dual-track controller identifies the first contribution of this work. We use a dual-track, instead of merging them to unify the action selection process, because the knowledge track is only used for the purpose of maintaining beliefs (not for action selection), and modeling the uncertainty of unknown entities in a single controller will result in unnecessarily long dialogs. Separating the two tracks reduces the learning complexity of the entire framework.

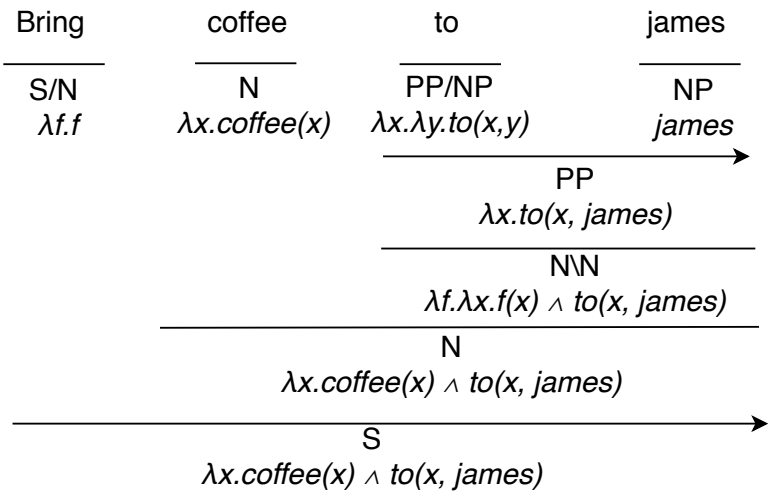

Figure 6.3: An example of parsing a service request sentence using CCG semantic parsing and λ calculus.

6.3.2 Language Understanding

In order to understand natural language and make observations for POMDPs, we use a semantic parser that builds on the Semantic Parsing Framework (SPF) described in [113]. The input of the semantic parser is natural language from human users, and the output is a list of possible parses for a given sentence. Using the semantic parser, the request in natural language is transformed to a formal representation compatible with the robot's internal KB.

Figure 6.3 shows an example of the parser recognizing a sentence. It can reason over the ontology of the known words when it parses a sentence, e.g., *james:pe* and *coffee:it*. The dialog manager can use this information to translate from words to the corresponding observation for the question asked by the robot. If the language understanding fails (e.g., producing parses that are malformed or do not comply with the preceding questions), then a random observation from Z will be made for the unknown part of the request (introducing enough entropy to move the dialog

along).

6.3.3 Domain Knowledge Representation

We use Answer Set Prolog (ASP) [8], a declarative language, for knowledge representation. The agent's knowledge base (KB), in the form of an ASP program, includes rules in the form of:

$$l_0 \leftarrow l_1,\ \cdots,\ l_n,\ \text{not } l_k,\ \cdots,\ \text{not } l_{n+k}$$

where l's are literals that represent whether a statement is true. The right side of a rule is the *body*, and the left side is the *head*. The `not` symbol is called default negation, representing no evidence supporting a statement.

The KB of our agent includes a set of *entities* in ASP: {*alice, sandwich, kitchen, office1, delivery,* $\cdots$}, where *delivery* specifies the task type. A set of *predicates*, such as {*recipient, item, task, room*}, are used for specifying a category for each object. As a result, we can use ASP rules to fully specify tasks, such as "*deliver a coke to Alice*":

$$task(delivery).$$

$$item(coke).$$

$$recipient(alice).$$

One can easily include more information into the ASP-based KB, such as rooms, positions of people, and a categorical tree of objects. Robot's KB is built on a lexicon that is a collection of information about the words of a language about

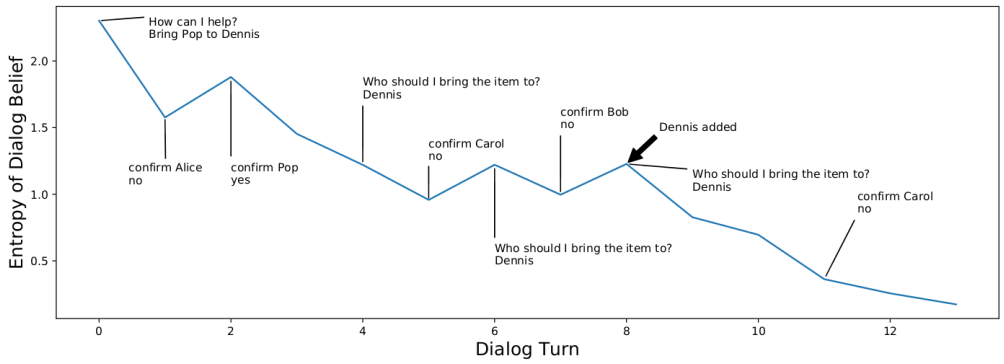

Figure 6.4: In this example, the user requested a *Pop* to a novel recipient, *Dennis*. The dialog agent understood *Pop*, but not *Dennis* (Turn 0), and it mistakenly observed *Alice* as the recipient. The user denied *Alice* (Turn 1), and confirmed *pop* (Turn 2). The conversation continued, as our dialog agent kept trying to identify the recipient, until the number of EFs crossed a threshold (5 in this case) in Turn 8. Accordingly, *Dennis* was added into the KB as a new recipient entity. The agent continued asking clarification questions while being aware of *Dennis*, until the dialog manager suggested the correct reporting action and delivers *pop* to *Dennis*. Although the clarification question may frustrate the human, it makes the robot more confident in estimating human request.

the lexical categories. This ASP-based KB can be used for query responding and task planning purposes, where the query and/or task are specified by our dialog agent.

6.3.4 Algorithm for Knowledge Augmentation

We define a few functions before introducing the main algorithm for simultaneous dialog management and knowledge augmentation. We use *entropy* to measure the uncertainty level of the agent's belief distribution:

$$H(b) = -\sum_{i=0}^{n-1} b(s_i) \cdot \log b(s_i)$$

When the agent is (un)confident about the state, the entropy value is (high) low. In particular, a uniform belief distribution corresponds to the highest entropy level. We use entropy for the following two purposes in our algorithm.

I) Rewording Service Request If the belief entropy is higher than threshold h (meaning the agent is highly uncertain about the dialog state), we encourage the human user to state the entire request in one sentence. Otherwise, the dialog manager decides the flow of the conversation.

II) Entropy Fluctuation We introduce the concept of *entropy fluctuation* (EF):

$$f(\mathbf{b}) = sign\Big(H(\mathbf{b}[1]) - H(\mathbf{b}[0])\Big) \oplus sign\Big(H(\mathbf{b}[2]) - H(\mathbf{b}[1])\Big)$$

where $\mathbf{b}$ is a belief queue that records dialog beliefs of the last three dialog turns, $f(\mathbf{b})$ outputs *true*, if there is an EF in the last three beliefs (i.e., entropy of the second last is the highest or lowest among the three), and $\oplus$ is the *xor* operator.

Algorithm for Dialog-Knowledge Management Algorithm 1 shows the main operation loop of the dialog system. $\mathcal{M}$ and $\mathcal{M}^+$ are models for dialog-track and knowledge-track control respectively; τ_b is a probability threshold; h is an entropy threshold; and Δ is a threshold over the number of EFs.

The algorithm starts by initializing the two beliefs with uniform distributions. δ, which counts the number of EFs, is initialized to 0. If the marginal probability over $\hat{s}^R$ (or $\hat{s}^I$) of knowledge belief b^+ is higher than threshold τ_b, or the number of EFs is higher than Δ, we add a new entity into the KB. If the entropy of dialog belief is higher than h, then the agent asks for rewording the original service re-

Algorithm 1 Dialog-based Knowledge Augmentation

Require: $\mathcal{M}, \mathcal{M}^+, \tau_b, h, \Delta$, and a POMDP *solver*

Initialize b, b^+ with uniform distributions
Initialize EF counter $\delta \leftarrow 0$
Initialize queue **b** of size 3 with $\{b, b, b\}$
repeat
  if $\sum_{s^R=\hat{s}^R} b^+\big(s(s^T,\ s^I,\ s^R)\big) > \tau_b$ **then**
    Add a new recipient entity in KB
  else if $\sum_{s^I=\hat{s}^I} b^+\big(s(s^T,\ s^I,\ s^R)\big) > \tau_b$ **then**
    Add a new item entity in KB
  else if $\delta > \Delta$ **then**
    Add (item or recipient) entity that is more likely
  end if
  if $f(\mathbf{b})$ **is** *true* **then**
    $\delta \leftarrow \delta + 1$
  end if
  if $H(b) > h$ **then**
    $a \leftarrow$ "*Please reword your service request*"
  else
    $a \leftarrow \pi(b)$
  end if
  $o \leftarrow parse(human\ response)$
  Update b based on observation o and action a
  b.enqueue(b)
  if $a \in A^C$ **then**
    Update b^+ based on observation
  end if
until s **is** *term*
return the request based on the last (reporting) action, and the (possibly updated) knowledge base.

quest. Otherwise, we use the dialog POMDP to maintain the dialog flow. Finally, the knowledge belief is only updated by confirming questions, which can invalidate the agent hypothesis of unknown entities. The algorithm returns the request and the updated knowledge base. When adding a new entity, the agent explicitly asks the user to help specify the name of the unknown item (or person). The KB is updated along with the robot's lexicon for the semantic parser. The index for the unknown item or person is associated with the new entry. We utilized two functions to calculate the parameters τ_b and Δ:

$$\tau_b(|\text{KB}|) = \frac{1}{1 + e^{-\left\lfloor \sqrt{|KB|} \right\rfloor}} - \frac{1}{\left\lfloor \sqrt{|KB|} \right\rfloor}$$

$$\Delta(|KB|) = max(0, \left\lfloor \sqrt{|KB|} \right\rfloor)$$

With the new knowledge added to KB, POMDPs are dynamically constructed so that the dialog can continue seamlessly, and the belief b is replaced with reinitialized b^+. Fig. 6.4 illustrates an example dialog.

6.4 Experimental Results

We have evaluated our dialog agent both in simulation and with human participants. When the user verbalizes the entire request, the agent receives a sequence of three (unreliable) observations on *task*, *item* and *recipient* in a row. Unreliable language understanding is modeled in POMDP observations, e.g., the agent can correctly recognize "coffee" with probability (0.8 in our case), and this probability decreases given more items in the KB. The reward of confirming questions is $R^C = -1.0$, and the reward of wh-questions is $R^W = -1.5$. The above settings were shared in experiments both in simulation and with human participants. POMDPs are solved using an off-the-shelf system [11].

Experiments were mainly designed to evaluate the following hypotheses: Our algorithm is able to I) Efficiently and accurately identify whether there is the need for KB augmentation or not, in case there is the need; II) Augment KB with

Table 6.1: F1 Score of KB Update Given Different KB Sizes

Agent	**KB Size**	**F1 Score (std.)**
Dual Track Manager	17	0.79 (0.020)
Baseline I		0.59 (0.030)
Baseline II		0.61 (0.017)
Dual Track Manager	26	0.77 (0.025)
Baseline I		0.52 (0.016)
Baseline II		0.66 (0.007)
Dual Track Manager	37	0.62 (0.011)
Baseline I		0.47 (0.019)
Baseline II		0.46 (0.022)

higher F1 score under the noise in language understanding; and III) Both augment KB and recognize human intention in the service request with higher success rate while minimizing QA cost.

We compared our algorithm with two learning baselines that use predefined strategies to update their KB: Baseline-I augments KB only when the marginal probability of $\hat{s}^R$ ($\hat{s}^I$) reaches τ_b, and Baseline-II augments KB only when the number of EF δ reaches threshold Δ.

Evaluation metrics used in the experiments consist of: **QA cost**, the total cost of QA actions; **Accuracy**, in an accurate trial, robot correctly identifies its KB entity inadequacy; **Success rate**, where a trial is deemed successful, if the service request is correctly identified *and* (if needed) the KB is correctly augmented; and **Dialog reward**, where QA cost and bonus/penalty are considered together. Focusing on the knowledge augmentation accuracy, we also use **F1 score** as a harmonic average of precision and recall in evaluation.

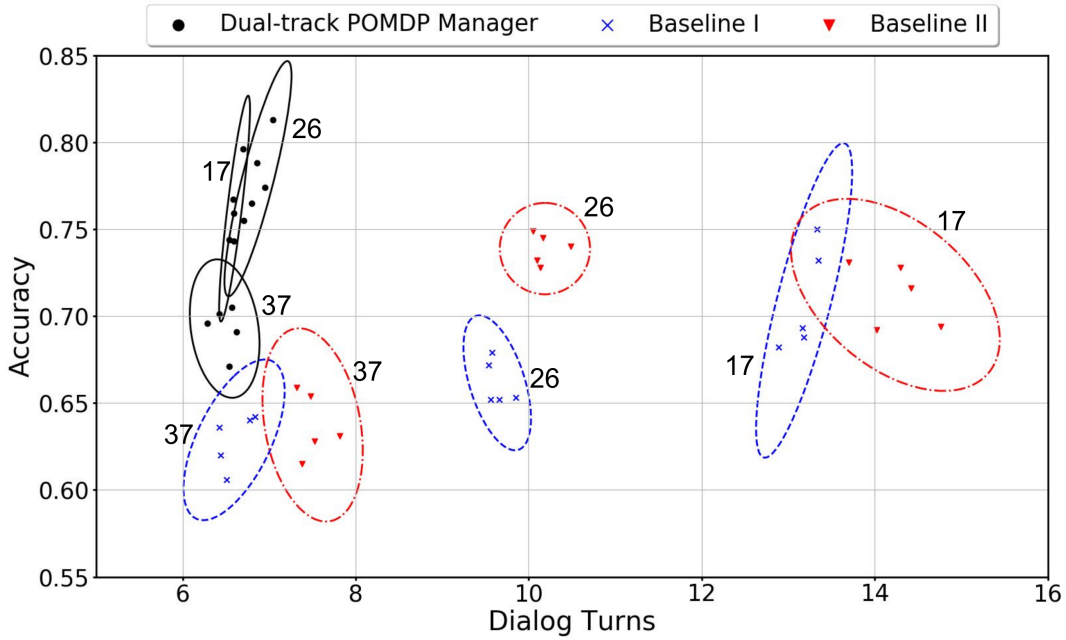

Figure 6.5: Our dialog agent (Shown in •) is able to detect the *need* of KB augmentation with higher accuracy in fewer dialog turns compared to baselines. Covariance error ellipses were calculated for 5 batches for each domain size. The numbers next to data points denote the KB size.

6.4.1 Experiments in Simulation

To evaluate each of the hypotheses, we simulated 3,000 trials over various domain sizes. In each trial, a task, an item, and a recipient are sampled. $|KB|$ is all possible combinations of item/recipient plus the terminal state. For instance, $|KB| = 17$ corresponds to a domain with 1 task, 4 items, and 4 recipients. In the first experiment, we evaluated KB update accuracy versus the dialog turn in which the KB update has occurred (Hypothesis-I). Our algorithm consistently detects the need for KB augmentation earlier (fewer dialog turns) and with higher accuracy while baselines require longer conversations to figure out if they need a KB update (Fig. 6.5).

We further evaluated whether the entity added to KB matches with the human intention or not (Hypothesis-II). As presented in Table 6.1, our algorithm

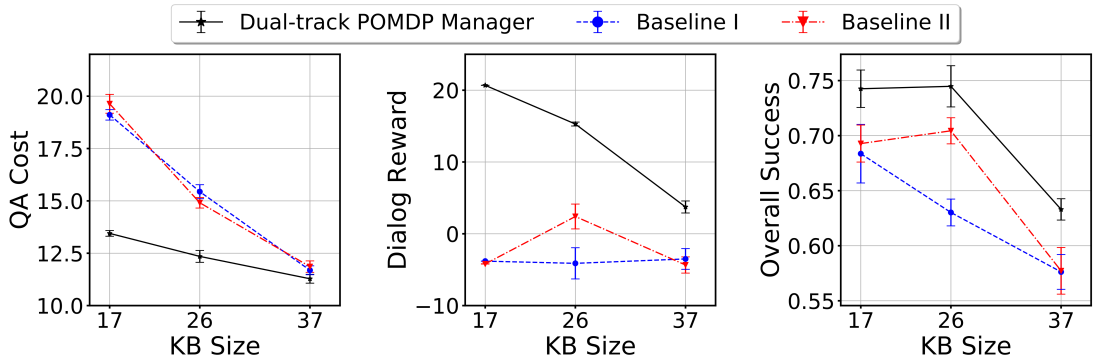

Figure 6.6: Comparison between our agent and baselines in terms of both dialog and knowledge management.

consistently maintains a higher F1 score in comparison to other baseline agents in the medium-sized KB. Finally, we evaluated how our algorithm is capable of both correct KB augmentation as well as correct execution of the task (Hypothesis-III). Figure 6.6 shows that, our agent consistently maintains higher dialog reward while achieving lower QA cost and higher overall success. As the domain size increases, the agent gives up asking further questions that results in lower overall success and reward.

