LEARNING TO REASON ABOUT CONTEXTUAL KNOWLEDGE FOR PLANNING UNDER UNCERTAINTY

BY

CHENG CUI

BE, Shanghai Maritime University, 2017

THESIS

Submitted in partial fulfillment of the requirements for
Master of Science in Computer Science
in the Graduate School of
Binghamton University
State University of New York
2021

Accepted in partial fulfillment of the requirements for
the degree of Name of Degree in Major
in the Graduate School of
Binghamton University
State University of New York
2021

12 08, 2021

Shiqi Zhang, Chair
Department of Computer Science, Binghamton University

Sujoy Sidkar, Faculty Advisor
Department of Computer Science, Binghamton University

Abstract

Sequential decision-making (SDM) methods aim to help agents compute an action policy toward achieving long-term goals under uncertainty. Existing research has shown that contextual knowledge in declarative forms can be used for improving the performance of SDM methods. However, the contextual knowledge from people tends to be incomplete and sometimes inaccurate, which greatly limits the applicability of knowledge-based SDM methods. In this thesis, we develop an algorithm called perceptual reasoning and interactive learning (PERIL) for knowledge-based SDM under partial observability. PERIL learns from interaction experience to reason about contextual knowledge, and has been applied to urban driving scenarios. We have extensively evaluated PERIL using CARLA, a widely used autonomous driving simulator. Results demonstrate PERIL's superiority in comparison to competitive baselines from the literature.

Acknowledgements

First, I would like to express my deep appreciation to my advisor, Professor Shiqi Zhang. Professor Shiqi guided me to the research world of artificial intelligence and robotics and helped me formulate the idea of this work. Thanks a lot for his mentorship, patience, responsiveness and all kinds of support.

I also really appreciate the help from my collaborator Saeid Amiri who has been working together with me on this project over the past year. As a senior PhD student, Saeid taught me a lot about research, explained different concepts and cleared my confusions during our many discussions.

Xingyue is another collaborator who support the project in many aspects and I really appreciate that.

Contents

List of Tables

List of Figures

List of Abbreviations

CNNs Convolutional Neural Networks

MDP Markov Decision Process

MLNs Markov Logic Networks

POMDPs Partially Observable MDPs

SDM Sequential Decision Making

1 Introduction

Intelligent agents are increasingly present in our everyday life, including those embodied agents (robots) that interact with the real world. The robots need to estimate current world state while determining what to do based on the current state estimation, resulting in problems of sequential decision making (SDM) under partial observability [1]. Existing research has demonstrated that agents' SDM capability can be improved by reasoning with contextual knowledge to estimate the current world state [2, 3, 4]. However, the contextual knowledge provided by domain experts can hardly be comprehensive, and sometimes includes inaccurate information. Motivated by the observation that the agents need significant efforts to recover from inaccurate knowledge in SDM tasks [4], we aim to develop an approach to help the SDM agents learn to reason about contextual knowledge.

Consider an urban driving scenario, where an autonomous vehicle needs to change lanes, say to merge left. On the one hand, the agent needs to perceive the environment, e.g., using Lidar range sensors, to detect if there is enough room in the desired lane. The perception output, together with other contextual information (say weather and traffic), is then processed in a reasoning system to estimate the world state, including the intentions of other drivers. We use **perceptual reasoning** to refer to

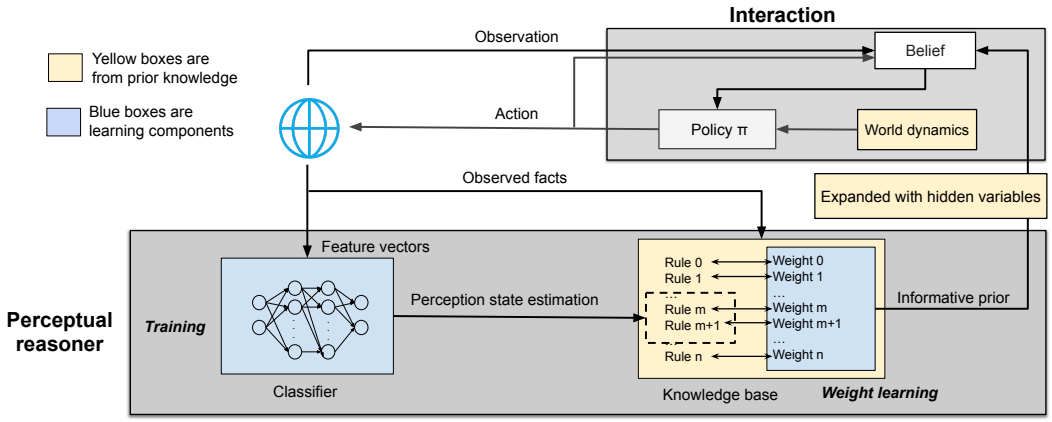

Figure 1.1: An overview of PERIL. The perceptual reasoner component consists of a classifier for passive perception and a knowledge base of rules and weights for automated reasoning. It receives streaming data of feature vectors and facts from the environment. Based on the perception state estimation of the classifier and observed facts, the perceptual reasoner uses the knowledge base to infer an informative prior to compute the initial belief for the interaction. In the interaction, a policy suggests the best action for information collection based on the belief at each time step. Finally, the loop is closed by providing feedback containing labels and ground truth to the perceptual reasoner, training the classifier and learning the new weights of rules.

the *passive* perception and reasoning about the current world state. On the other hand, the agent can plan actions to actively facilitate lane changing, such as using turn signals to request space, and slowing down to find room for the lane change. The data collected from the interaction experience with other vehicles can be used for learning purposes. We use **interactive learning** to refer to the *active* behavior generation and learning process. The "perceptual reasoning" and "interactive learning" components are the two building blocks of this work. We use urban driving scenarios for demonstration and evaluation purposes in this thesis.

In this thesis, we develop a knowledge-based SDM algorithm, called *perceptual reasoning and interactive learning* (**PERIL**), as shown in Figure 1.1. We use a perceptual reasoner that consists of a deep supervised learning classifier and a knowledge base of logic rules associated with weights. The perceptual reasoner takes as input

streaming data from on-board sensors and observable facts, such as current time and weather. The contextual information is processed together to compute a distribution representing the current world state estimation. The distribution is then provided to the interaction component as an informative prior to guide its action selections toward achieving long-term goals. PERIL learns from both contextual knowledge, e.g., people are less cooperative in rush hours, and data gathered at runtime to close the perceive-reason-act loop, which identifies the **main contribution** of this thesis. Our approach has been extensively evaluated in urban driving scenarios in simulation using CARLA, a widely used autonomous driving simulation platform [5]. In comparison to competitive baselines [4, 6], we found PERIL improves the agents' overall performance by increasing cumulative rewards and reducing interaction costs.