6.4.2 Experiments with Human Participants

Twelve students of ages 19-30 volunteered to participate in an experiment where they asked the robot to conduct delivery tasks using the items and recipients shown in Fig. 6.7. Two items and two recipients in the lists were unknown to the robot, resulting in about 49% (i.e., $1 - \frac{5}{7} \times \frac{5}{7}$) of the service requests not requiring knowledge augmentation. The participants were not aware of this setting, and arbitrarily chose any item-recipient pair to form a delivery task. Each participant conducted the experiment in two (randomly ordered) trials, where the robot used our dialog agent and a baseline agent with a static KB respectively.

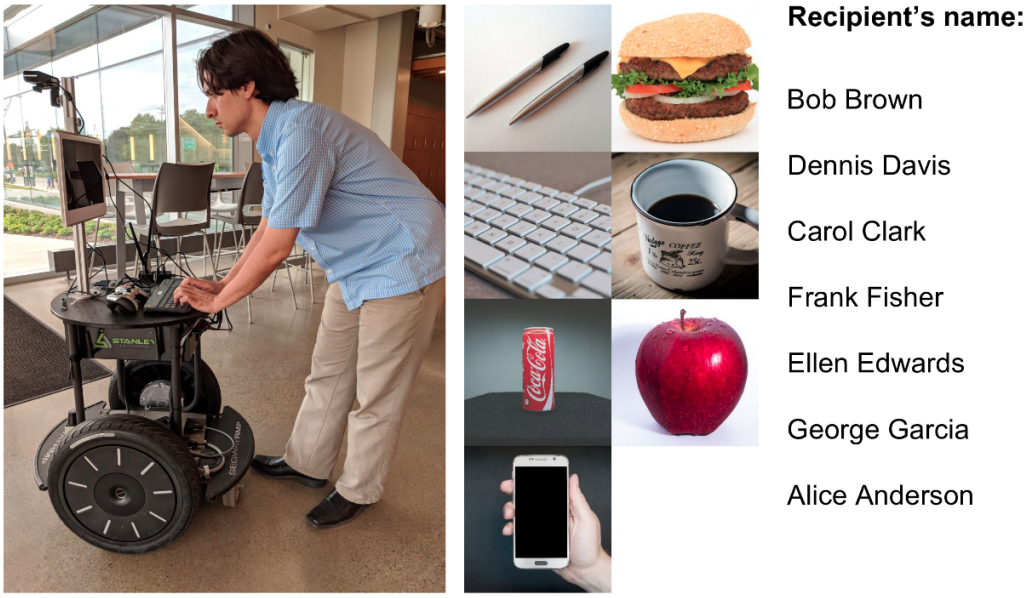

Figure 6.7: *Left*: A user is interacting with our robot. *Right*: Items and recipients used for specifying delivery tasks.

By the end of each dialog, each participant filled out a survey form that includes prompts: Q1, *Task is easy to participants*; Q2, *Robot understood participant*; Q3, *Robot frustrated participant*; Q4, *Participant will use the robot in the future.* The response choices range from 0 (Strongly disagree) to 4 (Strongly agree). Fig. 6.8 shows results from the survey papers, and Table 6.2 shows the average scores. At the 0.1 confidence level, our dialog agent performed significantly better in response for Q3 (frustrated) and Q4 (usefulness). There is no significant difference observed in responses to the other two questions.

Mechanical Turk Experiment Experiments have been conducted with 103 human participants via MTurk. The setup was the same as the robot experiment, except that we used a more challenging baseline. The baseline agent augments its KB after a fixed number of N dialog turns ($N = 8$ in our case). Each worker participated in only one trial (using our agent or the baseline, randomly selected).

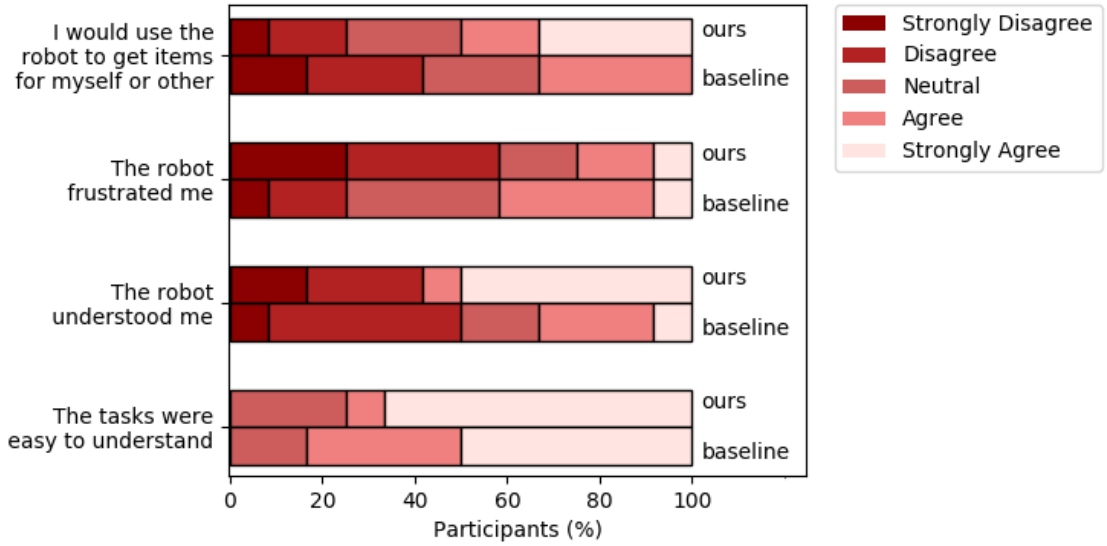

Figure 6.8: Results of survey papers from participants, including four statements with Likert-scale responses.

At a confidence level of 0.1, we found our agent to use significantly fewer dialog turns and achieve a significantly higher success rate on average. Despite the quantitative improvements, there was no significant difference observed from the scores collected from the survey prompts.[2]

Table 6.2: Results of the human participant experiment.

	Q1	Q2	Q3	Q4
Our dialog agent	3.42	2.50	**1.50**	**2.50**
A baseline agent with static KB	3.33	1.83	2.17	1.75

An Example Trial on a Mobile Robot Table 6.3 shows the dialog in a human-robot trial.[3] The semantic parser failed to correctly parse the request,

[2]This could be because the testing domain is static and relatively simple, in the sense of the numbers of items and people. Additionally, MTurk workers are less invested in qualitative feedback than human users in the presence of a real robot.

[3]The agent was implemented using Robot Operating System (ROS) [25].

because it did not know the meaning of *get*. As a result, the dialog agent made a random guess that is "*Alice requesting hamburger*". The robot asked again for clarification on what it should deliver. The user responded "*coffee*", making the robot confused between *coffee* and *hamburger*. The robot asked for confirmation about "*hamburger*", and got denied by the user. Although the user was explicitly guided to specify the recipient by names provided in Figure 6.7, this participant used "me", an unknown word to the robot. When the robot heard "*Nate*" the first time, it could not understand, and mistakenly interpreted it as "*Ellen*". As a result, the agent confirmed "*Ellen*", and got denied. After the negative responses, it added (*Nate*) to its KB, and became confident about the request.

6.5 Summary of SDM with Knowledge Acquisition

We introduced a dialog agent that simultaneously supports human intention identification and knowledge augmentation on an as-needed basis. Experiments show that our dual-track POMDP controller enables the agent to simultaneously conduct dialog and knowledge management. In comparison to a baseline that augments its knowledge base after a fixed number of turns, our dialog agent consistently produces higher overall dialog success. Experiments with human participants show that our agent is more successful in augmenting knowledge and estimating human intention, and people are more willing to use our system.

Table 6.3: An example dialog from a human participant.

Robot	**Human**
How can I help you?	Get me coffee
What item should I bring?	Coffee
Do you want me to deliver hamburger?	No
Who should I bring the item to?	Nate
Is this delivery for ellen?	No
It seems I do not know the person you are talking about. Please write their name so I can learn it.	Nate
Who should I bring the item to?	Nate
Is this delivery for Nate?	Yes
What item should I bring?	Coffee
Execute: Robot brings coffee for Nate; the dialog is over.	

7 SDM WITH COMMONSENSE REASONING

In this chapter, we discuss in detail why leveraging supervised learning and reasoning methods would assist the robot to do SDM under uncertainty.

7.1 Introduction

Mobile robots have been able to operate in everyday environments over extended periods of time, and travel long distances that have been impossible before, while providing services, such as escorting, guidance, and delivery [1, 2, 3]. Sequential decision-making (SDM) plays a key role toward robot long-term autonomy, because real-world domains are stochastic, and a robot must repeatedly estimate the current world state and decide what to do next.

There are at least three AI paradigms, namely *supervised learning*, *automated reasoning*, and *planning under uncertainty*, that can be used for robot SDM. Each of the three paradigms has a long history with rich literature. *However, none of the three completely meet the requirements in the context of robot SDM.* First, a robot can use supervised learning to make decisions, e.g., to learn from the demonstrations of people or other agents [7]. However, the methods are not designed for reasoning with declarative contextual knowledge that is widely available in practice. Second, knowledge representation and reasoning (KRR) methods can be

used for decision making [8]. However, such knowledge can hardly be comprehensive in practice, and robots frequently find it difficult to survive from inaccurate or outdated knowledge. Third, planning under uncertainty methods support active information collection for goal achievement, e.g., using decision-theoretic frameworks such as MDPs [9] and partially observable MDPs (POMDPs) [10]. However, the planning frameworks are ill-equipped for incorporating declarative contextual knowledge.

In this chapter, we develop a robot SDM framework that enables the simultaneous capabilities of learning from past experiences, reasoning with declarative contextual knowledge, and planning toward achieving long-term goals [39]. Specifically, we use *long short-term memory* (LSTM) [22] to learn a classifier for passive state estimation using streaming sensor data, and use *commonsense reasoning and planning under uncertainty* COmmonsense Reasoning and Probabilistic Planning (CORPP) [36] for active perception and task completions using contextual knowledge and human-robot interaction. Moreover, the dataset needed for supervised learning can be augmented through the experience of active human-robot communication, which identifies the second contribution of this work. The resulting algorithm is called learning-CORPP (LCORPP), as overviewed in Figure 7.1.

We apply LCORPP to the problem of *human intention estimation* using a mobile robot. The robot can passively estimate human intentions based on their motion trajectories, reason with contextual knowledge to improve the estimation, and actively confirm the estimation through human-robot interaction (dialog-based and motion-based). Results suggest that, in comparison to competitive baselines from the literature, LCORPP significantly improves a robot's performance in state

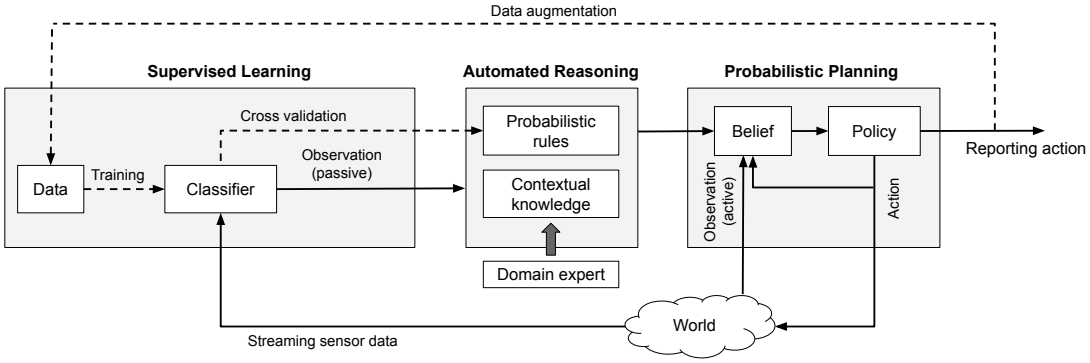

Figure 7.1: An overview of LCORPP that integrates supervised learning, automated reasoning, and planning under uncertainty. Streaming sensor data, e.g., from RGB-D sensors, is fed into an offline-trained classifier. The classifier's output is provided to the reasoner along with classifier's cross-validation. This allows the reasoner to encode the uncertainty of the classifier's output. The reasoner reasons with declarative contextual knowledge provided by the domain expert, along with the classifier's output and accuracies in the form of probabilistic rules, and produces a prior belief distribution for the probabilistic planner. The planner suggests actions to enable the robot to actively interact with the world, and determines what actions to take, including when to terminate the interaction and what to report. At each trial, the robot collects the information gathered from the interaction with the world. In case of limited experience for training, LCORPP supports data augmentation through actively seeking human supervision. Solid and dashed lines correspond to Algorithms 2 and 3 respectively, as detailed in Section 7.3.

estimation (in our case, human intention) in both accuracy and efficiency. [1]

7.2 Related Works

In this section, we discuss the existing research that has integrated at least two of the three aforementioned AI paradigms namely supervised learning, automated reasoning, and planning under uncertainty for SDM.

KRR and SDM: SDM algorithms can be grouped into two classes depending on the availability of the world model, namely planning under uncertainty, and reinforcement learning (RL). Next, we select a sample of the SDM algorithms

[1] A demo video is available: https://youtu.be/YgiB1fpJgmo

that leverages KRR methods, and make comparisons with LCORPP.

When the world model is unavailable, SDM can be realized using RL. People have developed algorithms for integrating KRR and RL methods [27, 28, 29, 30, 31, 32, 114]. Among them, Leonetti, Iocchi, and Stone 2016 used a declarative action knowledge to help an agent select only the reasonable actions in RL exploration. Yang et al. 2018 developed an algorithm called PEORL that integrates hierarchical RL with task planning, and this idea was applied to deep RL by Lyu et al. 2019.

When world models are provided beforehand, one can use planning under uncertainty methods to realize SDM. Contextual knowledge and declarative reasoning have been used to help estimate the current world state in planning under uncertainty [42, 36, 35, 43, 38, 44, 37]. Work closest to this contribution is the CORPP algorithm, where hybrid reasoning (both logical and probabilistic) was used to guide planning under uncertainty by calculating a probability for each possible state [36]. More recently, researchers have used human-provided declarative information to improve robot planning under uncertainty [38].

The main difference from the algorithms is that LCORPP is able to leverage the extensive data of (labeled) decision-making experiences for continuous passive state estimation. In addition, LCORPP supports collecting more data from the human-robot interaction experiences to further improve the passive state estimator over time.

KRR and Supervised Learning: Reasoning methods have been incorporated into supervised learning in natural language processing (Natural Language Processing (NLP)) [115, 116] and computer vision [117, 118, 119, 120] among others. For instance, Chen et al. in 2020 used commonsense knowledge to add missing

information in incomplete sentences (e.g., to expand "pour me water" by adding "into a cup" to the end); and Aditya et al. used spatial commonsense in the Visual Question Answering (Visual Question Answering (VQA)) tasks. Same as those methods, LCORPP includes components for both KRR and supervised learning. Beyond that, LCORPP is a SDM framework that further supports robot decision making to achieve long-term goals, which identifies the key difference between LCORPP and the above-mentioned algorithms.

SDM and Supervised Learning: Researchers have developed various algorithms for incorporating human supervision into SDM tasks [63, 74, 121, 64]. Among them, Amiri et al. 2018 used planning under uncertainty for SDM under mixed observability, where the observation model was learned from annotated datasets. Taylor and Borealis surveyed a few ways of improving robots' SDM capabilities with supervision (in the form of demonstration or feedback) from people. In comparison to the above methods, LCORPP is able to leverage human knowledge, frequently in declarative forms, toward more efficient and accurate SDM.

To summarize, LCORPP benefits from the advantages of three SDM paradigms, namely learning, reasoning, and planning. Although each of the paradigms has a rich literature, their complementary advantages have not been exploited. To the best of our knowledge, this is the first general-purpose SDM framework that equips robots with the simultaneous capabilities of supervised learning for passive perception, automated reasoning with contextual knowledge, and active information gathering toward achieving long-term goals under uncertainty.

7.3 LCORPP Framework

In this section, we first introduce a few definitions, then focus on LCORPP, our robot SDM framework, and finally present a complete instantiation of LCORPP in detail.

Definitions: Before describing the algorithm, it is necessary to define three variable sets of $\mathbf{V}^{lrn}$, $\mathbf{V}^{rsn}$ and $\mathbf{V}^{pln}$ that are modeled in the learning, reasoning, and planning components respectively. For instance, $\mathbf{V}^{lrn}$ includes a finite set of variables:

$$\mathbf{V}^{lrn} = \{V_0^{lrn}, V_1^{lrn}, ...\}$$

We consider factored spaces, so the three variable sets can be used to specify the three state spaces respectively, i.e., S^{lrn}, S^{rsn} and S^{pln}. For instance, the learning component's state space, S^{lrn}, can be specified by $\mathbf{V}^{lrn}$, and includes a finite set of states in the form of:

$$S^{lrn} = \{s_0^{lrn}, s_1^{lrn}, ...\}$$

Building on the above definitions, we next introduce the LCORPP algorithm, followed by a complete instantiation.

7.3.1 Algorithms

We present LCORPP in the form of Algorithms 2 and 3, where Algorithm 2 calls the other.

The input of LCORPP includes the dataset Ω for sequence classification,

Algorithm 2 LCORPP

Require: A set of instance-label pairs Ω (training dataset), Logical-probabilistic rules θ, Termination probability threshold ϵ, POMDP model $\mathcal{M}$, Batch size K

Compute classifier ρ using Ω
$C \leftarrow$ CROSS-VALIDATE(ρ), where C is a confusion matrix
Update rules in θ with the probabilities in C
Compute action policy π with $\mathcal{M}$ (using POMDP solvers)
Initialize $\hat{C}$ (same shape of C) using uniform distributions
Initialize counter: $c \leftarrow 0$
while SSUB($\hat{C} - C$) $> \epsilon$ **do**
    Collect new instance I
    $L \leftarrow$ DT-CONTROL($\rho, I, \mathcal{M}$), where L is the label of I
    Store instance-label pair: $\omega \leftarrow (I, L)$
    $\Omega \leftarrow \Omega \cup \omega$
    $c \leftarrow c + 1$
    if $c == K$ **then**
        Compute classifier ρ using augmented dataset Ω
        $\hat{C} \leftarrow$ CROSS-VALIDATE(ρ)
        Update rules in θ with the probabilities in $\hat{C}$
        $C \leftarrow \hat{C}$
        $c \leftarrow 0$
    end if
end while

declarative rules θ, logical facts β, and a POMDP model $\mathcal{M}$. Each sample in Ω is a matrix of size $T \times N$, where T is the time length and N is the number of features. Each sample is associated with a label, where each label corresponds to state $s^{lrn} \in S^{lrn}$. Logical-probabilistic rules, θ, are used to represent contextual knowledge from human experts. Facts, β, are collected at runtime, e.g., current time and location, and are used together with the rules for reasoning. Finally, the POMDP model $\mathcal{M}$ includes world dynamics and is used for planning under uncertainty toward active information gathering and goal accomplishments.

Algorithm 2 starts with training classifier ρ using dataset Ω, and confusion matrix C that is generated by cross-validation. The probabilities in C are passed to the reasoner to update probabilistic rules θ. Action policy π is then generated

using the POMDP model $\mathcal{M}$ and off-the-shelf solvers from the literature. Matrix $\hat{C}$ (of shape C) and counter c are initialized with uniform distribution and 0 respectively. $\textsc{ssub}(\hat{C} - C)$ is a function that sums up the absolute values of the element-wise subtraction of matrices $\hat{C}$ and C. As long as the output of SSUB is greater than the termination threshold ϵ, the robot collects a new instance I, passes the instance along with the classifier and the model to Algorithm 3 to get label L (Lines 8-9). The pair of instance and label, ω, is added to the dataset Ω and the counter (c) is incremented. If c reaches the batch size K, new classifier ρ is trained using the augmented dataset. Then, it is cross-validated to generate $\hat{C}$, and the rules θ are updated. Also, c is set to 0 for collection of a new batch of data (Lines 13-18).

Algorithm 3 requires classifier ρ, data instance I, and POMDP model $\mathcal{M}$. The classifier (ρ) outputs the learner's current state, $s^{lrn} \in S^{lrn}$. This state is then merged into β in Line 2, which is later used for reasoning purposes. A set of variables, $\hat{\mathbf{V}}^{rsn}$, is constructed in Line 3 to form state space $\hat{S}^{rsn}$, which is a partial state space of both S^{rsn} and S^{pln}. b^{rsn} is the reasoner's posterior belief. Belief distribution $\hat{b}^{rsn}$ over $\hat{S}^{rsn}$ bridges the gap between LCORPP's reasoning and planning components: $\hat{b}^{rsn}$ is computed as a marginal distribution of the reasoner's output in Line 6; and used for generating the prior distribution of b^{pln} for active interactions. LCORPP initializes POMDP prior belief b^{pln} over the state set S^{pln} with $\hat{b}^{rsn}$ in Line 7, and uses policy π to map b^{pln} to actions. This sense-plan-act loop continues until reaching the terminal state and the estimation label would be extracted from the reporting action (described in detail in Section 7.4).

Algorithm 3 DT-CONTROL: Decision Theoretic Control

Require: Classifier ρ, Instance I, and POMDP model $\mathcal{M}$

Update state: $s^{lrn} \leftarrow \rho(I)$, where $s^{lrn} \in S^{lrn}$
Collect facts β from the world, and add s^{lrn} into β
$\hat{\mathbf{V}}^{rsn} \leftarrow \{\hat{V} | \hat{V} \in \mathbf{V}^{rsn}, \text{and } \hat{V} \in \mathbf{V}^{pln}\}$
Use $\hat{\mathbf{V}}^{rsn}$ to form the state space of $\hat{S}^{rsn}$, where $\hat{S}^{rsn} \subset S^{rsn}$ and $\hat{S}^{rsn} \subset S^{pln}$
Reason with θ and β to compute belief b^{rsn} over S^{rsn}
Compute $\hat{b}^{rsn}$ (over $\hat{S}^{rsn}$), i.e., a marginal of b^{rsn}
Initialize belief b^{pln} (over S^{pln}) using $\hat{b}^{rsn}$
repeat
    Select action $a \leftarrow \pi(b^{pln})$, and execute a
    Make observation o
    Update b^{pln} based on a and o using Bayes update rule
until reaching terminal state $term \in S^{pln}$
$L \leftarrow$ EXTRACTLABEL(a)

7.4 Instantiation

Learning component

In order to make correct state estimation based on the streaming sensor data while considering the dependencies at various time steps, we first train and evaluate the classifier ρ using dataset Ω. We split the dataset into training and test sets, and produce the confusion matrix C, which is later needed by the reasoner. Human intention estimation is modeled as a classification problem for the LSTM-based learner:

$$s^{lrn} = \rho(I)$$

where robot is aiming at estimating $s^{lrn} \in S^{lrn}$ using streaming sensor data I. In our case, I is in the form of people motion trajectories during N timesteps ($I = [x_0, y_0, ..., x_N, y_N]$); and there exists only one binary variable, *intention* $\in \mathbf{V}^{lrn}$. The trajectory instances have different trajectory lengths, in that case, we interpolate or extrapolate the trajectory to transform it the fixed size N. As a

result, state set S^{lrn} includes only two states:

$$S^{lrn} = \{s_0^{lrn}, s_1^{lrn}\}$$

where s_0^{lrn} and s_1^{lrn} correspond to the person having the intention of interacting with the robot or not. Since the human trajectories are in the form of sequence data, we use LSTM to train a classifier for estimating human intentions with motion trajectories. Details of the classifier training are presented in Section 7.5. Next, we explain how the classifier output is used for reasoning.

Reasoning component

Contextual knowledge, provided by domain experts, can help the robot make better estimations. For instance, in the early mornings of workdays, people are less likely to be interested in interacting with the robot, in comparison to the university open-house days. The main purpose of the reasoning component is to incorporate such contextual knowledge to help the passive state estimation.

The knowledge base consists of logical-probabilistic rules θ in P-log [122], a declarative language that supports the representation of (and reasoning with) both logical and probabilistic knowledge. The reasoning program consists of random variables set $\mathbf{V}^{rsn}$. It starts collecting facts and generating β. The confusion matrix generated by the classifier's cross-validation is used to update θ. The variables that are shared between the reasoning and planning components are in the set $\hat{\mathbf{V}}^{rsn}$. The reasoner produces a belief b^{rsn} over S^{rsn} via reasoning with θ and β.

In the problem of human intention estimation, the reasoner contains random

variables:

$$\mathbf{V}^{rsn} = \{\texttt{location},\ \texttt{time},\ \texttt{identity},\ \texttt{intention}, \cdots\},$$

where the range of each variable is defined as below:

```
location:  {classroom, library}

time:  {morning, afternoon, evening}

identity:  {student, professor, visitor}

intention:  {interested, not interested}
```

We further include probabilistic rules into the reasoning component. For instance, the following two rules state that the probability of a visitor showing up in the afternoon is 0.7, and the probability of a professor showing up in the library (instead of other places) is 0.1, respectively.

```
pr(time=afternoon|identity=visitor)=0.70.

pr(location=library|identity=professor)=0.10.

pr(classifier=one|intention=interested)=0.76.

pr(intention=interested|identity=visitor)=0.80.
```

It should be noted that *time* and *location* are facts that are fully observable to the robot, whereas human *identity* is a latent variable that must be inferred. Time, location, and intention are probabilistically determined by human identity. We use time and location to infer human identity, and then estimate human intention.

The binary distribution over human intention, $\hat{b}^{rsn}$, a marginal distribution of

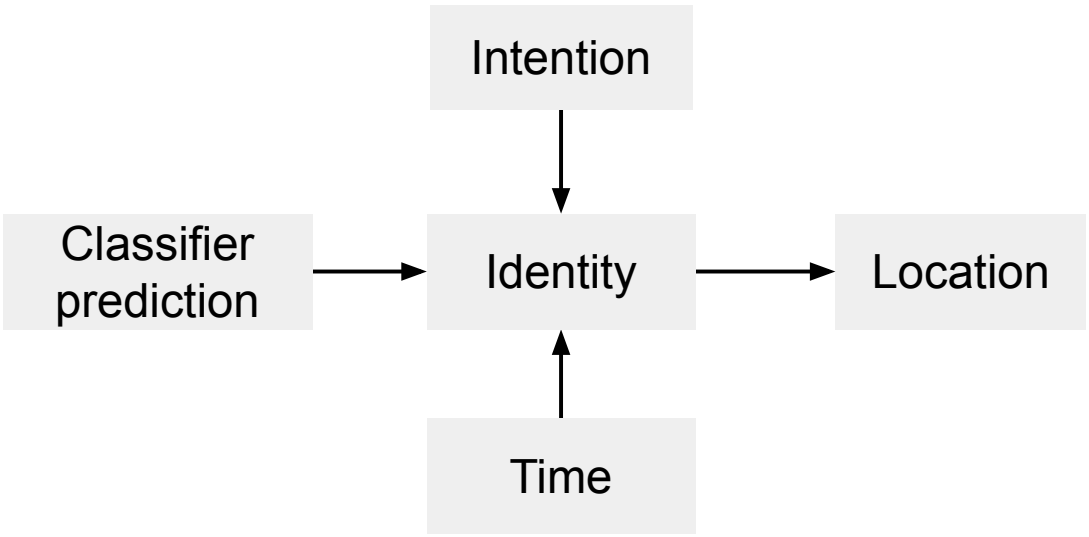

Figure 7.2: Bayesian network of the reasoner, where the robot infers human intentions based on observable facts including time, location, and classifier prediction.

b^{rsn} over $\hat{S}^{rsn}$, is provided to the POMDP-based planner as informative priors.[2]

Planning component

Robots can *actively* take actions to reach out to people and gather information. We use POMDPs to build probabilistic controllers. A POMDP model can be represented as a tuple $(S^{pln}, A, T, R, Z, O, \gamma)$. We briefly discuss how each component is used in our models:

- $S^{pln} : \hat{S}^{rsn} \times S_l^{pln} \cup \{term\}$ is the state set. $\hat{S}^{rsn}$ includes two states representing human being interested to interact or not. S_l^{pln} includes two states representing whether the robot has turned towards the human or not and $term$ is the terminal state.

- $A : A_a \cup A_r$ is the set of actions. A_a includes both motion-based and

[2]The reasoning component can be constructed using other logical-probabilistic paradigms that build on first-order logic, such as Probabilistic Soft Logic (PSL) [123] and Markov Logic Network (MLN) [124]. In comparison, P-log directly takes probabilistic, declarative knowledge as the input, instead of learning weights with data, and meets our need of utilizing human knowledge in declarative forms.

language-based interaction actions, including *turning* (towards the human), *greeting*, and *moving forward* slightly. A_r includes two actions for reporting the human being interested in interaction or not.

- $T(s, a, s') = P(s'|s, a)$ is the transition function that accounts for uncertain or non-deterministic action outcomes where $a \in A$ and $s \in S$. Reporting actions deterministically lead to the *term* state.

- $Z = \{pos, neg, none\}$ is the observation set modeling human feedback in human-robot interaction.

- $O(s', a, z) = P(z|a, s')$, where $z \in Z$, is the observation function, which is used for modeling people's noisy feedback to the robot's interaction actions.

- $R(s, a)$ is the reward function, where costs of interaction actions, $a_a \in A_a$, correspond to the completion time. A correct (wrong) estimation yields a big bonus (penalty).

Reporting actions deterministically lead to the *term* state. We use a discount factor $\gamma = 0.99$ to give the robot a long planning horizon. Using an off-the-shelf solver (e.g., [11]), the robot can generate a behavioral policy that maps its belief state to an action toward efficiently and accurately estimating human intentions.

To summarize, the robot's LSTM-based classifier estimates human intention based on the human trajectories. The reasoner uses human knowledge to compute a distribution of human intention. The reasoner's intention estimation serves as the prior of the POMDP-based planner, which enables the robot to actively interact with people to figure out their intention. The reasoning and planning components of CORPP are constructed using human knowledge, and do not involve

learning. The reasoning component aims at correcting and refining the LSTM-based classifier's output, and the planning component is for active perception.

7.5 Experimental Results

We did pairwise comparisons between LCORPP with the following methods for human intention estimation to investigate several hypotheses. Our baselines include: **Learning (L)**: learned classifier only. **Reasoning (R)**: reasoning with contextual knowledge. **Planning (P)**: POMDP-based interaction with uniform priors. **Learning+Reasoning (L+R)**: reasoning with the classifier's output and knowledge. **Reasoning+Planning (R+P)**: reasoning with knowledge and planning with POMDPs.

Our experiments are designed to evaluate the following hypotheses. I) Given that LCORPP's prior belief is generated using reasoner and learner, it would outperform baselines in intention estimation in F1 score while receiving less action costs; II) In case of inaccurate knowledge, or III) the learner not having a complete sensor input, LCORPP's planner can compensate these imperfections by taking more actions to maintain higher F1 score. IV) In case of scarcity of data, the robot's most recent interaction can be used to augment the dataset and improve the overall performance of LCORPP.

In each simulated trial, we first sample human identity randomly, and then sample time and location accordingly, using contextual knowledge, such as professors tend to show up early. According to the time, location, and identity, we sample human intention. Finally, we sample a trajectory from the test set of the dataset, according to the previously sampled human intention. We added

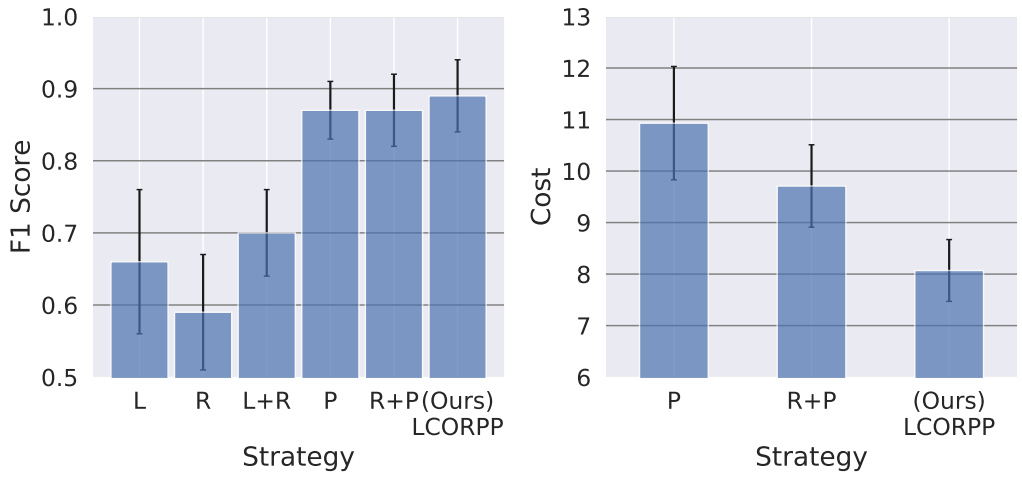

Figure 7.3: Pairwise comparisons of LCORPP with five baseline sequential decision-making strategies. The right subfigure excludes the strategies that do not support active human-robot interaction and hence produce zero costs.

30% noise to the human reactions (robot's observation) being compliant with the ground truth but independent of the robot's actions. LCORPP requires considerable computation time for training the classifier (∼10 min) and generating the POMDP-based action policy (∼1 min). The training and policy generation are conducted offline, so they do not affect the runtime efficiency. Reasoning occurs at runtime, and typically requires less than 1 millisecond.

LCORPP vs. Five Baselines: Figure 7.3 shows the overall comparisons using the six SDM strategies, each number is an average of 5000 trials, where the setting is the same in all following experiments. The three strategies, **P**, **R+P** and LCORPP, include the POMDP-based planning component, and perform better than no-planning baselines in F1 score. Among the planning strategies, ours produces the best overall performance in F1 score, while reducing the interaction costs (dialog-based and motion-based)(Hypothesis I).