2 Related Work

This thesis is about incorporating perceptual reasoning and interactive learning into a sequential decision making framework for autonomous driving behaviors. We discuss research topics that are relevant to this work.

Researchers have developed methods that incorporate human knowledge in declarative forms into planning under uncertainty frameworks [7, 2, 8, 3, 4]. There are other works that studied how human knowledge can be used to improve the performance of reinforcement learning (RL) agents [9, 10, 11, 12, 13, 14, 15]. A survey paper summarized research on knowledge-based sequential decision making [16]. Those methods use a knowledge base that cannot be updated as the agent becomes more experienced. In comparison, PERIL learns to reason about contextual knowledge, producing agent behaviors that are robust to imperfect knowledge.

Robots, including autonomous vehicles, that operate in the real world require the simultaneous capabilities of perception for estimating the current world state, and planning to achieve long-term goals [17, 18]. It is a common practice that the perception component outputs the current world state in a symbolic form to the planning component [19, 20, 21]. There is recent research from the literature that tightly integrates the perception and planning components [22, 23, 24]. There is

the survey paper on interactive perception that summarized relevant research [25]. They used machine learning techniques, e.g., a deep neural network, to estimate the current world state. What is passed to the planning component includes not only the current state in symbolic forms, but also the (un)reliability information. Also, a recent research developed an approximate algorithm to help the agent choose a subset of exogenous state variables to reason about when planning and planning in such a reduced state space can often be significantly more efficient than planning in the full model [26]. PERIL shares the same spirit with the above-mentioned methods by learning complex representations for estimating the current world state. Beyond that, PERIL leverages contextual knowledge from domain experts to refine the output from neural networks (CNNs in our case) before passing it along to the planning component.

Similar to the other types of autonomous robots, autonomous vehicles need to plan their behaviors under partial observability [27]. More specifically, the on-board sensors cannot provide a global view of the environment, and the vehicles need to estimate the current world state based on the streaming data collected over time. POMDPs are well suitable for planning behaviors under partial observability [1], and have been used in planning for autonomous vehicles [6, 28, 29, 30]. For instance, Wray et al. used POMDPs to reason at the times when the perception data is limited, but their approach does not leverage any contextual knowledge for reasoning.

In line with those methods, we use POMDPs for planning autonomous driving behaviors. Researchers also leveraged declarative knowledge for visual reasoning about perceptual data in autonomous driving (e.g, reasoning about pedestrians not being

detected temporarily by the sensors) [29], but their system can not learn to improve the reasoning. Within the context of autonomous driving, the uniqueness of this work comes from PERIL's perceptual reasoning component that learns to reason about contextual knowledge for state estimation.

Work closest to this research is an algorithm called LCORPP [4] that learns from data and reasons about human knowledge to estimate the world state. LCORPP was an extension of CORPP [2] that combined logical-probabilistic reasoning and probabilistic planning. Sharing the same spirit with CORPP, there were other works that proposed to use knowledge to guide planning under uncertainty including connecting logical reasoning and probabilistic planning [31] and using action knowledge to guide probabilistic planning [32]. LCORPP used LSTM [33] for sequence classification, and P-log [34] for representing and reasoning about contextual knowledge. Other than a different application domain, the main difference from that work is that PERIL is able to learn from the interaction experience to improve its reasoning capability, using Markov logic networks [35, 36]. To the best of our knowledge, PERIL is the first work that learns to use human knowledge within a sequential decision making framework.

3 Background

In this chapter, we summarize three key techniques used in this thesis: convolutional neural networks, Markov logic networks, and partially observable Markov decision processes.

3.1 Convolutional Neural Networks (CNNs)

A CNN is comprised of convolutional layers followed by fully connected layers as in a standard multilayer neural network [37]. The basic building blocks of CNN consist of convolutional, pooling, activation, and fully-connected layers. In a convolutional layer, a filter is passed over the 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). We use CNNs for the perception of road conditions in this thesis.

3.2 Markov Logic Networks (MLNs)

Markov networks are undirected cyclic probabilistic graphical models where each edge has a potential function [35, 36]. MLNs are a template for building Markov networks.

They are first-order knowledge bases with a weight attached to each rule. A first-order logic knowledge base is a set of hard constraints on the set of possible worlds: if a world violates even one formula, it has zero probability. The basic idea in MLNs is to soften these constraints: *When a world violates one formula in the knowledge base it is less probable, but not impossible.* Each formula has an associated weight that reflects how strong a constraint it is.

An MLN program is a set of pairs (F_i, w_i), where F_i is a formula in first-order logic and w_i is a real number that specifies the weight of the formula. Learning in MLNs can be done using the following equation:

$$\frac{\partial log P_w(X = x)}{\partial w_i} = n_i(x) - \sum_{x'} P_w(X = x') n_i(x')$$

where the sum is over all possible databases x', and $P_w(X = x')$ is $P(X = x')$ computed using current weight vector $w = (w_1, ..., w_i, ...)$ and $n_i(x)$ is the true groundings in data x. In this thesis, we use MLNs to enable an SDM agent to not only reason with human knowledge, but also learn to improve its reasoning capability from experience.

3.3 Partially Observable MDPs (POMDPs)

SDM can be achieved via Markov decision process (MDP) methods. When the environment is not fully observable, we can use POMDPs that generalize MDPs by assuming partial observability of the current state [1]. A POMDP is 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 a discount factor that determines the planning horizon.

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. Solving a POMDP produces a policy that maps the current belief state distribution to an action toward maximizing long-term utilities.

In this chapter, we describe the key models and techniques that are used in our framework. Next, we present our problem statement, including the variables that we use for world modeling in the PERIL algorithm and system.