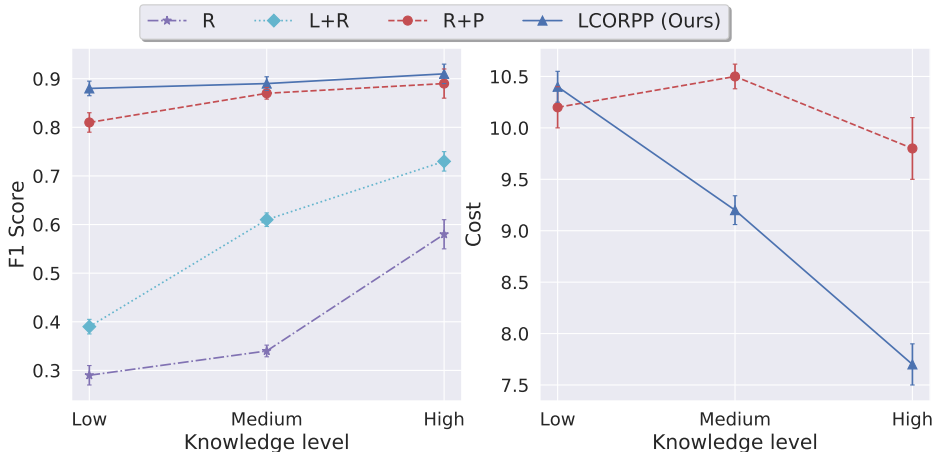

Figure 7.4: Performances of LCORPP and baselines given contextual knowledge of different accuracy levels: High, Medium, and Low. Baselines that do not support reasoning with human knowledge are not included in this experiment.

Inaccurate Knowledge: In this experiment, we evaluate the robustness of LCORPP to inaccurate knowledge (Hypothesis II). Our hypothesis is that, in case of contextual knowledge being inaccurate, LCORPP is capable of recovering via actively interacting with people. We used knowledge bases (KBs) of different accuracy levels: **High**, **Medium**, and **Low**. A high-accuracy KB corresponds to the ground truth. Medium- and low-accuracy KBs are incomplete, and misleading respectively. For instance, low-accuracy knowledge suggests that *professors* are more likely to interact with the robot, whereas *visitors* are not, which is opposite to the ground truth. Figure 7.4 shows the results where we see the performances of **R** and **L+R** baseline strategies drop to lower than 0.4 in F1 score, when the contextual knowledge is of low accuracy. In F1 score, neither **R+P** nor LCORPP is sensitive to low-accuracy knowledge, while LCORPP performs consistently better than **R+P**. In particular, when the knowledge is of low accuracy, LCORPP retains the high F1 score (whereas **R+P** could not) due to its learning component.

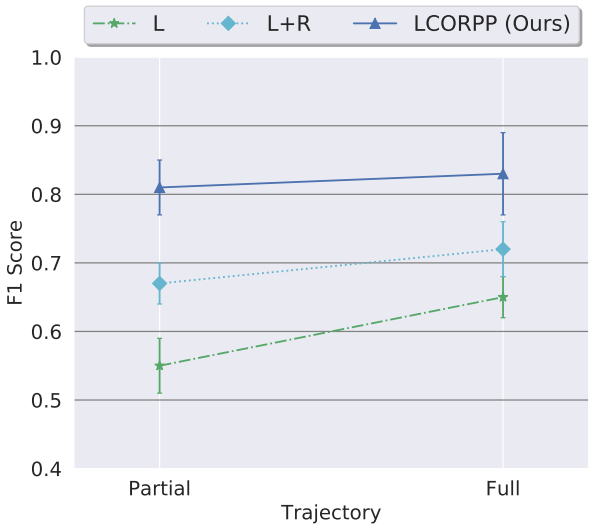

Figure 7.5: Experiments with incomplete sensor input (trajectory), in comparison to the two "learning-included" baselines.

Partial Motion Trajectories: Within the human intention estimation context, the robot's goal is to identify human intention as accurately and early as possible. However, there is the trade-off between efficiency and accuracy. For instance, reaching out to people early usually means the robot only has a short piece of human motion trajectory, and frequently produces less accurate intention estimation results. In this experiment, we evaluate how sensitive our platform is to partial human motion trajectories (Hypothesis III). The experimental results can provide robot practitioners a reference on how early a modern robot can produce reasonably accurate intention estimation results.

We only provided part of each trajectory to the classifier (the first quarter in our case). The goal is to evaluate, if the robot is forced to produce the estimation result using only 1/4 piece of the trajectories, how reliable the three "learning-included" strategies are, where "L" includes only the LSTM-based classifiers, "L+R" further combines the reasoner, and LCORPP is our approach.

The results are presented in Figure 7.5. We can see, in the case of partial

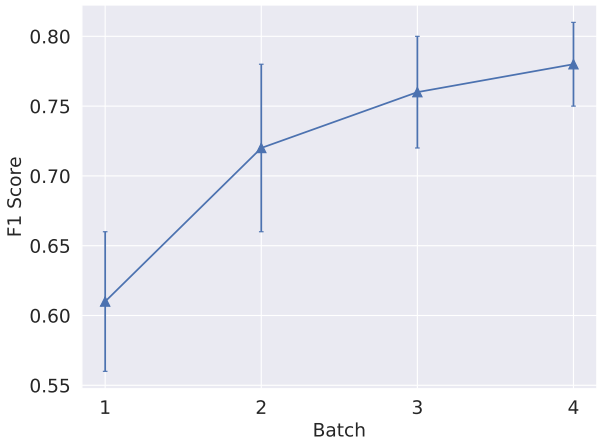

Figure 7.6: Active data augmentation from the experience of human-robot interaction.

trajectories, LCORPP is able to retain its human intention estimation accuracy in F1 score. We attribute this achievement to the interaction capability of LCORPP based on experimental results (Figure 7.5) shown in this work. The results support Hypothesis III, which states that when the sensor input is incomplete, LCORPP can take more actions to refine the likely inaccurate learner's state estimation.

Active Data Augmentation: In this experiment, we explored the effectiveness of augmenting the dataset (for supervised learning) using the experience of human-robot interaction, including trajectories and corresponding labels generated by the task planner. Our hypothesis is that, as more trajectories are collected and added into the dataset, our robot is able to re-train the classifier, and produce increasingly better overall performance in F1 score (Hypothesis IV).

Instead of training the initial classifier using all data, we divided the dataset into four batches, each containing 500 instances. In the first batch, we ran LCORPP without offering any training data. As a result, the "supervised learning" component produced a random classifier. In the subsequent batches, the robot

interacts with a human using the interaction instances collected from early bathes. Figure 7.6 shows the results. As the robot becomes more experienced (i.e., more instances of human-robot interaction trials become available), LCORPP performs better in terms of F1 score.

7.6 Summary of SDM with Commonsense Reasoning

In this chapter, we develop a robot sequential decision making framework, called LCORPP, that integrates supervised learning for passive state estimation, automated reasoning for incorporating declarative contextual knowledge, and probabilistic planning for active perception and task completions. LCORPP has been applied to a human intention estimation problem using a mobile robot.

Results suggest that the integration of supervised deep learning, logical-probabilistic reasoning, and probabilistic planning enables simultaneous passive and active state estimation, producing the best overall performance in efficiency and accuracy while conducting human intention estimation tasks. We also demonstrated that, in case of scarcity of data, LCORPP enables the robot to use its most recent interaction to augment the dataset to further improve the overall performance.

8 SDM WITH GRAPH-BASED REASONING

In this chapter we investigate how to leverage scene graph methods [125] to address curse of dimensionality in tasks that require reasoning about many objects. this aims to improve the scale of knowledge (compared with our previous work) toward reasoning about large open-source knowledge bases for robot decision making under uncertainty.

8.1 Introduction

There has been great progress in the development of service robots in recent years (e.g., STRANDS and BWI project robots [3, 1]). Those robots are able to conduct everyday tasks in human-inhabited environments over extended periods of time. Robot perception in such domains is partial and unreliable, which brings a major challenge to robot decision making.

Partially Observable Markov Decision Process (POMDP) is a framework that models the uncertainty in both observations and action outcomes [10], and has been used for policy generation in partially observable domains. However, the challenges are two-fold. First, constructing POMDPs requires that the robot has a complete world model, which tends to be infeasible in practice. In particular, real-world environments (say a kitchen) frequently include many objects, making

it troublesome to use POMDPs to have a universal representation of all objects. Second, the complexity of reasoning about these objects and their relationships grows exponentially as more objects are considered. In this dissertation, we aim to develop an approach that reasons with contextual information for scene analysis to enable POMDP-based robot planning.

One of the recent advancements in computer vision has been scene graph generation networks [126, 127, 128, 125, 129, 130, 131]. Given an image, scene graph systems generate a graph consisting of detected objects (e.g, a book and a table), their corresponding bounding boxes, and the relationships among the objects (e.g., *book on a table*). Scene graphs provide a robot with a structured understanding of the world in terms of objects, and their relations. From the robotics perspective, however, current scene graph research has the limitation that the context analysis does not go beyond individual images, even though a robot can easily capture images from different angles and locations for analysis purposes. With the *active* perception capabilities of robots, we have the objective of developing an approach for domain-wide active scene analysis for mobile robots.

In this chapter, we propose an algorithm called *scene analysis for robot planning* (SARP) for planning robot actions for context-aware, object-centric scene analysis [132]. SARP is going to use local scene graphs of single images to build and augment global scene graphs toward context-aware robot planning under partial observability. An overview of SARP is shown in Figure 8.1. More specifically, a global scene graph will be incrementally constructed "on the fly" using local scene graphs generated at different locations when new objects are perceived. Reasoning with this global scene graph produces useful information to help the robot esti-

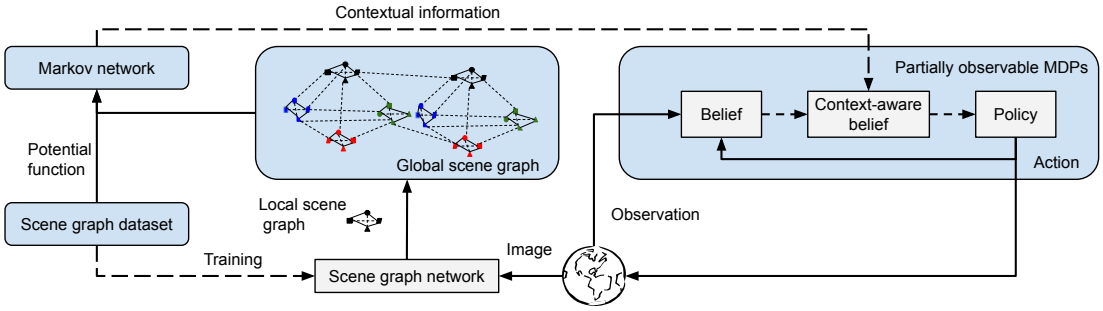

Figure 8.1: An overview of SARP, where the robot takes an action using the policy, and receives an observation from the world, updating the belief using the action and observation. In each timestep, a graph is accumulated using the perception sensor to bias the belief. This biased belief provides contextual information to the policy. Top dashed lines show that the belief biasing does not occur in every iteration. The bottom dashed line indicates that the scene graph networked is trained offline.

mate the current world state. This enhanced state estimation enables the robot to improve its performance in goal achievement.

We have evaluated SARP using target search tasks where a robot needs to locate an object in an indoor environment. We use POMDPs to model the robot's perception and actuation skills [10], use Neural Motifs [126] to compute local scene graphs, and use approximate inference methods to build Markov networks computed from large datasets. We have extensively evaluated SARP through comparisons with competitive baselines in simulation. Results show that SARP reduced the overall action costs by 16% compared with a predefined action policy. Also, SARP helps the robot maintain its policy quality in the presence of an increased number of objects, and enables the robot to focus on the areas that are most relevant to the current task.

8.2 Related Work

This work aims to enable a robot to represent and reason with contextual information to guide robot planning under uncertainty. Researchers have developed algorithms that reason with contextual knowledge to guide sequential decision making [26]. The contextual knowledge can be in a variety of forms, such as commonsense knowledge [133], action knowledge [134, 135], and graph-based knowledge [136, 125]. Such contextual knowledge can also be leveraged to guide RL agents. In this section, we examine each of these categories.

Rule-based Human Commonsense Knowledge Researchers have used rule-based commonsense knowledge to guide the robot planning under uncertainty [36, 37, 38]. In their methods, a robot reasons about human knowledge to compute an informative prior to help a probabilistic planner estimate the current world state, enabling the robot to achieve complex goals with less information-gathering behaviors. Another example is an algorithm that uses first-order logic to construct decision tree policies for goal-oriented factored POMDPs [45]. Others have used action knowledge to build a hierarchical robot planner where the higher level computes a sequence of abstract actions and the lower level implements the higher-level actions using primitive behaviors [35]. Algorithm iCORPP enables a robot to reason with contextual knowledge to compute parameters of a planning agent's reward and transition functions [46]. Others have used commonsense knowledge to guide a classical planner to reason and plan in open worlds [47, 48]. Compared to those methods, SARP uses contextual, object-centric information, in the form of a graph, to estimate the current world state and guide robot planning under

uncertainty.

Hierarchical frameworks Researchers have been using hierarchies to construct their framework's knowledge-base or planner [35, 49]. In a recent work, researchers used a visual hierarchical planning algorithm for long-horizon manipulation tasks. Their framework integrates neuro-symbolic task planning and graph- based motion generation on graph-based scene representations [49]. Their method has two-level abstractions of a manipulation scene with geometric scene graphs and symbolic scene graphs. To enable a robot to operate in open-world domains, researchers have developed a three-layer hierarchy for reasoning about action knowledge and default knowledge [41]. In particular, the knowledge at a higher level was used for correcting lower-level knowledge to guide probabilistic planning. In comparison, SARP (ours) incrementally builds and reasons about scene graphs to guide robot planning under uncertainty.

Graph-based human knowledge Graph-based representations have been used for reasoning with contextual information to guide robot planning [49, 50, 51]. For instance, researchers have used Conditional Random Fields [52] to construct contextual knowledge bases for robot target search by maintaining a belief over the locations of target objects and landmark objects while exploiting the knowledge of their co-appearances [50].

Other methods were developed to extract prior knowledge from data, e.g., using Long Short-Term Memory (LSTM) networks [22], to guide a planning agent in navigating new environments [51]. In comparison to those methods, SARP leverages contextual information in the form of automatically constructed scene

graphs, avoiding domain experts manually developing knowledge bases.

POMDP-based target search Several reasearchers have used POMDPs for the target search task [137, 138, 139, 140]. In one work, a robot arm is tasked to search for an object in the clutter where it should learn the synergies of acting and seeing objects. In this chapter, the policy learned by a POMDP model could determine when is a good time to look at and detect objects and when it should move objects around in order to reduce the clutter [139]. In another work, a robot searched for multiple objects by specifiying their locations from the user query (e.g., *Find the mugs in the kitchen and books in the library*) [138]. In their framework, a robot can associate the locations to each object class so as to improve its search. To the best of our knowledge, none of these works leverage graph reasoning for target search tasks.

Knowledge-based RL RL agents are able to leverage contextual information as well. For instance, existing research has shown that an RL agent is able to reason with action knowledge to decompose complex tasks into smaller, tractable subtasks [28]. The concept of reward machines has been introduced for using temporal knowledge toward reusing interaction experience and creating extra feedback for learning purposes [53]. Recent research has shown that action knowledge can be used to guide a model-based RL agent by providing an optimistic, artificial interaction experience to speed up the learning process [33, 34]. We assume the availability of the world dynamics, and use planning under uncertainty methods (instead of RL) for robot decision making. To the best of our knowledge, SARP is the first that enables a planning agent to automatically construct, and use scene

graphs for context analysis under partial observability.

8.3 SARP Algorithm

Problem Formulation In this work, we are interested in the problem of target search. A mobile robot receives the task of searching for object Q and can navigate in the environment E in order to find the object. The robot is provided with the environment map and is localized initially. Once the robot navigates in the environment sufficiently, it reports the location of the target object. In order for the robot to better reason about Q, robot requires the scene graph network that is pretrained on a scene graph dataset $\mathcal{D}$.

POMDP model In order to solve this target search task, we first define the POMDP model $\mathcal{M}$ as the tuple $< S, A, T, R, O, Z, \gamma >$. Its factored state space set S, is a Cartesian product of two dimensions, and the terminal state. S^E includes a set of discrete partially observable locations of Q (the target object) and S^R includes the set of robot's fully observable locations. We define these equidistant, discrete locations manually in the robot's motion planning workspace. SARP's action set A consists of navigation and termination actions. The robot can take navigation action $go_i \in A$ to go to the location i. We model the action transition function $T(s'|s, a)$ so that the robot can only go to its closest neighboring locations in the absence of obstacles. Each navigation action has a cost (negative value, $R(s, a) < 0$), which is proportional to the distance the robot needs to travel to reach that location. The robot can take the termination action, and receive bonus (penalty) if it correctly locates (not find) the target object. Observation set Z is {*Detected*, *NotDetected*, *NotApplicable*} where *Detected* happens when the

perception detects the query object, *NotDetected* happens when the query object is not detected, and *NotApplicable* when the agent takes the termination action. To maximize its planning horizon, we set γ to 0.99. SARP assigns the observation function $O = pr(z|s, a)$ based on the target's object detection accuracy on a test set of $\mathcal{D}$.

Algorithm Description After defining the problem and the POMDP model, we present our novel algorithm, called *scene analysis for robot planning* (SARP), for context-aware robot planning under partial observability. SARP computes a policy that enables a robot to accomplish its task using less action cost using the contextual knowledge. SARP is an object-centric algorithm that bridges the representation gap between visual scene analysis and robot planning under partial observability.

Algorithm 4 presents SARP, whose input is an object of interest that a mobile robot needs to locate (Q). SARP requires a POMDP solver for policy generation, a pre-trained scene graph generation network, a domain map for navigation, and a dataset $\mathcal{D}$ that consists of scene graphs. In Line 1, SARP constructs a POMDP according to the object of interest (Q), and computes policy π using the provided POMDP solver. After that, there are a few steps for initializing beliefs (over object locations), and a scene graph (Lines 2-4). It should be noted that we maintain two beliefs b and b' over the location of the target object, where b is updated using POMDP observations, and b' is updated using the current b and available contextual information. This design allows the robot to use different beliefs for decision making and action selection, and avoids possible issues caused by error propagation. Also, this mechanism enables a SARP agent to avoid reusing

contextual information in belief updates. This mechanism enables a SARP agent to avoid reusing contextual information in belief updates.

Lines 5-20 form the main control loop of SARP, and it terminates when the current state is a terminal state. In each iteration, the robot uses a captured image to generate G^L, a local scene graph (Line 6), and uses this local scene graph to update G^G, the global scene graph (Line 8).

After that, SARP computes a potential function ϕ (Line 9) values for each edge, using the function $CALC\phi$ (Algorithm 5). This function calculates four values for each relation (v, ϵ, v') in G^L that serve as the potential function. $CALC\phi$ queries the dataset $\mathcal{D}$, to find out the ratio of the times that each object (v, v' or both) in a relation has appeared (not appeared) in $\mathcal{D}$, resulting in four values to form the potential function.

We use ϕ to form a Markov network together with global scene graph G in Line 10 where SARP queries the number of times that each pair of nodes of an edge in G^G has appeared (not appeared) in $\mathcal{D}$ scene graphs.

In our implementation, $\mathcal{N}$ is incrementally updated in each iteration, if the robot detects new scene graphs (that were not previously detected), it will add the new nodes and relationships to the existing global scene graph G^G. We use the robot's localization and the camera depth sensor to approximately localize the detected objects on the map, in order to distinguish different instances of the same objects.

Lines 11-19 correspond to the belief update, and action selection processes of POMDPs. Contextual knowledge in the form of a Markov network ($\mathcal{N}$) is used for biasing belief b only if the object of interest is visually detected in the current

Algorithm 4 SARP

Input: Query object Q

Require: a POMDP solver, a scene graph generation network, a domain occupancy grid map, robot localization software, scene graph dataset $\mathcal{D}$.

1: Construct a POMDP based on Q, and compute policy π
2: Uniformly initialize beliefs b and b' over S^E
3: Initialize vertices $V = \{Q, Q'\}$ and edges $\mathcal{E} = \{\epsilon_{Q-Q'}\}$, where Q' is a duplicate of Q
4: Initialize a scene graph: $G \leftarrow (V, \mathcal{E})$
5: **while** current state is not terminal **do**
6: Take an image, and generate local scene graph $G^L = (V', \mathcal{E}')$, where V' are detected evidence objects
7: $V \leftarrow V \cup V'$; $\mathcal{E} \leftarrow \mathcal{E} \cup \mathcal{E}'$
8: $G^G \leftarrow (V, \mathcal{E})$
9: Compute potential function $\phi = CALC\,\phi(G^L, \mathcal{D})$
10: Form a Markov network: $\mathcal{N} \leftarrow (G^G, \phi)$
11: Select action $a \leftarrow \pi(b')$, and execute a
12: Make observation z on Q
13: Update b based on a and z
14: **if** z is *Detected* **then**
15: $pr(Q|V) \leftarrow BeliefPropagation(\mathcal{N}, Q, V)$
16: $b' \leftarrow \eta \cdot pr(Q|V) \cdot b$
17: **else if** z is *NotDetected* **then**
18: $b' \leftarrow b$
19: **end if**
20: **end while**

image. This is because of avoiding the bias drift that could result from too many biasing at every timestep. We use a *belief propagation*[1] method [141] to compute the probability of Q being collocated with objects V (Line 15).

$$pr(Q|V) = BeliefPropagation(\mathcal{N}, Q, V)$$

where the computations of $pr(Q|V)$ and b are independent. In Line 16, SARP uses $pr(Q|G)$ to compute b', the posterior belief distribution.

CALC function CALC function that takes as input the scene graph dataset $\mathcal{D}$ and the local scene graph G^L. For each pair of nodes v and v' that are connected

[1]Any approximate inference method can be used.

Algorithm 5 $CALC_\phi$

Input: Scene graph dataset $\mathcal{D}$, local scene graph G^L

Initialize the potential function set ϕ as empty
for $(v, \epsilon, v')^i$ in G^L **do**
  $m \leftarrow$ number of images in $\mathcal{D}$ containing v or v'.
  $\phi^i_{v^1,v'^1} \leftarrow$ (number of images in $\mathcal{D}$ containing v, v', and ϵ)$/m$
  $\phi^i_{v^0,v'^0} \leftarrow$ (number of images in $\mathcal{D}$ containing v, v', but not ϵ)$/m$
  $\phi^i_{v^0,v'^1} \leftarrow$ (number of images in $\mathcal{D}$ containing v' without v)$/m$
  $\phi^i_{v^1,v'^0} \leftarrow$ (number of images in $\mathcal{D}$ containing v without v')$/m$
  $\phi \leftarrow \phi \cup \phi^i$
end for
return ϕ

in the ith edge ϵ, the algorithm calculates four values by querying the number of times each node and their corresponding edge have (not) appeared in the dataset $\mathcal{D}$. We denote v^1 as the node exists in $\mathcal{D}$ and v^0 when it does not exist. It returns the computed potential function ϕ.

SARP enables the agent to leverage contextual information towards task completion through building and reasoning with global scene graphs to guide a probabilistic planner. Next, we discuss our experiments for evaluating SARP.

8.4 Experiments

We conducted two sets of experiments in simulation where the robot is tasked with finding a target object accurately and as quickly as possible. In all the trials, the robot is provided with a domain map, a dataset of scene graphs, and a pretrained network for generating scene graphs. It receives 360-degree images of the environment as the input, and the output is the location of the target object. Our first baseline method is a naive POMDP planner [139] (with uniform prior belief) where the robot action policy solely depends on the model of the world. The second baseline uses a predefined policy where the robot exhaustively

visits all discrete positions, updates the belief at each timestep, and reports the object's location based on the argmax of the belief. Our third baseline method is CORPP [36] as another competitive baseline where **only** the initial belief is biased based on the commonsense rules defined by the human developer in the form of logical probabilistic rules (e.g., *a book is likely to be on a desk with* 0.8 *probability*). Similar to SARP, all the baselines maintain a belief of the object's location and update it at each timestep using Bayes update rule. However, none of them use graph reasoning like SARP. We have two evaluation metrics. First, the average action cost that represent the average execution time of all actions taken until the terminal state. Second, is the average success rate in finding the target object's location correctly. By using SARP, we hypothesize that:

1. The robot's overall action cost would be less compared to baselines (H-1).

2. SARP performs better than the baselines in action cost and success rate in domains with a large number of objects (H-2).

The reward of successfully finding the target object is 100, and the penalty of failure in finding the target object is -100. The reward for all actions go_i is -10 which is proportional to the time it takes for the robot to execute the action. We solve the POMDP model using an off-the-shelf point-based system [11].

We use Neural Motif [126] for generating local scene graphs where there are a total of 50 predicate classes. The most prevalent objects in this dataset are humans and the most appeared predicates are *in*, *on*, and *belongs*. Given an input image, Neural Motifs produces a scene graph which is a list of objects, their probabilities, relationships and bounding boxes. SARP requires a dataset to assign the Markov network potential function (Line 9 in Algorithm 4). We are

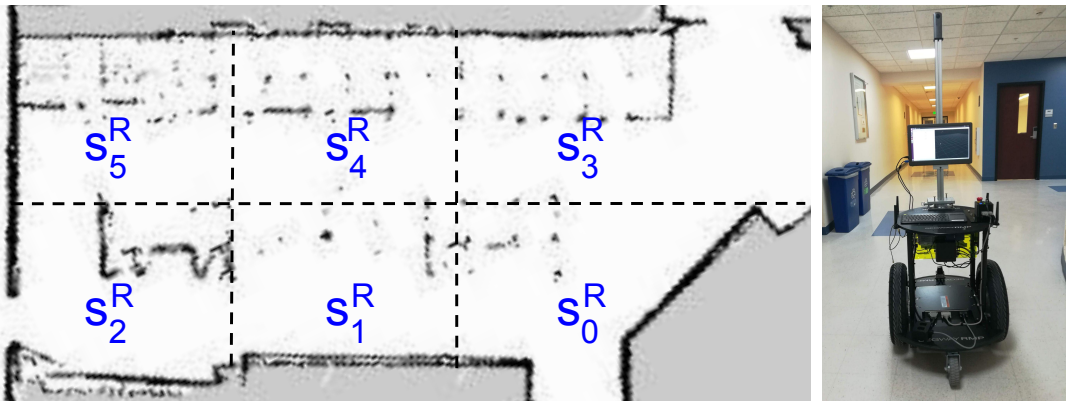

Figure 8.2: (**Left**): Map of the environment where the robot can navigate. The discretized locations are shown in blue color. (**Right**): The Segway RMP110 mobile platform, equipped with 360-degree vision, used in this research.

using Visual Genome [142] that contains round 108K images, 3.8M objects, 2.8M relationships.

8.4.1 Setup

We used the robot to navigate through the environment and collect images in a hallway in an indoor educational environment (Figure 8.2) in order to build a dataset. We call it the *hybrid* dataset. We used this dataset to simulate robot's behavior in the experiments. We manually placed multiple objects including *banana, laptop, human, books, mug, etc.* at different locations.

In addition to the dataset collected by the robot, we collected rendered images using an embedded agent in AI2THOR [143], an open-source interactive environment for embodied AI. AI2THOR provides 30 instances of four types of environments (shown in Figure 8.4): *Kitchens*, *Living Rooms*, *Bedrooms*, and *Bathrooms*, totaling 120 environments where an embedded robot is able to take navigation actions. Navigation actions include moving and rotating in orthogonal directions. To make both datasets consistent, we manually create 360 images using the AI2THOR

Table 8.1: The experimental results from the hybrid and rendered datasets show that on average of 1500 trials, SARP performs with less overall action costs. Bold font shows the best result. The average number indicates the average number of objects involved in a single episode conducted in that environment.

Dataset Type	Average # of objects	**SARP (Ours)**		CORPP		Uniform POMDP [139]		Predefined	
		Cost (std.)	Success	Cost (std.)	Success	Cost (std.)	Success	Cost (std.)	Success
Hybrid	15	**52.1 (13.1)**	**0.89**	82.5 (8.9)	0.83	102.3 (14.6)	0.84	61.1 (0.0)	0.7
Rendered									
Kitchen	70	**41.7 (21.6)**	0.87	55.5 (18.4)	0.84	59.4 (24.1)	0.82	51.3 (0)	0.79
Living room	37	**43.5 (19.1)**	**0.91**	60.1 (33.8)	0.86	69.7 (23.9)	0.73	57.8 (0)	0.79
Bathroom	40	31.2 (16.4)	**0.75**	45.1 (17.6)	0.73	49.8 (21.7)	0.71	**21.2** (0)	0.69
Bedroom	35	23.9 (12.9)	**0.74**	24.4 (19.1)	0.75	43.7 (22.5)	0.68	**19.4** (0)	0.64
Overall	45	**35.7 (18.1)**	**0.81**	46.3 (21.5)	0.79	55.7 (23.0)	0.73	37.3 (0)	0.72

monocular camera. We use the default value of $1.5m$ for the camera visibility and 90 degrees for the field of view. The actions executions are stochastic with the default Gaussian noise of average 0.001 and standard deviation of 0.005. There are a total of 125 objects in AI2THOR platform[2], while for each scene in the platform, there are 61 number of objects on average. We randomly select one of those objects for each individual trial in the experiments. For inference on the scene graph, we use pgmpy library [144].

8.4.2 Results

Table 8.1 shows the result of the first set of experiments that evaluates the first hypothesis (H-1) where we ran this experiment 1500 times over three batched of 500 experiments. We ran the experiment both using the dataset collected by

[2]https://ai2thor.allenai.org/ithor/documentation/objects/object-types/

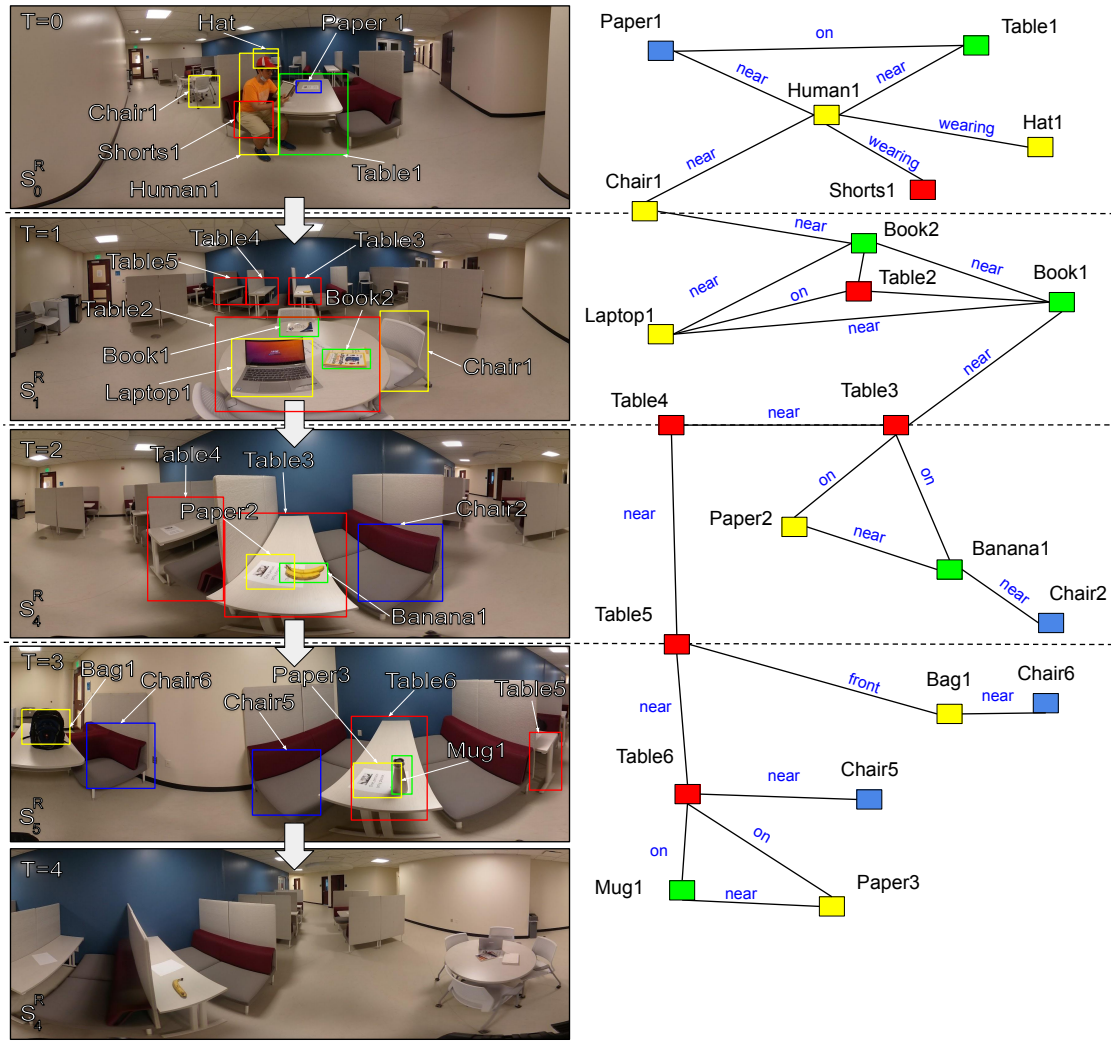

Figure 8.3: The global scene graph being constructed incrementally as the robot navigates. Solid lines indicate the relationships between the detected objects, and dashed lines show how the global scene graph is completed from the local scene graphs. At timestep 4, the global scene graph is not augmented, since the robot has not detected new instances of objects.

the real robot and by AI2THOR. We call the first one, the *hybrid* dataset and the latter one, the *rendered* dataset. In the hybrid dataset, the average cost of target search for our robot is consistently less than all the baselines while all the methods maintain a high success rate. In the rendered dataset, we categorize the results based on the four different types of indoor environments: *kitchen*, *living room*, *bedroom* and *bathroom*. We randomly selected five environment from each of the four types, totaling 20 different environments. In each environment, we conducted

Figure 8.4: Different types of environment provided by AI2THOR including *kitchen*, *bathroom*, *bedroom*, and *living room*.