4 Problem Statement

In this chapter, we present the problem for the PERIL (*perceptual reasoning and interactive learning*) agent. We first define the problem domain by the tuple below:

$$\langle \Theta, \underbrace{E, F, H, \overbrace{Q}^{\mathbf{V}^I}}_{\mathbf{V}^R}, V, A, T, Z, O \rangle.$$

The agent is provided with contextual knowledge Θ, a finite set of first-order logical statements (rules). Θ is about a finite set of variables $\mathbf{V}^R = F \cup E \cup H \cup Q$ as shown in Figure 4.1, where F, E, H, and Q are sets of fact, evidence, hidden, and query variables respectively. V is the set of unobservable variables for interaction, their values are initialized based on a uniform distribution. Interaction variables $\mathbf{V}^I$ consists of query and unobservable variables ($\mathbf{V}^I = Q \cup V$). The agent is provided with a finite set of actions A that the agent can perform. T is a transition function: $T(s, a, s') = Pr(s'|s, a)$, where $s, s' \in S$ is the factored space specified by $\mathbf{V}^I$. Z is an observation set, and O is an observation function: $O(s, a, z) = Pr(z|s, a)$.

Figure 4.1 depicts the two sets of variables $\mathbf{V}^R$ and $\mathbf{V}^I$ for reasoning and interaction respectively, and their overlap on Q. Variable sets E, F, H, and Q are mutually exclusive. Logical reasoning with Θ produces the combinatorial possible settings of $\mathbf{V}^R$ that are consistent to the logical statements. The query variables are shared by

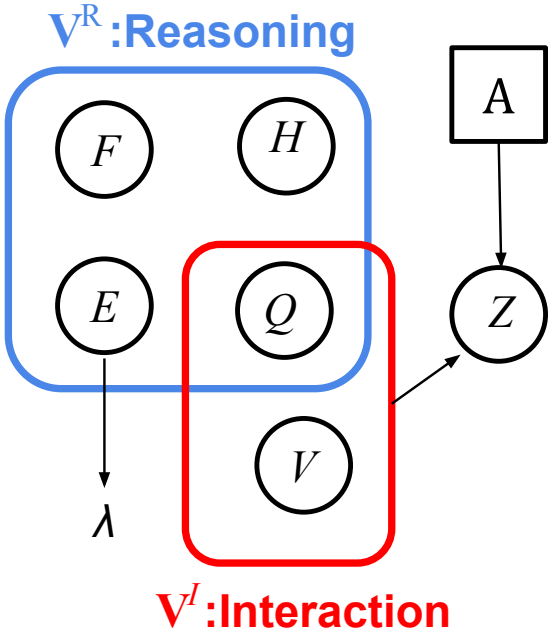

Figure 4.1: Domain variables and dependencies.

both interaction and reasoning variables ($Q = \mathbf{V}^R \cap \mathbf{V}^I$, and $Q \neq \emptyset$). The agent cannot directly observe the variables of $H \cup Q \cup V$. Values of variables F can be directly collected from the world. Variables E are estimated via streaming data λ.

4.1 Assumptions

We make assumptions here to categorize the variables and help the readers understand their relationships.

- Variables that are not observable:
  - variables H are latent in the reasoning
  - variables V are latent in the interaction
- Variables that are directly observable:

  - fact variables F are directly observable from the environment for reasoning

- Variables that can be estimated or inferred
  - variables E can be estimated from streaming data λ with supervised learning model during perceptual reasoning
  - variables Q can be inferred from E, F and H with relational learning model in reasoning

- Input and output of the reasoning:
  - the input to the reasoning are streaming data λ, fact variables F, hidden variables H
  - the output of the reasoning are query variables Q

- Input and output of the interaction:
  - the input to the interaction are query variables Q, latent variables V
  - the output of the interaction are actions A
  - the belief of state space $\mathbf{V}^I$ $(Q \cup V)$ is updated based on the observation Z after the action of A is taken

In episode i and at execution time t, the agent receives $z_t \in Z$, and sensory readings λ_t, where λ_t^i is a perception of E_i. After each episode i (i.e., when a terminal state is reached), values of $\mathbf{V}^R \cup \mathbf{V}^I$ are provided by human for training purposes. The task is specified by a reward function $R(s, a) \to \mathbb{R}$. The objective is to compute a policy π for the robot to choose actions at each time step toward maximizing its

expected future discounted reward, $\mathbf{E}\left[\sum_{t=0}^{\infty} \gamma^t r_t\right]$, where γ is a discount factor, and r_t is the reward received at time t. Given the stated domain and variables, PERIL is able to solve it as learning through the perceptual reasoning and interaction with the environment.

Next, we introduce our PERIL algorithm in Chapter 5 (the main contribution of this thesis), followed by a detailed instantiation in Chapter 6, including how the above-mentioned variables are implemented in a real-world problem.

5 Algorithm

In this chapter, we present PERIL, a novel algorithm that addresses problems where the agent needs to learn to reason about contextual knowledge for sequential decision-making. A PERIL agent perceives the environments using supervised learning, reasons over domain variables using contextual knowledge, and generates interaction behaviors using a decision-theoretic planning approach. PERIL's reasoning capability is enhanced via relational learning as the agent is more experienced over time.

Algorithm 1 describes PERIL, the key contribution of this research. The input includes a domain description, a problem description specified by reward function R, and parameter N for batch-based learning. Implementing PERIL systems requires software tools for relational learning (Lrn^R) and supervised learning (Lrn^S), as well as MLN and POMDP systems (Sol^R and Sol^P). Lines 1-2 are for initialization, where Φ and Ψ are for storing data for supervised learning and relational learning respectively.

There are three loops in PERIL. Each iteration of the **outer while loop** (Lines 4-21) corresponds to one batch where supervised learning and relational learning are activated once (Lines 19-20). The nested **for-loop** (Lines 5-18) includes N iterations

– each corresponding to a sequence of perceptual reasoning (Lines 5-9), interaction (Lines 10-14), and data augmentation (Lines 15-17). In perceptual reasoning, the agent infers the query variables Q, using the logical weighted rules (Θ, W), and the direct observations (f) or the estimated observations (e) from the world (Lines 6-8). Using the union of inferred Q and V, PERIL builds the initial prior belief b (Line 9), where the posterior is calculated in the inner interaction while-loop (Lines 10-14). Once the interaction loop is done, two datasets are augmented with the newly collected data instances. The **inner while loop** (Lines 10-14) corresponds to one episode, where the agent takes one action a, makes an observation z, and updates belief b. The actions in this loop are suggested by policy π calculated by Sol^P. PERIL is a lifelong learning algorithm for SDM, and does not have a termination condition.

PERIL enables an agent to learn from interaction experience to improve its capabilities of reasoning with contextual knowledge from people, and planning under uncertainty. To the best of our knowledge, no existing algorithm supports this "learning to reason and plan" capability. Next, we describe a full instantiation of PERIL, as applied to an urban driving domain.

Algorithm 1 PERIL

Ensure: Domain $\langle \Theta, E, F, H, Q, V, A, T, Z, O \rangle$, problem R, and parameter N

Require: MLN system Sol^R, POMDP system Sol^P, relational learning system Lrn^R, and supervised learning system Lrn^S

Initialize dataset $\Phi \leftarrow \emptyset$; dataset $\Psi \leftarrow \emptyset$; $\pi \leftarrow$ random; classifier $C \leftarrow$ random
Initialize weights W, $w \leftarrow 1.0$, each w corresponds to $\theta \in \Theta$
Compute π using Sol^P for POMDP: $(Q \cup V, A, T, Z, O, R)$
while true **do** {No termination condition – lifelong learning}
    for $i \in [0, N-1]$ **do**
        Receive λ, and f from the world, where f is a vector of F
        $e \leftarrow C(\lambda)$; e is a vector and includes the values of E
        $Pr(Q) \leftarrow Sol^R(\Theta, W, f, e)$
        Compute distribution b over state set $S = Q \cup V$ using $Pr(Q)$ and uniform distributions over variables V
        while s is not a terminal state **do**
            Select action $a \leftarrow \pi(b)$ and execute a
            Make an observation z
            Update b based on a and z
        end while
        Collect ground truth values $\mathbf{v}^R = \{\hat{e}, \hat{f}, \hat{h}, \hat{q}\}$
        Augment dataset: $\Phi \leftarrow \Phi \cup \{\lambda : \hat{e}\}$
        Augment dataset: $\Psi \leftarrow \Psi \cup \{\mathbf{v}^R\}$
    end for
    $C \leftarrow Lrn^S(\Phi)$ {Supervised learning}
    $W \leftarrow Lrn^R(\Theta, \Psi, W)$ {Relational learning}
end while

6 Instantiation

We use CARLA (Car Learning to Act), an open-source simulation platform developed for autonomous driving research [5]. A CARLA environment consists of 3D models of vehicles, traffic signs, buildings, and pedestrians. It supports training, prototyping, and validation of autonomous driving models,including both perception and control. Figure 6.1 shows a scenario where our vehicle is in a lane merging process. Next, we provide technical details of each component of our PERIL framework.

(a) Cooperative

(b) Not cooperative

Figure 6.1: Two situations of an urban driving scenario in CARLA simulation where an ego vehicle is merging left. (a) The vehicle on the left is cooperative and yields the right of way. (b) The vehicle on the left is not cooperative

6.1 CNNs for Perception:

C is our classifier that takes as input raw sensory data (3D Lidar sensory readings in our case), and outputs the road condition. We use CNN to build classifier C, and to process streaming data λ from Lidar sensors. Figure 6.2 shows how C is constructed in our instantiation. The 3D sensory readings are first projected to 2D space. Then

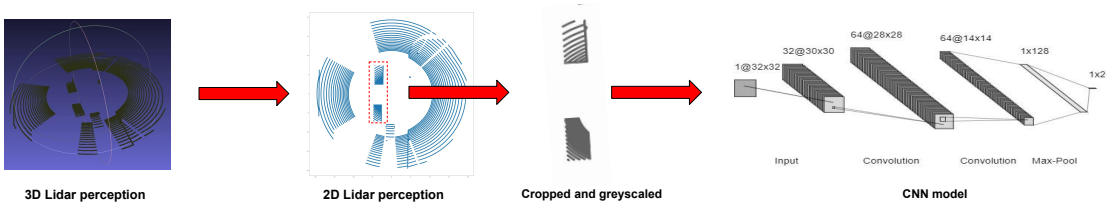

Figure 6.2: An overview of the perception component where the vehicle receives raw data from the Lidar sensor. The sensory readings are projected to 2D space, and converged into an image. Finally, a CNN outputs if the desired lane is sensed crowded.

the road area is cropped out to generate a 2D image, which is fed into CNNs for classification. The output of classifier C is saved in variable $CarsDetected$ ($true$ or $false$). In our domain, E includes only one element: $E = \{CarsDetected\}$.

6.2 MLNs for Logical Probabilistic Reasoning:

MLN is the core of our perceptual reasoning component as it is the part that can learn to reason with contextual knowledge. Specifically, we use the Alchemy package for statistical relational learning and probabilistic logic inference, based on the Markov logic representation. Our MLN-based reasoner includes five variables: $Weather$, $Time$, $Crowded$, $CarsDetected$, and $Cooperative$. Among them, $Weather$ and $Time$ are fact variables: $F = \{Weather, Time\}$. The weather can be $Sunny$ or $Rainy$, and the time is either $Busy$ or $Normal$, which is used for reasoning about traffic condition. $Crowded$ and $Cooperative$ are query variables: $Q = \{Crowded, Cooperative\}$. $H = \emptyset$. There is one evidence variable $E = \{CarsDetected\}$. Other drivers' behaviors are simplified to a binary variable of $Cooperative$ with a domain of $true$ or $false$.

An MLN program includes a set of first-order logical statements, where each is associated with a weight. We use MLN to build our logical probabilistic reasoner Sol^R

and relational learning system Lrn^R. First-order logic rules Θ form the declarative domain knowledge base.