100 target searches. Except for the *bedroom* and *bathroom* environments where the predefined policy has lower cost, SARP produced the lowest average action cost while maintaining the highest accuracy consistently. The reason is that, in *bedroom* and *bathroom* environments, robot has a smaller navigation area. As a result, it takes less action costs using the predefined policy. To evaluate the second hypothesis (H-2), we incrementally added more objects to the POMDP baseline, and to the scene graph of SARP. Figure 8.5, shows that SARP's average cost remains almost the same with an average of 50.6 s while the POMDP baseline's cost increases as the number of additional objects is increased. With an increase in the number of non-target objects, the uniform POMDP baseline tries to take more

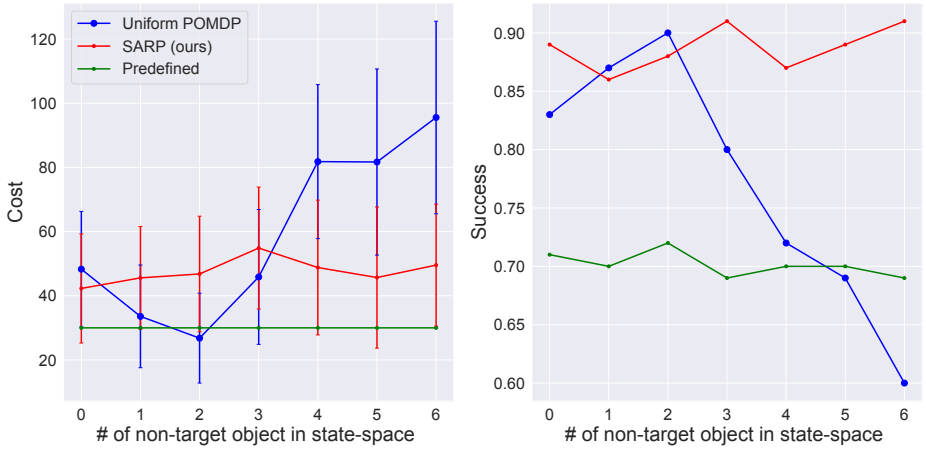

Figure 8.5: Results of the experiment conducted on the renderred dataset. The state space consists of three locations, blue curve is the baseline that does not use contextual information but includes non-target objects in its state space incrementally, and the orange one is SARP that adds objects to the graph incrementally.

actions to find the target object with more confidence, however with more than two non-target objects, it finds that taking more actions is not helpful any more, and therefore is not successful. Also, the POMDP baseline failed to maintain the success rate at a high value due to its poor-quality policy.

Enumerating Objects of the Same Instance As the robot is navigating the environment, it may perceive different instances of the same objects. Our scene graph network is not able to enumerate these objects. To better differentiate these instances, we use a hashmap to store the labels and location(s) of the detected objects. This facilitates the situations where multiple instances of the same objects need to be distinguished. We approximate the detected objects' locations using the centroid of the bounding boxes outputted by the scene graph. This provides the relative location of the object with respect to the robot. Then, we use coordinates transformation to get the object global scene graph. For the real

Table 8.2: An example trial where the robot is tasked with the search of a *banana* located in s_4^E while robot is initially at s_3^R .Here, we show how the belief is updated at each timestep.

Step	**A:** Action **O:** Observe	Robot's belief of the target $[s_0^E\ ,\ s_1^E\ ,s_2^E\ ,\ s_3^E\ ,s_4^E,s_5^E\ ]$
0	A: go $l1$ O: No Update: Bias:No	$[0.18, 0.09, 0.18, 0.18, 0.18, 0.18]$
1	A: go $l4$ O: Yes Update: Bias:**Yes**	$[0.15, 0.07, 0.15, 0.15, 0.33, 0.15]$ $[0.12, 0.06, 0.11, 0.12, 0.47, 0.12]$
2	A: go $l5$ O: No Update: Bias:No	$[0.13, 0.05, 0.13, 0.14, 0.49, 0.07]$
3	A: go $l4$ O: Yes Update: Bias: **Yes**	$[0.09, 0.03, 0.07, 0.10, 0.68, 0.05]$ $[0.07, 0.01, 0.04, 0.06, 0.81, 0.04]$
4	A: terminate	

robot experiments, we use the "Robot Operating System (ROS) [25] transforms" package to transform the locations to global coordinates. It should be noted that vision-based object association is generally difficult, and introduces errors into our system, which is beyond the scope of this dissertation.

8.5 Demonstration

In this demonstration trial, the robot was assigned the task of searching for a *banana* as shown in Figure 8.3 while its initial location is in location 0. In Table 8.2, we show an example trial where the robot is tasked with the search of a banana located at s_4^E. As the robot takes action suggested by the policy, it updates its belief over possible locations of the target object. In this example, the robot follows the trajectory of $0 \rightarrow 1 \rightarrow 4 \rightarrow 5 \rightarrow 4$ until it terminates the trial

and reports successfully that banana is in location 4. The robot visits locations $0, 1, 4$, and 5 to build and augment its scene graph. Figure 8.3 shows the graph generated at each location. At timesteps 1 and 3, robot visits the location that it can easily detect the banana, therefore it biases its belief by inferring the whole scene graph. The overall cost for this trial is 60.

8.6 Summary of SDM with Graph-based Reasoning

Probabilistic planning methods under partial observability allow the robot to accomplish complex tasks toward maximizing long-term goal. Scene graphs allow the detection of objects and their relationships in images. Aiming at robots capable of accomplishing sophisticated tasks, we design SARP framework where a robot can use scene understanding information to obtain domain knowledge to provide contextual information to the robot planner where limited perception is a bottleneck. Results show that, by using our approach, the robot can benefit from the contextual information in the form of graph network by reducing action collection cost and avoiding scalability issues.

This work was based on a few assumptions that are sometimes unrealistic. The object detection accuracy and the set of are all limited to the scene graph generation model quality. The scene graph we use, is incapable of face recognition, therefore we assume that humans are stationary (e.g, *they are sitting at the time of the robot's task execution*).

9 CONCLUSION

Human knowledge is available in many practical domains and can be leveraged by service robots in order to do complex tasks. This dissertation introduces novel ways of integrating various forms of knowledge in robot sequential decision-making under uncertainty.

We first investigated how robots can leverage data of different sensor modalities in order to find out what actions can help the robot identify object properties with the highest accuracy while minimizing the object exploration cost. We observed that our approach was able to outperform predefined and random baselines in both reward and accuracy.

In another work, we investigated how the robot can deal with knowledge base lack of sufficient entities to reason and plan. We proposed an approach in a robotics spoken dialogue system that interacts with humans in order to provide delivery services. Our dual-track POMDP controller is able to make decisions on what to ask the human as well to decide whether it needs to augment its knowledge base or not. We used the fluctuation in the robot's belief entropy to find out when the robot needs to augment its knowledge base.

Additionally, we studied how both reasoning and planning can assist the robot in accomplishing complex tasks. We proposed LCORPP framework that can simultaneously learn from past data, reason with contextual knowledge, and act to

collect information. Our experiments demonstrated that using both reasoning and learning can improve robot's SDM cost.

Finally, we investigated leveraging graph-based knowledge where we used graph reasoning in order to guide robot's POMDP planner in the target search task. We used scene graphs in order to detect objects, their relationships, and their bounding boxes. We further augmented these scene graphs as the robot navigated in the environment to generate larger global scene graphs and reasoned about them to bias robot's belief about the target object.

All of the contributions used high-level domain knowledge, either in the form of rule-based declarative contextual knowledge, graph-based knowledge, or learned observation model.

All of the contributions resulted in the reduction of the costs of the action the robot takes, while maintaining a higher accuracy in accomplishing the task successfully. These observations demonstrated that leveraging knowledge and learning methods can boost the efficiency of the SDM methods. In two of the contributions (Chapters 5 and 8), we introduced methods to avoid modeling all the objects in the state-space and dealing with the curse of dimensionality. Our approaches, enabled the robot to only model the task-relevant objects and use graphs, to represent and reason with objects and their relationships in the robot SDM framework.

Knowledge representation and reasoning (KRR) is one of the earliest research topics in artificial intelligence, and sequential decision making (SDM) is one of the most important topics in intelligent robotics. Due to their very different formulations and computational paradigms, research developments in KRR and

SDM are largely isolated in the past. In this dissertation, we develop a few novel methods to leverage the complementary properties of KRR methods and SDM. Results show that our developed algorithms and systems have enabled robots to complete a variety of tasks more efficiently, while at the same time improving the success rate.

10 FUTURE WORK

This dissertation focuses on developing algorithms for leveraging declarative and graph-based knowledge to guide sequential decision making (SDM), presents an approach of acquiring knowledge from SDM experience of human-robot dialog, and discusses how interactive perception can be modeled as an SDM problem.

Within the context of robot multi-sensory perception, we focused on a robot exploring objects in a tabletop scenario and developed multi-modal predicate identification (MPI) using SDM methods. In the future, we plan to investigate applying the approach presented in Chapter 5 to tasks that involve more human-robot interaction and mobile robot platforms, where exploration would require navigation actions and perceptual modalities such as human-robot dialog. In this dissertation, the robot has been enabled to do multi-modal predicate identification given a query. Towards a more general research application of this approach, one can expand this problem to using the robot's policy to identify and rank the object predicate(s) without any queries. Another direction of the future research could be conducting multi-modal predicate identification when more than one object is in robot's proximity. In other words, given a query such as "*which object is heavy and rectangular?*", the robot would need to identify the object matching the query among the many existing objects . A hierarchical approach can be beneficial, where a higher level suggests the robot regarding object selection, and a

lower level performs multi-modal SDM to identify the object predicates.

In Chapter 6, we develop a dual-track controller for dialog management and knowledge acquisition. In the future, we intend to improve our agent by minimizing the dialog duration (e.g. less double-checking attempts) as well as augmenting its knowledge with more complex structures, e.g., to model subclasses of items. Such structures make knowledge management more difficult. Another direction is to enable the agent to learn and merge synonyms via human-robot dialog, and remove incorrect (or unnecessary) concepts on an as-needed basis. Another way to improve the current dialog system would be to develop the end-to-end dialog systems that learn policies for both dialog management and knowledge augmentation at the same time. Such data-driven methods can potentially generalize knowledge acquisition beyond domains of fulfilling service requests. However, large datasets need to be collected in order for such systems to perform efficiently.

In Chapter 7, we develop LCORPP framework which is the first SDM framework that simultaneously supports the AI capabilities of supervised learning, automated reasoning, and probabilistic planning. There are a number of ways to improve LCORPP, the key contribution made in Chapter 7. First of all, LCORPP lacks a self-diagnosis module for monitoring the performance and correctness of the framework state estimations. This diagnosis component can be used to enable reconciliation among the learning, reasoning, and planning components. It can be interesting to consider situations where contradictions exist among LCORPP's components, e.g., the classifier's output consistently does not agree with the reasoning results from human knowledge. Second, the current version of LCORPP can only handle states and actions in discrete spaces while many sequential deci-

sion making problems require continuous states, actions, or both. For instance, in the human intention estimation problem, the robot can figure out how close it should approach the human using parametrized action spaces [145], whereas LCORPP only supports discrete actions. As a consequence, using LCORPP, our robot can only move and turn for predefined distances and degrees respectively. In such cases, the coupling between a regression-based learning component and a probabilistic logic-based reasoner can no longer be based on cross-validation confusion matrix. Third, in this work, we assumed robot has access to the world model, which may not be true in real-world applications. Therefore, robot needs to leverage learning algorithms, e.g. reinforcement learning [102], to help the robot learn to make decisions in the absence of world models. Forth, in this work, our assumption is that, whenever the robot retrains the classifier, it is using the same hyperparamters (in deep learning-based classifiers). While the robot is augmenting its data, there is the potential of further improving the classifier's performance through updating hyperparamters.

From the application perspective, there are many promising ways of further making progress of LCORPP. For instance, our robot's depth camera has a limited sensing range of only six meters, making it very difficult to detect and track humans farther away from the robot. Also, if people approach the robot from behind, the robot would not be able to detect them. To mitigate this problem, we are exploring methods of using multiple cameras mounted in different directions and including other sensory modalities, such as audio to estimate human intention. The human detection algorithm's frequency is lower than the frequency of human's steps during fast walking, which frequently causes the robot losing track of people

in it field of view. In addition, in large crowds, robot can only interact with one person at a time, as the current LCORPP does not allow the robot to interact with multiple humans simultaneously. Mitigating this limitation requires a more complex hardware component added to the robot (e.g. a rotating humanoid head allowing it to communicate and have eye contact with multiple people all at the same time). People tend to change their intention over time, and such changes, which are out of the robot's control, are not modeled in this work. For instance, the human trajectory may indicate the lack of human intention in interacting with the robot, but, once the robot starts to take motion-based and language-based actions, the person might change his/her mind due to the robot's behaviors, and decide to interact with the robot (e.g., in cases where human got distracted and overlooked the robot). In our datatset, there were features such as head direction that we did not use for classifier training because of hardware limitations. In practice, leveraging those features improves the classifier accuracy.

There exists a number of SDM domains in the real world such as games, medical diagnosis, finance, urban planning, education, and robotics where people can potentially leverage LCORPP to improve their decision-making performance. One can apply LCORPP to problems of SDM under uncertainty, as long as there exist labeled datasets that contain past made decisions, experts' declarative contextual knowledge, and a model for active information collection. One important line of research for future work is to apply LCORPP to other application domains to demonstrate and evaluate its performance toward further improving LCORPP's effectiveness and applicability.

In Chapter 8, we develop the SARP algorithm to leverage graph-based scene

analysis and perceptual knowledge to guide robot planning. In the future, we intend to further improve this work both in perception capabilities (e.g., *enabling face detection*) and reasoning capabilities(e.g., *considering cases where objects may be replaced during the task execution*). In particular, we leverage knowledge graphs in order to scale up on the number of objects the robot is able to interact with. However, the robot's belief is limited to small state space that represents discrete locations, which limits the scale of the environment the robot may need to navigate. One direction for future work would be to replace the point-based POMDP solver [11] with those POMDP systems that use Monte Carlo tree search, e.g., POMCP [146], where the effectiveness of the Monte Carlo tree search idea has been demonstrated in challenging sequential decision making problems [147]. Another approach to resolve this limitation would be to use data-driven methods in order to implicitly represent the belief state using neural networks [148, 149, 150] where the robot's current observation can be fed into layers of convolutional neural networks, and then concatenated with other state variables such as robot pose or the constructed graph embeddings. Another advantage of using the data-driven approach would be the continuous modelling of the state space or using solvers for continuous POMDP such as [151]. Similar approaches can be applied to learning policy for continuous action spaces using neural network-based methods or using parameterized action space methods [152].

There are some general future research directions in knowledge-based SDM under partial observability. In this dissertation, the knowledge has been constructed by humans directly or indirectly. One way to pursue the future research is to develop methods of automating knowledge graph construction, and validation using

perception and learning. In addition, our focus has been on knowledge-based SDM of single agents, while the SDM of the multi-agent robotics systems [153, 154] could be improved by using various forms of the knowledge. Similarly, in applications that involved humans, our assumption has been that our robot has always been interacting with no more than one person, while in real-world scenarios, there are cases where multiple people interact with the robot, making robot perception more challenging. Safety of decision making is critical for robots in human-inhabited environments. Another line of research to pursue would be to use human knowledge in order to improve the safety of robot's decision making [155, 156].

Bibliography

[1] Nick Hawes, Christopher Burbridge, Ferdian Jovan, Lars Kunze, Bruno Lacerda, Lenka Mudrova, Jay Young, Jeremy Wyatt, Denise Hebesberger, Tobias Kortner, et al. "The strands project: Long-term autonomy in everyday environments". In: *IEEE Robotics & Automation Magazine* 24.3 (2017), pp. 146–156.

[2] Manuela M Veloso. *The increasingly fascinating opportunity for human-robot-ai interaction: The cobot mobile service robots*. 2018.

[3] Piyush Khandelwal, Shiqi Zhang, Jivko Sinapov, Matteo Leonetti, Jesse Thomason, Fangkai Yang, Ilaria Gori, Maxwell Svetlik, Priyanka Khante, Vladimir Lifschitz, et al. "Bwibots: A platform for bridging the gap between ai and human–robot interaction research". In: *The International Journal of Robotics Research* 36.5-7 (2017), pp. 635–659.

[4] Sebastian Thrun, Maren Bennewitz, Wolfram Burgard, Armin B Cremers, Frank Dellaert, Dieter Fox, Dirk Hahnel, Charles Rosenberg, Nicholas Roy, Jamieson Schulte, et al. "MINERVA: A second-generation museum tour-guide robot". In: *Proceedings 1999 IEEE International Conference on Robotics and Automation (Cat. No. 99CH36288C)*. Vol. 3. IEEE. 1999.

[5] Wolfram Burgard, Armin B Cremers, Dieter Fox, Dirk Hähnel, Gerhard Lakemeyer, Dirk Schulz, Walter Steiner, and Sebastian Thrun. "The interactive museum tour-guide robot". In: *Aaai/iaai*. 1998, pp. 11–18.

[6] Malik Ghallab, Dana Nau, and Paolo Traverso. *Automated planning and acting*. Cambridge University Press, 2016.

[7] Brenna D Argall, Sonia Chernova, Manuela Veloso, and Brett Browning. "A survey of robot learning from demonstration". In: *Robotics and autonomous systems* 57.5 (2009), pp. 469–483.

[8] Michael Gelfond and Yulia Kahl. *Knowledge representation, reasoning, and the design of intelligent agents: The answer-set programming approach*. Cambridge University Press, 2014.

[9] Martin L Puterman. *Markov decision processes: discrete stochastic dynamic programming*. John Wiley & Sons, 2014.

[10] Leslie Pack Kaelbling, Michael L Littman, and Anthony R Cassandra. "Planning and acting in partially observable stochastic domains". In: *Artificial intelligence* 101.1-2 (1998), pp. 99–134.

[11] Hanna Kurniawati, David Hsu, and Wee Sun Lee. "Sarsop: Efficient point-based pomdp planning by approximating optimally reachable belief spaces." In: *Robotics: Science and systems*. Vol. 2008. Citeseer. 2008.

[12] Joelle Pineau, Geoffrey J Gordon, and Sebastian Thrun. "Applying Metric-Trees to Belief-Point POMDPs." In: *NIPS*. 2003, pp. 759–766.

[13] Matthijs TJ Spaan and Nikos Vlassis. "Perseus: Randomized point-based value iteration for POMDPs". In: *Journal of artificial intelligence research* 24 (2005), pp. 195–220.

[14] Nan Ye, Adhiraj Somani, David Hsu, and Wee Sun Lee. "Despot: Online pomdp planning with regularization". In: *Journal of Artificial Intelligence Research* 58 (2017), pp. 231–266.

[15] David L Poole and Alan K Mackworth. *Artificial Intelligence: foundations of computational agents*. Cambridge University Press, 2010.

[16] Randall Davis, Howard Shrobe, and Peter Szolovits. "What is a knowledge representation?" In: *AI magazine* 14.1 (1993), pp. 17–17.

[17] Michael Gelfond and Vladimir Lifschitz. "The stable model semantics for logic programming". In: *Proceedings of International Logic Programming Conference and Symposium*. MIT Press, 1988, pp. 1070–1080.

[18] Gerhard Brewka, Thomas Eiter, and Mirosław Truszczyński. "Answer set programming at a glance". In: *Communications of the ACM* 54.12 (2011), pp. 92–103.

[19] Vladimir Lifschitz. *Answer set programming*. Springer Berlin, 2019.

[20] Justin Johnson, Ranjay Krishna, Michael Stark, Li-Jia Li, David Shamma, Michael Bernstein, and Li Fei-Fei. "Image retrieval using scene graphs". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015, pp. 3668–3678.

[21] John J Hopfield. "Neural networks and physical systems with emergent collective computational abilities". In: *Proceedings of the national academy of sciences* 79.8 (1982), pp. 2554–2558.

[22] Sepp Hochreiter and Jürgen Schmidhuber. "Long short-term memory". In: *Neural computation* 9.8 (1997), pp. 1735–1780.

[23] Alex Graves, Abdel-rahman Mohamed, and Geoffrey Hinton. "Speech recognition with deep recurrent neural networks". In: *2013 IEEE international conference on acoustics, speech and signal processing*. IEEE. 2013, pp. 6645–6649.

[24] Oriol Vinyals, Alexander Toshev, Samy Bengio, and Dumitru Erhan. "Show and tell: A neural image caption generator". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015, pp. 3156–3164.

[25] Morgan Quigley, Ken Conley, Brian Gerkey, Josh Faust, Tully Foote, Jeremy Leibs, Rob Wheeler, and Andrew Y Ng. "ROS: an open-source Robot Operating System". In: *ICRA workshop on open source software*. 2009.

[26] Shiqi Zhang and Mohan Sridharan. "A Survey of Knowledge-based Sequential Decision Making under Uncertainty". In: *arXiv preprint arXiv:2008.08548* (2020).

[27] Matteo Leonetti, Luca Iocchi, and Peter Stone. "A synthesis of automated planning and reinforcement learning for efficient, robust decision-making". In: *Artificial Intelligence* 241 (2016), pp. 103–130.

[28] Fangkai Yang, Daoming Lyu, Bo Liu, and Steven Gustafson. "Peorl: Integrating symbolic planning and hierarchical reinforcement learning for robust decision-making". In: *arXiv preprint arXiv:1804.07779* (2018).

[29] Daoming Lyu, Fangkai Yang, Bo Liu, and Steven Gustafson. "Sdrl: Interpretable and data-efficient deep reinforcement learning leveraging symbolic planning". In: *Proceedings AAAI*. Vol. 33. 2019, pp. 2970–2977.

[30] Mohan Sridharan and Sarah Rainge. "Integrating reinforcement learning and declarative programming to learn causal laws in dynamic domains". In: *International Conference on Social Robotics*. 2014.

[31] Shane Griffith, Kaushik Subramanian, Jonathan Scholz, Charles L Isbell, and Andrea L Thomaz. "Policy shaping: Integrating human feedback with reinforcement learning". In: *Advances in neural information processing systems*. 2013, pp. 2625–2633.

[32] León Illanes, Xi Yan, Rodrigo Toro Icarte, and Sheila A McIlraith. "Symbolic Planning and Model-Free Reinforcement Learning: Training Taskable Agents". In: *4th Multidisciplinary Conference on RLDM*. 2019.

[33] Yohei Hayamizu, Saeid Amiri, Kishan Chandan, Keiki Takadama, and Shiqi Zhang. "Guiding robot exploration in reinforcement learning via automated planning". In: *Proceedings of the International Conference on Automated Planning and Scheduling*. Vol. 31. 2021, pp. 625–633.

[34] Matteo Leonetti, Luca Iocchi, and Peter Stone. "A synthesis of automated planning and reinforcement learning for efficient, robust decision-making". In: *Artif. Intell.* (2016).

[35] Mohan Sridharan, Michael Gelfond, Shiqi Zhang, and Jeremy Wyatt. "REBA: A refinement-based architecture for knowledge representation and reasoning in robotics". In: *Journal of Artificial Intelligence Research* 65 (2019), pp. 87–180.

[36] Shiqi Zhang and Peter Stone. "CORPP: Commonsense reasoning and probabilistic planning, as applied to dialog with a mobile robot". In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 29. 1. 2015.

[37] Dongcai Lu, Shiqi Zhang, Peter Stone, and Xiaoping Chen. "Leveraging commonsense reasoning and multimodal perception for robot spoken dialog systems". In: *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2017, pp. 6582–6588.

[38] Rohan Chitnis, Leslie Pack Kaelbling, and Tomás Lozano-Pérez. "Integrating human-provided information into belief state representation using dynamic factorization". In: *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2018, pp. 3551–3558.

[39] Saeid Amiri, Mohammad Shokrolah Shirazi, and Shiqi Zhang. "Learning and reasoning for robot sequential decision making under uncertainty". In: *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*. Vol. 34. 03. 2020, pp. 2726–2733.

[40] Moritz Göbelbecker, Charles Gretton, and Richard Dearden. "A switching planner for combined task and observation planning". In: *Twenty-Fifth AAAI Conference on Artificial Intelligence*. 2011.

[41] Marc Hanheide, Moritz Göbelbecker, Graham S Horn, Andrzej Pronobis, Kristoffer Sjöö, Alper Aydemir, Patric Jensfelt, Charles Gretton, Richard Dearden, Miroslav Janicek, et al. "Robot task planning and explanation in open and uncertain worlds". In: *Artificial Intelligence* 247 (2017), pp. 119–150.

[42] Shiqi Zhang, Mohan Sridharan, and Jeremy L Wyatt. "Mixed logical inference and probabilistic planning for robots in unreliable worlds". In: *IEEE Transactions on Robotics* 31.3 (2015), pp. 699–713.

[43] Leonardo Ferreira, Reinaldo Bianchi, Paulo Santos, and Ramon Lopez de Mantaras. "Answer set programming for non-stationary markov decision processes". In: *Applied Intelligence* (2017).

[44] Keting Lu, Shiqi Zhang, Peter Stone, and Xiaoping Chen. "Robot Represention and Reasoning with Knowledge from Reinforcement Learning". In: *arXiv:1809.11074* (2018).

[45] Brendan Juba. "Integrated common sense learning and planning in POMDPs". In: *The Journal of Machine Learning Research* 17.1 (2016), pp. 3276–3312.

[46] Shiqi Zhang, Piyush Khandelwal, and Peter Stone. "Dynamically Constructed (PO) MDPs for Adaptive Robot Planning." In: *AAAI*. 2017, pp. 3855–3863.

[47] Yuqian Jiang, Nick Walker, Justin Hart, and Peter Stone. "Open-World Reasoning for Service Robots". In: *Proceedings of the 29th International Conference on Automated Planning and Scheduling (ICAPS)*. 2019.

[48] Guowei Cui, Wei Shuai, and Xiaoping Chen. "Semantic Task Planning for Service Robots in Open Worlds". In: *Future Internet* 13.2 (2021), p. 49.

[49] Yifeng Zhu, Jonathan Tremblay, Stan Birchfield, and Yuke Zhu. "Hierarchical Planning for Long-Horizon Manipulation with Geometric and Symbolic Scene Graphs". In: *arXiv preprint arXiv:2012.07277* (2020).

[50] Zhen Zeng, Adrian Röfer, and Odest Chadwicke Jenkins. "Semantic linking maps for active visual object search". In: *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2020, pp. 1984–1990.

[51] Yi Wu, Yuxin Wu, Aviv Tamar, Stuart Russell, Georgia Gkioxari, and Yuandong Tian. "Bayesian relational memory for semantic visual navigation". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2019, pp. 2769–2779.

[52] Hanna M Wallach. "Conditional random fields: An introduction". In: *Technical Reports (CIS)* (2004), p. 22.

[53] Rodrigo Toro Icarte, Toryn Klassen, Richard Valenzano, and Sheila McIlraith. "Using reward machines for high-level task specification and decomposition in reinforcement learning". In: *ICML*. 2018.

[54] Yuqian Jiang, Fangkai Yang, Shiqi Zhang, and Peter Stone. "Task-Motion Planning with Reinforcement Learning for Adaptable Mobile Service Robots". In: *2019 IEEE/RSJ International Conference on IROS*. 2019.

[55] Harsha Kokel, Arjun Manoharan, Sriraam Natarajan, Balaraman Ravindran, and Prasad Tadepalli. "RePReL: Integrating Relational Planning and Reinforcement Learning for Effective Abstraction". In: *Proceedings of the International Conference on Automated Planning and Scheduling*. Vol. 31. 2021, pp. 533–541.

[56] Daniel Neider, Jean-Raphael Gaglione, Ivan Gavran, Ufuk Topcu, Bo Wu, and Zhe Xu. "Advice-Guided Reinforcement Learning in a non-Markovian Environment". In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 35. 10. 2021, pp. 9073–9080.

[57] Liangpeng Zhang, Ke Tang, and Xin Yao. "Explicit planning for efficient exploration in reinforcement learning". In: *Advances in Neural Information Processing Systems* 32 (2019), pp. 7488–7497.

[58] Leonardo Ferreira, Thiago dos Santos, Reinaldo Bianchi, and Paulo Santos. "Solving Safety Problems with Ensemble Reinforcement Learning". In: *AJCAI*. Springer. 2019, pp. 203–214.

[59] Haodi Zhang, Zihang Gao, Yi Zhou, Hao Zhang, Kaishun Wu, and Fangzhen Lin. "Faster and Safer Training by Embedding High-Level Knowledge into Deep Reinforcement Learning". In: *arXiv preprint arXiv:1910.09986* (2019).

[60] León Illanes, Xi Yan, Rodrigo Toro Icarte, and Sheila A McIlraith. "Symbolic Plans as High-Level Instructions for Reinforcement Learning". In: *ICAPS*. Vol. 30. 2020, pp. 540–550.

[61] Daniel Gordon, Dieter Fox, and Ali Farhadi. "What should I do now? marrying reinforcement learning and symbolic planning". In: *arXiv preprint arXiv:1901.01492* (2019).

[62] Rodrigo Toro Icarte, Ethan Waldie, Toryn Klassen, Rick Valenzano, Margarita Castro, and Sheila McIlraith. "Learning reward machines for partially observable reinforcement learning". In: *Advances in Neural Information Processing Systems* 32 (2019), pp. 15523–15534.

[63] Matthew E Taylor and AI Borealis. "Improving Reinforcement Learning with Human Input." In: *IJCAI*. 2018, pp. 5724–5728.

[64] Andrew M Wells, Neil T Dantam, Anshumali Shrivastava, and Lydia E Kavraki. "Learning feasibility for task and motion planning in tabletop environments". In: *IEEE robotics and automation letters* (2019).

[65] Sunandita Patra, James Mason, Amit Kumar, Malik Ghallab, Paolo Traverso, and Dana Nau. "Integrating Acting, Planning, and Learning in Hierarchical Operational Models". In: *Proceedings of the International Conference on Automated Planning and Scheduling*. Vol. 30. 2020, pp. 478–487.

[66] Todd Hester, Matej Vecerik, Olivier Pietquin, Marc Lanctot, Tom Schaul, Bilal Piot, Dan Horgan, John Quan, Andrew Sendonaris, Ian Osband, et al. "Deep q-learning from demonstrations". In: *Thirty-second AAAI conference on artificial intelligence*. 2018.

[67] Neha P Garg, David Hsu, and Wee Sun Lee. "Learning to grasp under uncertainty using POMDPs". In: *2019 International Conference on Robotics and Automation (ICRA)*. IEEE. 2019, pp. 2751–2757.

[68] Jaime G Carbonell, Craig A Knoblock, and Steven Minton. "Prodigy: An integrated architecture for planning and learning". In: *Architectures for intelligence*. Psychology Press, 2014, pp. 255–292.

[69] Dongkyu Choi and Pat Langley. "Evolution of the ICARUS cognitive architecture". In: *Cognitive Systems Research* 48 (2018), pp. 25–38.

[70] John R Anderson, Michael Matessa, and Christian Lebiere. "ACT-R: A theory of higher level cognition and its relation to visual attention". In: *Human–Computer Interaction* 12.4 (1997), pp. 439–462.

[71] John E Laird. *The Soar cognitive architecture*. MIT press, 2012.

[72] Jeannette Bohg, Karol Hausman, Bharath Sankaran, Oliver Brock, Danica Kragic, Stefan Schaal, and Gaurav S Sukhatme. "Interactive perception: Leveraging action in perception and perception in action". In: *IEEE Transactions on Robotics* 33.6 (2017), pp. 1273–1291.

[73] Sylvie CW Ong, Shao Wei Png, David Hsu, and Wee Sun Lee. "Planning under uncertainty for robotic tasks with mixed observability". In: *The International Journal of Robotics Research* 29.8 (2010), pp. 1053–1068.

[74] Saeid Amiri, Suhua Wei, Shiqi Zhang, Jivko Sinapov, Jesse Thomason, and Peter Stone. "Multi-modal Predicate Identification using Dynamically Learned Robot Controllers." In: *IJCAI*. 2018, pp. 4638–4645.

[75] Virgile Högman, Marten Björkman, and Danica Kragic. "Interactive object classification using sensorimotor contingencies". In: *IROS*. IEEE. 2013.

[76] Jivko Sinapov, Connor Schenck, Kerrick Staley, Vladimir Sukhoy, and Alexander Stoytchev. "Grounding semantic categories in behavioral interactions: Experiments with 100 objects". In: *Robotics and Autonomous Systems* 62.5 (2014), pp. 632–645.

[77] Jesse Thomason, Jivko Sinapov, Maxwell Svetlik, Peter Stone, and Raymond J Mooney. "Learning Multi-Modal Grounded Linguistic Semantics by Playing I Spy". In: *IJCAI*. 2016.

[78] Jivko Sinapov and Alexander Stoytchev. "From acoustic object recognition to object categorization by a humanoid robot". In: *the RSS Workshop: Mobile Manipulation in Human Environments*. 2009.