We have three rules in our knowledge base of MLN as indicated below:

- `Time(+t,s)`$\rightarrow$ `Crowded(+c,s)`
- `Crowded(+c,s)`$\rightarrow$ `CarsDetected(+d,s)`
- `Weather(+w,s)` $\wedge$ `Crowded(+c,s)`$\rightarrow$ `*Cooperative(s)`

6.2.1 MLN Syntax

We briefly introduce the MLN syntax here so as to better explain our knowledge base. The syntax for logical connectives in MLN is as follows: "!" for "not", "$\wedge$" for "and", "$\vee$" for "or" and "$\rightarrow$" for "implies". The "+" operator makes it possible to learn the weight for the each value of the variable. When a variable in a formula is preceded by "+", a separate weight is learned for each formula obtained by grounding that variable to one of its values. If multiple variables are preceded by "+", a weight is learned for each combination of their values. For example, for the first rule, separate weights are learned for the following combinations:

- `Time(Busy,s)`$\rightarrow$ `Crowded(True,s)`
- `Time(Normal,s)`$\rightarrow$ `Crowded(True,s)`
- `Time(Busy,s)`$\rightarrow$ `Crowded(False,s)`
- `Time(Normal,s)`$\rightarrow$ `Crowded(False,s)`

When predicates in a formula are preceded by "*" as shown in the third rule (*Cooperative(s)), the MLN considers all possible ways in which * can be replaced by !. For example, for the third rule the separate weights for the combinations would be like:

- `Weather(+w,s)` $\wedge$ `Crowded(+c,s)`$\rightarrow$ `Cooperative(s)`
- `Weather(+w,s)` $\wedge$ `Crowded(+c,s)`$\rightarrow$ `!Cooperative(s)`

6.2.2 Rules of Knowledge Base

Now we explain what each rule means and their logics. The first rule

`Time(+t,s)`$\rightarrow$ `Crowded(+c,s)`

indicates that the time implies the crowdedness of the road. If it is at busy time, it is likely the road is crowded. If it is at normal time, it is more likely that the road is not crowded. The second rule

`Crowded(+c,s)`$\rightarrow$ `CarsDetected(+d,s)`

states that, when the road is crowded, it is more likely that the ego vehicle can detect surrounding cars. The third rule

`Weather(+w,s)` $\wedge$ `Crowded(+c,s)`$\rightarrow$ `*Cooperative(s)`

states that the weather condition and the road crowdedness affects the surrounding vehicles (drivers) being cooperative or not. All rules Θ are associated with weights. During weight (relational) learning, each rule is converted to conjunctive normal form, and a weight is learned for each of its clauses. It should be noted that those are "commonsense" rules that are normally correct but not always. MLNs are well suited for learning to reason with those rules. We then use the input of H,E and F to infer the value of Query variables Q from MLN.

6.3 POMDPs for Planning under Uncertainty

We use POMDPs to construct a probabilistic planner for active information gathering, and goal achievement. $S : Q \times V \cup \{term\}$ is the state space, where $term$ is a terminal state that identifies the end of an episode. $V = \{RoomAvailable\}$. $RoomAvailable = true$ means that there is room available in the desired lane for the ego vehicle's lane merging behavior. $Q = \{Crowded, Cooperative\}$ which are the query variables of reasoning as the output of MLN in the previous section. $Crowded = true$ and $Cooperative = true$ means that the road is crowded and the driver on the left side is cooperative. We consider three behaviors in our action space: $A = \{signal,\ move,\ merge\}$, where we assume the vehicle can only merge to one side of the road (say left). $signal$ means that the vehicle uses turn signal to indicate its intention to merge. $move$ means that the vehicle adjusts its position to get prepared for lane changing, which is also useful for communicating its intention to the other drivers. Intuitively, after the vehicle is confident that there is room in the desired lane, and the other drivers are cooperative, the vehicle should take the $merge$ action.

We use transition function $T(s, a, s') = Pr(s'|s, a)$ to model how action a leads the transition from s to s'. Actions except for $merge$ have different costs (a small negative value). Action $merge$ causes either a big reward or a big penalty (a big negative value), depending on the road condition (values of $Cooperative$, and $RoomAvailable$). For instance, if $Cooperative = false$ or $RoomAvailable = false$, action $merge$ will result in a big penalty. Action costs, success reward, and failure penalty are modeled in reward function $R(s, a)$.

The observation set is $Z : \{true, false, na\}$. We use the observation function $O(s, a, z) = Pr(z|s, a)$ to describe the perception model of the vehicle. For instance, when $Cooperative = true$, there is 0.7 probability that the vehicle observe $true$ (the other drivers are cooperative).

7 Experiments

We have conducted experiments using the CARLA simulator to evaluate the key hypothesis that learning to reason about domain knowledge improves the agent's performance within the sequential decision-making context. We have compared PERIL with the following baselines.

- **LCORPP** is a baseline method that uses supervised learning for perception, and automated reasoning to guide a probabilistic planner [4]. LCORPP's knowledge base is hardcoded, so it cannot learn to reason about knowledge.
- **PERIL w/o POMDP** is the same as PERIL except that the action policy is manually crafted: the vehicle takes up to two *signal* actions (depending on the confidence on state estimation), then a *move* action, and *merge*.
- **POMDP-LC** is a classic POMDP-based approach for planning lane changing behaviors [6], which includes neither supervised learning nor relational learning.

7.1 Experiment Setup:

In each trial, we first spawn our ego vehicle, which is tasked to merge to the left lane. We set the range of Lidar sensor to $20m$. We sequentially spawn M vehicles on the left lane ($0 \leq M \leq 8$ in our case) within an area of $radius = 20m$ around the ego

vehicle. If a vehicle has any contact with an existing one, then this vehicle is moved and re-spawned.

We annotated the Lidar sensory data: if there exist two vehicles in the left lane that are at most $10m$ away from each other, then a Lidar instance is labeled *true*, i.e., $CarsDetected = true$. Otherwise, the label is $false$. Fact variables $Time$ and $Weather$ were sampled uniformly. $Crowded$ and $Cooperative$ were sampled using the Markov network of our MLN program. For instance, if $Time = normal$, then there is probability 0.7 that $Crowded = true$.

We have added perception noise into the observation model. For instance, the vehicle's observation is correct in 0.7 probability. The costs of *signal* and *move* actions are $10s$ and $15s$ respectively. Successful and unsuccessful trials receive 100 and -100 reward respectively.

We used Alchemy for MLN-based relational learning and logical probabilistic reasoning.[1] POMDPs were solved using an off-the-shelf solver [38]. We used PyTorch [39] for training the CNNs.

7.2 Experimental Results

Every data point in our figures is an average of 4,000 trials, evenly distributed into 5 runs. We evaluated the mean values of the 5 runs for each data point, and used the 5 mean values to generate the standard errors.

Figure 7.1 shows the results of comparing PERIL with three baseline methods. We see that PERIL achieved the highest cumulative reward on average, and required the

[1] https://alchemy.cs.washington.edu/

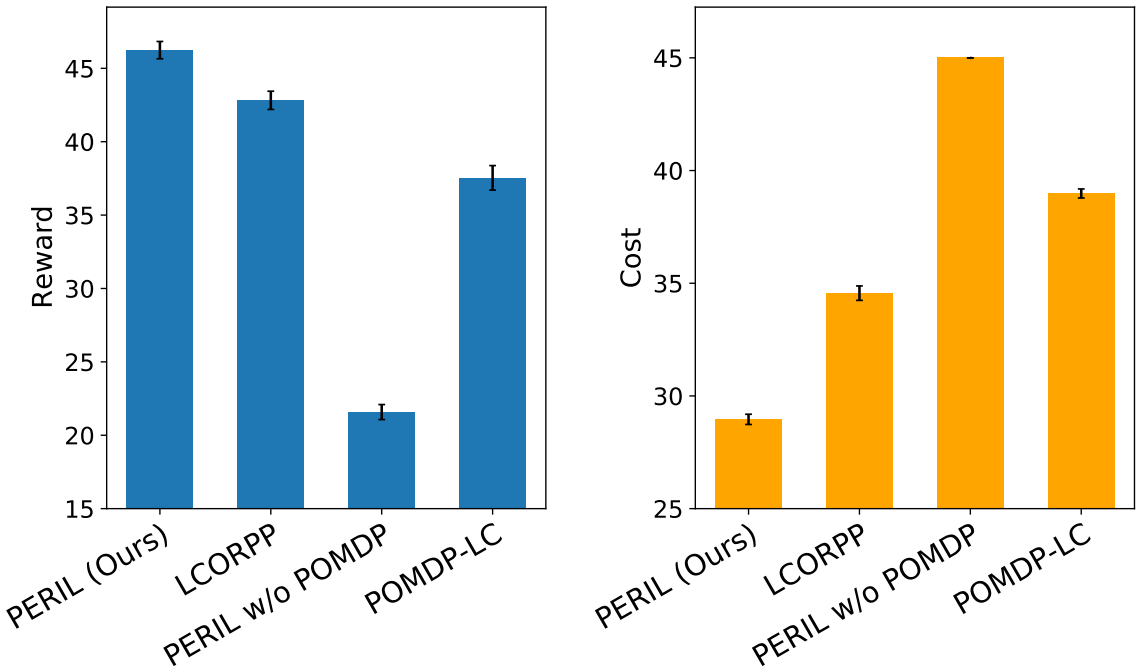

Figure 7.1: PERIL performed better than the baselines in both overall reward, and interaction cost.

lowest interaction cost on average. The LCORPP baseline produced the second best performance in both reward and cost, which indicates the usefulness of perceptual reasoning. PERIL uses MLN to learn to reason about contextual knowledge, which contributes to the best performance among the four methods. All methods produced an average success rate between 0.87 to 0.89, where we did not observe statistically significant difference among the methods. A successful merge is when the ego vehicle merges left in the presence of enough room and vehicle cooperation. An unsuccessful merge in our setup functions like a risky situation in practice, and does not indicate a collision, because autonomous vehicles (or human drivers) have collision-avoidance mechanisms, which are not considered in our experiments. Results here support our key hypothesis that PERIL outperforms baseline methods with higher rewards and lower costs.

Table 7.1: The performances of PERIL and baselines in reward and cost for vehicles of different observation reliability levels. Bold font indicates the max (or min) value of a "Reward" (or "Cost") column, and the numbers in italic font indicate statistically significant improvement over the other methods.

Algorithm	POMDP Observation with different reliabilities					
	0.7		0.8		0.9	
	Reward	Cost	Reward	Cost	Reward	Cost
PERIL	***46.5*** (0.5)	***28.7*** (0.3)	***55.3*** (0.4)	***28.3*** (0.4)	***64.1*** (0.7)	***27.1*** (0.3)
LCORPP	43.5 (1.0)	34.1 (0.3)	53.6 (0.7)	32.9 (0.1)	62.4 (0.4)	31.0 (0.2)
PERIL w/o POMDP	20.9 (0.9)	45.0 (0.0)	20.5 (1.1)	45.0 (0.0)	20.2 (1.0)	45.0 (0.0)
POMDP-LC	40.2 (0.8)	39.7 (0.1)	52.1 (0.5)	35.6 (0.2)	62.4 (0.4)	32.3 (0.2)

Table 7.1 shows the performances of PERIL and baselines under different perception capabilities e.g., $reliability = 0.5$ indicates that the observation is random. In reality, the *reliability* of observation of the ego vehicle should be much higher than a random, so we demonstrate the results starting from $reliability = 0.7$. The results suggest that PERIL outperformed the baselines as long as the vehicle's perception capability is reasonably good (≥ 0.7).

7.3 Ablation Study

We did an ablation study to evaluate the importance of the two learning components in PERIL (supervised learning and relational learning). The results are shown in Figure 7.2. Our first observation is that PERIL performed better than its two ablations in both overall reward, and interaction cost, except for the very early learning phase. Another observation is that relational learning plays an important role in the PERIL system. When relational learning was disabled, there was significant increase in interaction cost, in comparison to the ablation with supervised learning removed. This is potentially because the MLN-based reasoner can learn to "compensate" for the missing perception component.