[79] Vivian Chu, Ian McMahon, Lorenzo Riano, Craig G McDonald, Qin He, Jorge Martinez Perez-Tejada, Michael Arrigo, Trevor Darrell, and Katherine J Kuchenbecker. "Robotic learning of haptic adjectives through physical interaction". In: *Robotics and Autonomous Systems* 63 (2015), pp. 279–292.

[80] Muhannad Alomari, Paul Duckworth, David C. Hogg, and Anthony G. Cohn. "Natural Language Acquisition and Grounding for Embodied Robotic Systems". In: *AAAI*. Feb. 2017.

[81] David Whitney, Miles Eldon, John Oberlin, and Stefanie Tellex. "Interpreting Multimodal Referring Expressions in Real Time". In: *ICRA*. 2016.

[82] Jayant Krishnamurthy and Thomas Kollar. "Jointly Learning to Parse and Perceive: Connecting Natural Language to the Physical World". In: *Transactions of the Association for Computational Linguistics* 1 (2013), pp. 193–206.

[83] Cynthia Matuszek, Nicholas FitzGerald, Luke Zettlemoyer, Liefeng Bo, and Dieter Fox. "A Joint Model of Language and Perception for Grounded Attribute Learning". In: *ICML*. 2012.

[84] Yang Gao, Lisa Anne Hendricks, Katherine J Kuchenbecker, and Trevor Darrell. "Deep learning for tactile understanding from visual and haptic data". In: *ICRA*. IEEE. 2016, pp. 536–543.

[85] Antons Rebguns, Daniel Ford, and Ian R Fasel. "InfoMax Control for Acoustic Exploration of Objects by a Mobile Robot." In: *Lifelong Learning*. 2011.

[86] Jeremy Fishel and Gerald Loeb. "Bayesian Exploration for Intelligent Identification of Textures". In: *Frontiers in Neurorobotics* 6 (2012), p. 4.

[87] Jesse Thomason, Jivko Sinapov, Raymond Mooney, and Peter Stone. "Guiding Exploratory Behaviors for Multi-Modal Grounding of Linguistic Descriptions". In: *AAAI*. New Orleans, Louisiana, 2018.

[88] Mohan Sridharan, Jeremy Wyatt, and Richard Dearden. "Planning to see: A hierarchical approach to planning visual actions on a robot using POMDPs". In: *Artificial Intelligence* 174.11 (2010), pp. 704–725.

[89] Robert Eidenberger and Josef Scharinger. "Active perception and scene modeling by planning with probabilistic 6d object poses". In: *IROS*. IEEE. 2010, pp. 1036–1043.

[90] Joni Pajarinen and Ville Kyrki. "Robotic manipulation of multiple objects as a POMDP". In: *Artificial Intelligence* (2015).

[91] Misha Denil, Pulkit Agrawal, Tejas D Kulkarni, Tom Erez, Peter Battaglia, and Nando de Freitas. "Learning to Perform Physics Experiments via Deep Reinforcement Learning". In: *International Conference on Learning Representations*. 2017.

[92] Jivko Sinapov, Connor Schenck, and Alexander Stoytchev. "Learning relational object categories using behavioral exploration and multimodal perception". In: *ICRA*. 2014, pp. 5691–5698.

[93] Yingfeng Chen, Feng Wu, Wei Shuai, and Xiaoping Chen. "Robots serve humans in public places — KeJia robot as a shopping assistant". In: *International Journal of Advanced Robotic Systems* 14.3 (2017).

[94] Jesse Thomason, Shiqi Zhang, Raymond J. Mooney, and Peter Stone. "Learning to interpret natural language commands through human-robot dialog". In: *Proceedings of the 24th International Conference on Artificial Intelligence*. 2015, pp. 1923–1929.

[95] Saeid Amiri, Sujay Bajracharya, Cihangir Goktolgal, Jesse Thomason, and Shiqi Zhang. "Augmenting knowledge through statistical, goal-oriented human-robot dialog". In: *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2019, pp. 744–750.

[96] Cynthia Matuszek, Evan Herbst, Luke Zettlemoyer, and Dieter Fox. "Learning to parse natural language commands to a robot control system". In: *Experimental Robotics*. 2013, pp. 403–415.

[97] Dipendra K Misra, Jaeyong Sung, Kevin Lee, and Ashutosh Saxena. "Tell me dave: Context-sensitive grounding of natural language to manipulation instructions". In: *The International Journal of Robotics Research* 35.1-3 (2016), pp. 281–300.

[98] Stefanie Tellex, Thomas Kollar, Steven Dickerson, Matthew Walter, Ashis Banerjee, Seth Teller, and Nicholas Roy. "Understanding natural language commands for robotic navigation and mobile manipulation". In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 25. 1. 2011.

[99] Michael Spranger and Luc Steels. "Co-acquisition of syntax and semantics: an investigation in spatial language". In: *Proceedings of the 24th International Conference on Artificial Intelligence*. 2015.

[100] Ze Gong and Yu Zhang. "Temporal Spatial Inverse Semantics for Robots Communicating with Humans". In: *Proceedings of (ICRA)*. 2018.

[101] Satinder Singh, Diane Litman, Michael Kearns, and Marilyn Walker. "Optimizing dialogue management with reinforcement learning: Experiments with the NJFun system". In: *JAIR* 16 (2002), pp. 105–133.

[102] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

[103] Heriberto Cuayáhuitl. "SimpleDS: A simple deep reinforcement learning dialogue system". In: *Dialogues with Social Robots*. 2017.

[104] Keting Lu, Shiqi Zhang, and Xiaoping Chen. "Goal-oriented Dialogue Policy Learning from Failures". In: *AAAI*. 2019.

[105] Aishwarya Padmakumar, Jesse Thomason, and Raymond J. Mooney. "Integrated learning of dialog strategies and semantic parsing". In: *Proceedings of the 15th Conference of the European Chapter of the ACL*. 2017, pp. 547–557.

[106] Çetin Meriçli, Steven D. Klee, Jack Paparian, and Manuela Veloso. "An interactive approach for situated task specification through verbal instructions". In: *Proceedings of the 2014 international AAMAS conference*. 2014.

[107] Vittorio Perera, Robin Soetens, Thomas Kollar, Mehdi Samadi, Yichao Sun, Daniele Nardi, René van de Molengraft, and Manuela Veloso. "Learning task knowledge from dialog and web access". In: *Robotics* 4.2 (2015), pp. 223–252.

[108] Amos Azaria, Jayant Krishnamurthy, and Tom M Mitchell. "Instructable Intelligent Personal Agent." In: *AAAI*. 2016, pp. 2681–2689.

[109] Mycal Tucker, Derya Aksaray, Rohan Paul, Gregory J Stein, and Nicholas Roy. "Learning unknown groundings for natural language interaction with mobile robots". In: *ISRR*. 2017.

[110] Lanbo She and Joyce Chai. "Interactive Learning of Grounded Verb Semantics towards Human-Robot Communication". In: *Proceedings of ACL*. 2017, pp. 1634–1644.

[111] Joyce Y Chai, Qiaozi Gao, Lanbo She, Shaohua Yang, Sari Saba-Sadiya, and Guangyue Xu. "Language to Action: Towards Interactive Task Learning with Physical Agents." In: *IJCAI*. 2018, pp. 2–9.

[112] Steve Young, Milica Gašić, Blaise Thomson, and Jason D Williams. "POMDP-based statistical spoken dialog systems: A review". In: *Proceedings of the IEEE* 101.5 (2013), pp. 1160–1179.

[113] Yoav Artzi. "Cornell SPF: Cornell semantic parsing framework". In: *arXiv preprint arXiv:1311.3011* (2013).

[114] Scott Sanner et al. "Relational dynamic influence diagram language (rddl): Language description". In: *Unpublished ms. Australian National University* 32 (2010), p. 27.

[115] Haonan Chen, Hao Tan, Alan Kuntz, Mohit Bansal, and Ron Alterovitz. "Enabling robots to understand incomplete natural language instructions using commonsense reasoning". In: *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2020, pp. 1963–1969.

[116] Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. "HellaSwag: Can a machine really finish your sentence?" In: *arXiv preprint arXiv:1905.07830* (2019).

[117] Rowan Zellers, Yonatan Bisk, Ali Farhadi, and Yejin Choi. "From recognition to cognition: Visual commonsense reasoning". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2019, pp. 6720–6731.

[118] Somak Aditya, Rudra Saha, Yezhou Yang, and Chitta Baral. "Spatial knowledge distillation to aid visual reasoning". In: *2019 IEEE Winter Conference on Applications of Computer Vision (WACV)*. IEEE. 2019, pp. 227–235.

[119] Sonia Chernova, Vivian Chu, Angel Daruna, Haley Garrison, Meera Hahn, Priyanka Khante, Weiyu Liu, and Andrea Thomaz. "Situated bayesian reasoning framework for robots operating in diverse everyday environments". In: *Robotics Research*. Springer, 2020, pp. 353–369.

[120] Tiago Mota and Mohan Sridharan. "Commonsense Reasoning and Knowledge Acquisition to Guide Deep Learning on Robots". In: *Robotics Science and Systems. Freiburg, Germany* (2019).

[121] Andrea Lockerd Thomaz, Cynthia Breazeal, et al. "Reinforcement learning with human teachers: Evidence of feedback and guidance with implications for learning performance". In: *Aaai*. Vol. 6. Boston, MA. 2006, pp. 1000–1005.

[122] Chitta Baral, Michael Gelfond, and Nelson Rushton. "Probabilistic reasoning with answer sets". In: *Theory and Practice of Logic Programming* 9.1 (2009), pp. 57–144.

[123] Stephen H Bach, Matthias Broecheler, Bert Huang, and Lise Getoor. "Hinge-loss markov random fields and probabilistic soft logic". In: *The Journal of Machine Learning Research* 18.1 (2017), pp. 3846–3912.

[124] Matthew Richardson and Pedro Domingos. "Markov logic networks". In: *Machine learning* 62.1-2 (2006), pp. 107–136.

[125] Danfei Xu, Yuke Zhu, Christopher B Choy, and Li Fei-Fei. "Scene graph generation by iterative message passing". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017, pp. 5410–5419.

[126] Rowan Zellers, Mark Yatskar, Sam Thomson, and Yejin Choi. "Neural motifs: Scene graph parsing with global context". In: *CVPR*. 2018, pp. 5831–5840.

[127] Yikang Li, Wanli Ouyang, Bolei Zhou, Kun Wang, and Xiaogang Wang. "Scene graph generation from objects, phrases and region captions". In: *Proceedings of the IEEE International Conference on Computer Vision*. 2017, pp. 1261–1270.

[128] Tianshui Chen, Weihao Yu, Riquan Chen, and Liang Lin. "Knowledge-embedded routing network for scene graph generation". In: *CVPR*. 2019, pp. 6163–6171.

[129] Antoni Rosinol, Arjun Gupta, Marcus Abate, Jingnan Shi, and Luca Carlone. "3D Dynamic Scene Graphs: Actionable Spatial Perception with Places, Objects, and Humans". In: *arXiv preprint arXiv:2002.06289* (2020).

[130] Martin Engelcke, Adam R Kosiorek, Oiwi Parker Jones, and Ingmar Posner. "Genesis: Generative scene inference and sampling with object-centric latent representations". In: *arXiv preprint arXiv:1907.13052* (2019).

[131] Zhixuan Lin, Yi-Fu Wu, Skand Vishwanath Peri, Weihao Sun, Gautam Singh, Fei Deng, Jindong Jiang, and Sungjin Ahn. "Space: Unsupervised object-oriented scene representation via spatial attention and decomposition". In: *arXiv preprint arXiv:2001.02407* (2020).

[132] Saeid Amiri, Kishan Chandan, and Shiqi Zhang. "Reasoning With Scene Graphs for Robot Planning Under Partial Observability". In: *IEEE Robotics and Automation Letters* 7.2 (2022), pp. 5560–5567. DOI: 10.1109/LRA.2022.3157567.

[133] Ernest Davis and Gary Marcus. "Commonsense reasoning and commonsense knowledge in artificial intelligence". In: *Communications of the ACM* 58.9 (2015), pp. 92–103.

[134] Dana Nau, Malik Ghallab, and Paolo Traverso. *Automated Planning: Theory & Practice*. 2004.

[135] Patrik Haslum, Nir Lipovetzky, Daniele Magazzeni, and Christian Muise. "An introduction to the planning domain definition language". In: *Synthesis Lectures on Artificial Intelligence and Machine Learning* 13.2 (2019), pp. 1–187.

[136] Daphne Koller and Nir Friedman. *Probabilistic graphical models: principles and techniques*. MIT press, 2009.

[137] Kaiyu Zheng, Deniz Bayazit, Rebecca Mathew, Ellie Pavlick, and Stefanie Tellex. "Spatial Language Understanding for Object Search in Partially Observed City-scale Environments". In: *2021 30th IEEE International Conference on RO-MAN*. IEEE. 2021, pp. 315–322.

[138] Arthur Wandzel, Yoonseon Oh, Michael Fishman, Nishanth Kumar, Lawson LS Wong, and Stefanie Tellex. "Multi-object search using object-oriented pomdps". In: *ICRA*. IEEE. 2019, pp. 7194–7200.

[139] Jue Kun Li, David Hsu, and Wee Sun Lee. "Act to see and see to act: POMDP planning for objects search in clutter". In: *2016 IEEE/RSJ International Conference IROS*. IEEE. 2016, pp. 5701–5707.

[140] Yuchen Xiao, Sammie Katt, Andreas ten Pas, Shengjian Chen, and Christopher Amato. "Online planning for target object search in clutter under partial observability". In: *2019 International Conference on Robotics and Automation (ICRA)*. IEEE. 2019, pp. 8241–8247.

[141] Jonathan S Yedidia, William T Freeman, Yair Weiss, et al. "Generalized belief propagation". In: *NIPS*. Vol. 13. 2000, pp. 689–695.

[142] Ranjay Krishna, Yuke Zhu, Oliver Groth, Justin Johnson, Kenji Hata, Joshua Kravitz, Stephanie Chen, Yannis Kalantidis, Li-Jia Li, David A Shamma, et al. "Visual genome: Connecting language and vision using crowdsourced dense image annotations". In: *International Journal of Computer Vision* 123.1 (2017), pp. 32–73.

[143] Eric Kolve, Roozbeh Mottaghi, Winson Han, Eli VanderBilt, Luca Weihs, Alvaro Herrasti, Daniel Gordon, Yuke Zhu, Abhinav Gupta, and Ali Farhadi. "Ai2-thor: An interactive 3d environment for visual ai". In: *arXiv preprint arXiv:1712.05474* (2017).

[144] Ankur Ankan and Abinash Panda. "pgmpy: Probabilistic graphical models using python". In: *Proceedings of the 14th Python in Science Conference (SCIPY 2015). Citeseer*. Vol. 10. Citeseer. 2015.

[145] Warwick Masson, Pravesh Ranchod, and George Konidaris. "Reinforcement learning with parameterized actions". In: *Thirtieth AAAI Conference on Artificial Intelligence*. 2016.

[146] David Silver and Joel Veness. "Monte-Carlo planning in large POMDPs". In: *Advances in neural information processing systems* 23 (2010).

[147] David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, et al. "Mastering the game of go without human knowledge". In: *nature* 550.7676 (2017), pp. 354–359.

[148] Peter Karkus, David Hsu, and Wee Sun Lee. "Qmdp-net: Deep learning for planning under partial observability". In: *Advances in neural information processing systems* 30 (2017).

[149] Matthew Hausknecht and Peter Stone. "Deep recurrent q-learning for partially observable mdps". In: *2015 aaai fall symposium series*. 2015.

[150] Maximilian Igl, Luisa Zintgraf, Tuan Anh Le, Frank Wood, and Shimon Whiteson. "Deep variational reinforcement learning for POMDPs". In: *International Conference on Machine Learning*. PMLR. 2018, pp. 2117–2126.

[151] Sebastian Brechtel, Tobias Gindele, and Rüdiger Dillmann. "Solving continuous POMDPs: Value iteration with incremental learning of an efficient space representation". In: *International Conference on Machine Learning*. PMLR. 2013, pp. 370–378.

[152] Matthew Hausknecht and Peter Stone. "Deep reinforcement learning in parameterized action space". In: *arXiv preprint arXiv:1511.04143* (2015).

[153] Frans A Oliehoek and Christopher Amato. *A concise introduction to decentralized POMDPs*. Springer, 2016.

[154] Miao Liu, Kavinayan Sivakumar, Shayegan Omidshafiei, Christopher Amato, and Jonathan P How. "Learning for multi-robot cooperation in partially observable stochastic environments with macro-actions". In: *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2017, pp. 1853–1860.

[155] Binghan He, Mahsa Ghasemi, Ufuk Topcu, and Luis Sentis. "A Barrier Pair Method for Safe Human-Robot Shared Autonomy". In: *arXiv preprint arXiv:2112.00279* (2021).

[156] Mohammed Alshiekh, Roderick Bloem, Rüdiger Ehlers, Bettina Könighofer, Scott Niekum, and Ufuk Topcu. "Safe reinforcement learning via shielding". In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 32. 1. 2018.