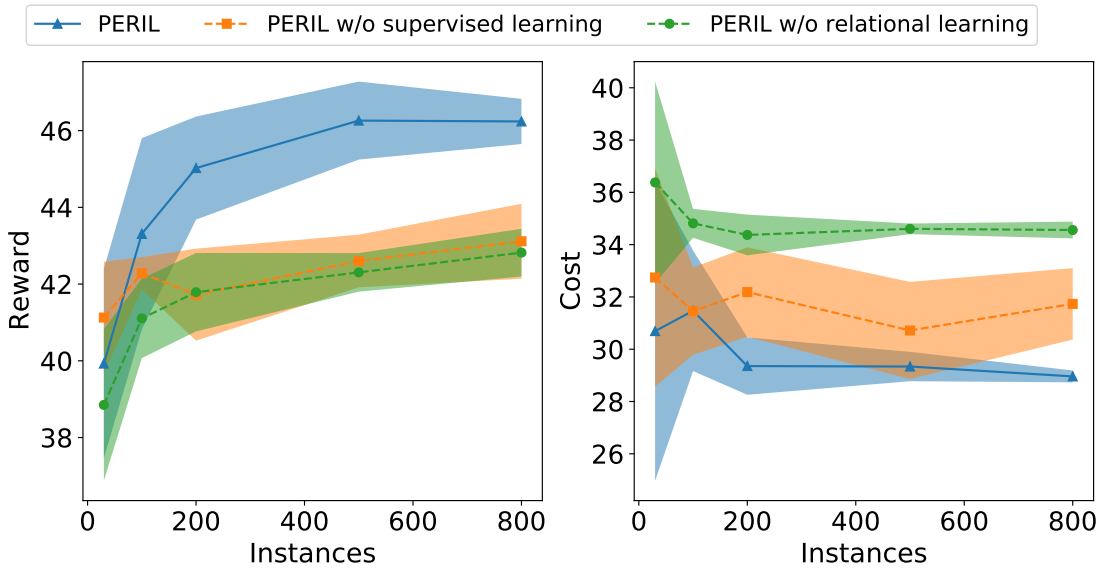

Figure 7.2: PERIL performed better than its two ablative versions as more data instances were provided for training.

7.4 Illustrative Example

Figure 7.3 shows an example trial. The vehicle first collected a "fact" that it was a rainy day at a busy time. The vehicle received streaming data, and the CNN classifier outputs that $CarsDetected = true$, meaning that the left lane is occupied by at least one vehicle. Reasoning with contextual knowledge about weather and time, our vehicle believed that it was likely the road was crowded and the other drivers were less cooperative. The ego vehicle then used our MLN-based reasoner to perform probabilistic inference, and found that $Pr(Crowded = true) = 0.995$, and $Pr(Cooperative = false) = 0.970$. Those probabilities were used to initialize the POMDP belief b. With the initial belief of the current state and sequential observations, the ego vehicle repeatedly selected actions as shown in Figure 7.4. After two *signal* and two *move* actions, the ego vehicle successfully completed a merging

lane task. [2]

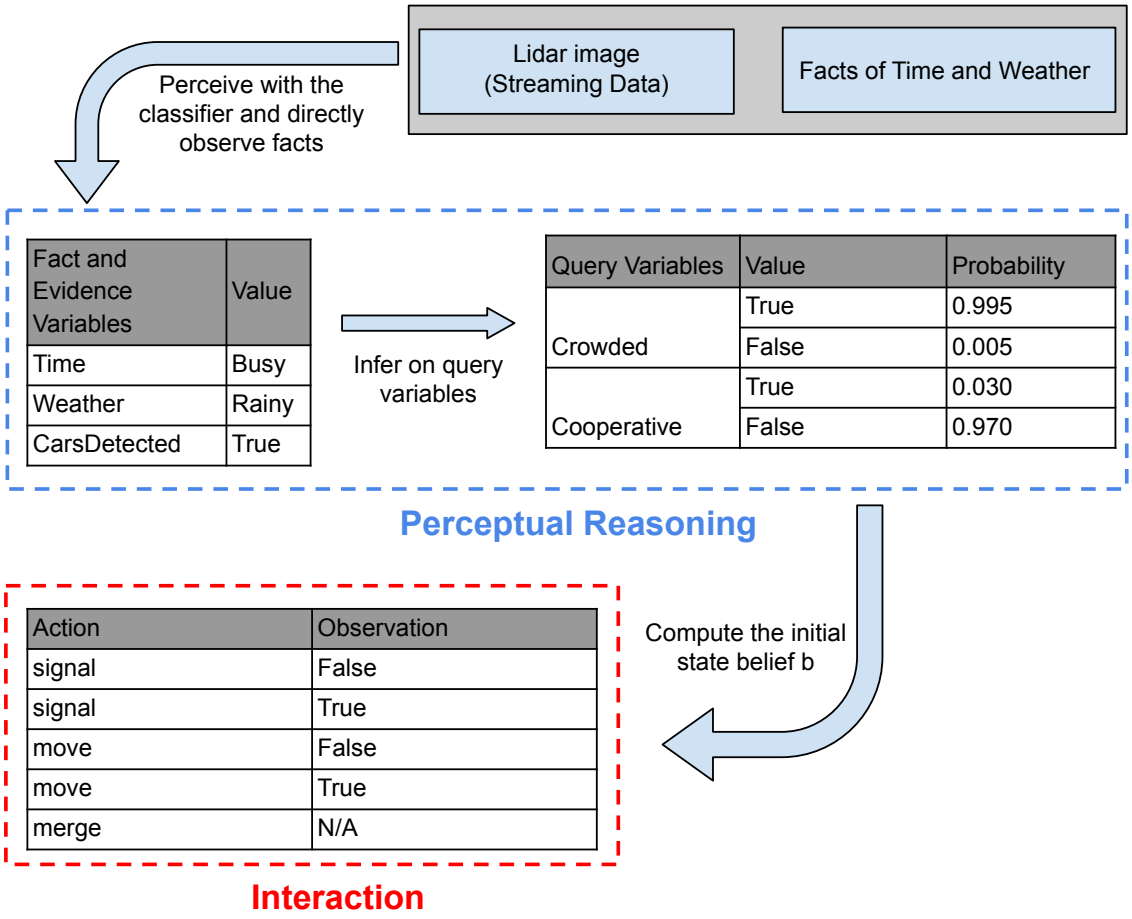

Figure 7.3: An illustrative example of PERIL.

[2] A demo video is uploaded as part of this submission.

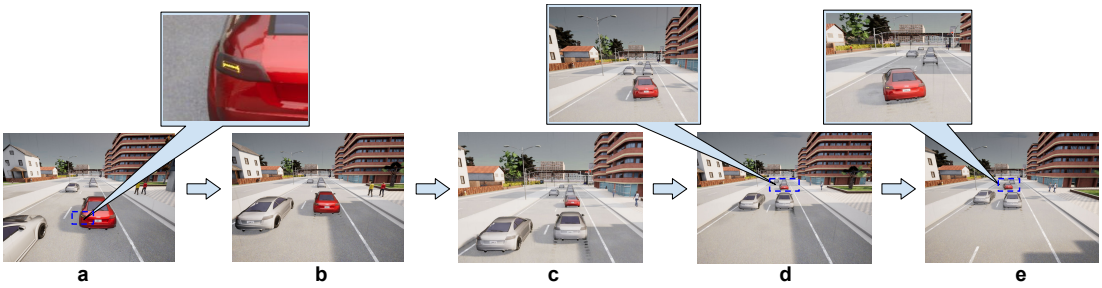

Figure 7.4: The ego vehicle took a sequence of actions in the interaction process to successfully merge left. (a) The ego vehicle intended to merge left. It turned on the left signal. (b) The surrounding vehicle on the left was not cooperative at first. The ego vehicle kept left blinking. (c) The surrounding vehicle on the left became cooperative, and the ego vehicle started to move left. (d) The ego vehicle kept moving left and found room in the left lane. (e) The ego vehicle successfully merged left.

8 Conclusions and Future Work

In this thesis, we develop an algorithm called PERIL that learns to reason with contextual knowledge for sequential decision making. PERIL uses convolutional neural networks for perception, Markov logic networks for reasoning, and partially observable Markov decision processes for planning under uncertainty. We have extensively evaluated PERIL in urban driving scenarios. Results suggest that PERIL outperformed competitive baselines, as well as its own ablations, in both overall reward and interaction cost.

Currently, the vehicle learns to perceive the environment (road condition) from data, and learns to improve its reasoning capability using MLN. One direction of future work is to replace the POMDP-based planner with a reinforcement learning component. By doing that, the vehicle will be able to learn to select actions from its task-completion experience. Another direction is to actively acquire knowledge from people to avoid hand-coding rules for MLNs.

Bibliography

[1] Leslie Pack Kaelbling, Michael L Littman, and Anthony R Cassandra. "Planning and acting in partially observable stochastic domains". In: *Artificial intelligence* (1998).

[2] 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.

[3] 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.

[4] 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.

[5] Alexey Dosovitskiy et al. "CARLA: An open urban driving simulator". In: *Conference on robot learning*. PMLR. 2017, pp. 1–16.

[6] S. Ulbrich and M. Maurer. "Probabilistic online POMDP decision making for lane changes in fully automated driving". In: *ITSC 2013*. 2013, pp. 2063–2067. DOI: 10.1109/ITSC.2013.6728533.

[7] 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.

[8] Marc Hanheide et al. "Robot task planning and explanation in open and uncertain worlds". In: *Artificial Intelligence* 247 (2017), pp. 119–150.

[9] 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.

[10] Fangkai Yang et al. "PEORL: integrating symbolic planning and hierarchical reinforcement learning for robust decision-making". In: *International Joint Conferences on Artificial Intelligence Organization*. 2018.

[11] Rodrigo Toro Icarte et al. "Using reward machines for high-level task specification and decomposition in reinforcement learning". In: *International Conference on Machine Learning*. PMLR. 2018, pp. 2107–2116.

[12] Yuqian Jiang et al. "Task-motion planning with reinforcement learning for adaptable mobile service robots". In: *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2019, pp. 7529–7534.

[13] Harsha Kokel et al. "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.

[14] Daniel Neider et al. "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.

[15] 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.

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

[17] Nils J Nilsson. *Shakey the robot*. Tech. rep. SRI INTERNATIONAL MENLO PARK CA, 1984.

[18] Sebastian Thrun, Wolfram Burgard, and Dieter Fox. *Probabilistic Robotics*. MIT Press, 2005.

[19] Piyush Khandelwal 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.

[20] Manuela M. Veloso. "The Increasingly Fascinating Opportunity for Human-Robot-AI Interaction: The CoBot Mobile Service Robots". In: *ACM Transactions on Human-Robot Interaction* 7.1 (May 2018). DOI: 10.1145/3209541. URL: https://doi.org/10.1145/3209541.

[21] Wei Shuai and Xiao-ping Chen. "KeJia: towards an autonomous service robot with tolerance of unexpected environmental changes". In: *Frontiers of Information Technology & Electronic Engineering* 20.3 (2019), pp. 307–317.

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

[23] Michelle A Lee et al. "Making sense of vision and touch: Learning multimodal representations for contact-rich tasks". In: *IEEE Transactions on Robotics* (2020).

[24] Chen Wang et al. "SwingBot: Learning Physical Features from In-hand Tactile Exploration for Dynamic Swing-up Manipulation". In: *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2020.

[25] Jeannette Bohg et al. "Interactive perception: Leveraging action in perception and perception in action". In: *IEEE Transactions on Robotics* 33.6 (2017), pp. 1273–1291.

[26] Rohan Chitnis and Tomás Lozano-Pérez. "Learning compact models for planning with exogenous processes". In: *Conference on Robot Learning*. PMLR. 2020, pp. 813–822.

[27] Haoyu Bai et al. "Intention-aware online POMDP planning for autonomous driving in a crowd". In: *2015 ieee international conference on robotics and automation (ICRA)*. IEEE. 2015.

[28] Kyle Hollins Wray, Stefan J. Witwicki, and Shlomo Zilberstein. "Online Decision-Making for Scalable Autonomous Systems". In: *International joint conference on artificial intelligence*. 2017, pp. 4768–4774. DOI: 10.24963/ijcai.2017/664. URL: https://doi.org/10.24963/ijcai.2017/664.

[29] Jakob Suchan, Mehul Bhatt, and Srikrishna Varadarajan. "Out of sight but not out of mind: An answer set programming based online abduction framework for visual sensemaking in autonomous driving". In: *arXiv preprint arXiv:1906.00107* (2019).

[30] Kyle Hollins Wray et al. "POMDPs for Safe Visibility Reasoning in Autonomous Vehicles". In: *2021 IEEE International Conference on Intelligence and Safety for Robotics (ISR)*. IEEE. 2021, pp. 191–195.

[31] 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.

[32] Mohan Sridharan et al. "REBA: A refinement-based architecture for knowledge representation and reasoning in robotics". In: *Journal of Artificial Intelligence Research* 65 (2019), pp. 87–180.

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

[34] 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.

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

[36] Pedro Domingos and Daniel Lowd. "Unifying logical and statistical AI with Markov logic". In: *Communications of the ACM* 62.7 (2019), pp. 74–83.

[37] Yann LeCun, Yoshua Bengio, et al. *Convolutional networks for images, speech, and time series, The handbook of brain theory and neural networks*. 1998.

[38] 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.

[39] Adam Paszke et al. "Pytorch: An imperative style, high-performance deep learning library". In: *Advances in neural information processing systems* 32 (2019), pp. 8026–8037.