Behavior Planning at Roundabouts

Aman Khurana

CMU-RI-TR-19-54

August 10, 2019

The Robotics Institute
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA

Thesis Committee:
John M. Dolan, CMU RI, (*Chair*)
David Held, CMU RI
Fahad Islam, CMU RI

*Submitted in partial fulfillment of the requirements
for the degree of Masters in Robotics.*

Abstract

Roundabouts or traffic circles represent a significant portion of unsignalized intersections commonly found in urban and rural roads due to their capability in managing significant traffic flow safely. Such circular intersections pose a specific challenge for autonomous or self-driving cars due to the variations in their geometric layout, difficulty in perception, increased interactions between the traffic participants, and possible driving strategies available to the drivers. This work investigates behavior planning approaches for a self-driving vehicle as a part of a hierarchical planning structure for such scenarios.

We present the benefits of using a POMDP formulation along with dividing the task into different stages (merging, traversal, and exit) to tackle the problem. Using recent advances in deep reinforcement learning we find that using recurrent elements with the given framework allows an autonomous vehicle to interact with other participants, make long-term decisions, account for perception errors, and safely navigate the roundabout. We compare these to traditional, rule-based methods and simple neural-network architectures like DQN. The model-free learning POMDP framework is further extended to include the agent's previous actions into the network architecture.

Additionally, we present multiple techniques to generalize policies across different traffic densities. The presented novel architecture involves explicitly encoding a continuous variable describing the non-stationary environment as an input to the network. We compare this to the hidden-mode method of dividing the problem into distinct modes of traffic densities and learning different policies for individual modes. These methods can also be extended to other intersection scenarios and/or to different deep-reinforcement learning formulations.

Acknowledgments

I would like first to thank my advisor, Dr. John Dolan, whose constant support, patience, and guidance have helped me navigate through the fog of research. His fearless approach to robotics and practical knowledge continue to captivate and inspire me. Thank you for giving me the freedom to explore, collaborate with others, to voice criticisms and to evolve as a researcher.

I am grateful to my thesis committee members for their invaluable feedback. Dr. David Held, thank you for your inputs in this work and other collaborations. Fahad Islam, thank you for your insightful and fundamental questions that helped in improving this work.

Robotics is a team effort, and I could not have asked for a better team than the people working along with John. Thanks especially to folks with whom I have had the opportunity to work with - (now Dr.) Chiyu Dong, Chen Fu, Zhiqian Qiao, Adam Villaflor, Yanjun Pan, Jing Zhao, Yeqiang, Chengfeng Zhao and Qin Lin. Thanks to the people involved in the CMU Assured Autonomy project- Edward Ahn, Dr. Stefan Mitsch, and others. Thanks to Emily Yunan and Jordan Ford for their work on the RC Car platform.

Thanks to my friends at CMU - Anne, Siddhant, Sowmya, Aayush, Yash, Pulkit - for navigating CMU with me, balancing academics with research, sharing lives and steadying the ship.

Finally, words cannot express my feelings for my family. Thanks, Mom, Dad and Brother, for your unconditional love, encouragement, and sacrifice - you have made us feel unbreakable. This achievement is as much yours as it is mine.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Overview

Autonomous or self-driving vehicles present an interesting look into the future of transportation due to their perceived advantages, including improved safety, reduced congestion, lower emissions, and greater mobility. Safe and efficient autonomous behavior in the presence of other participants on public roads is a challenging task. An autonomous vehicle must be programmed to do these things, unlike human drivers, who have the inborn ability of making decisions, perceiving the environment, extracting essential information and using past experiences.

Toward this end, a lot of work has been done for driving in lanes, negotiating intersections, etc., but only a limited focus has been given to unsupervised roundabouts or traffic circles, which are increasingly becoming popular in most countries as a means of regulating traffic flow.

1.2 The roundabout problem

Roundabouts or traffic-circles guide the traffic flow around a circular shape to avoid the need to make left turns. Compared to other traffic intersections roundabouts are capable of reducing the likelihood of collisions since the traffic flows in a circular direction. As per a 2015 survey [21], there are more than 5000 roundabouts in the

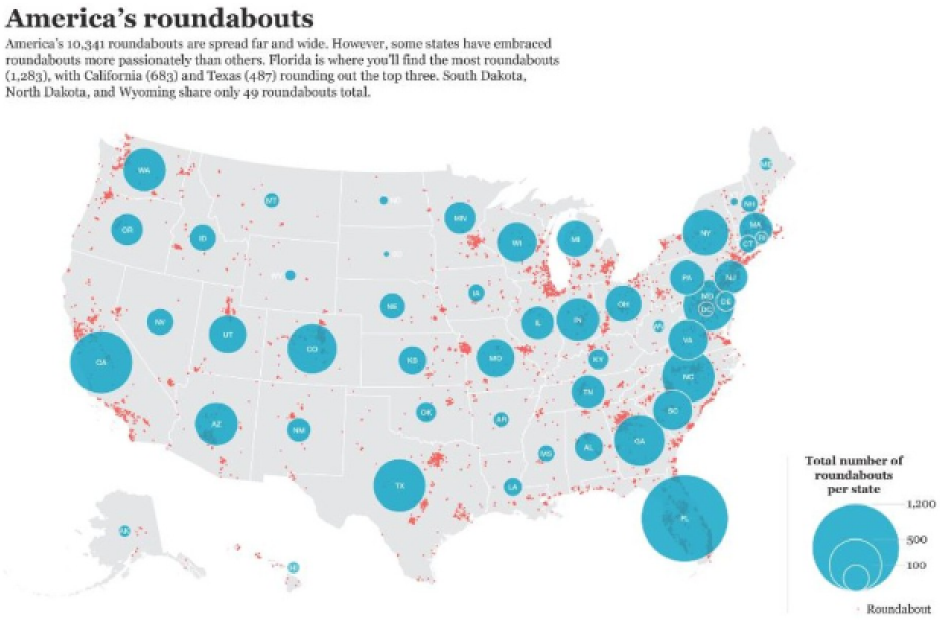
America's roundabouts
America's 10,341 roundabouts are spread far and wide. However, some states have embraced roundabouts more passionately than others. Florida is where you'll find the most roundabouts (1,283), with California (683) and Texas (487) rounding out the top three. South Dakota, North Dakota, and Wyoming share only 49 roundabouts total.
WA
OR
ID
MT
NV
CA
UT
CO
AZ
NM
TX
NE
KS
OK
MN
IA
MO
AR
LA
MS
WI
MI
IL
IN
OH
KY
TN
AL
GA
FL
SC
NC
VA
WV
PA
NY
NJ
MD
DE
DC
NH
MA
CT
RI
ME
AK
HI
Total number of roundabouts per state
1,200
500
100
Roundabout

Figure 1.1: Number of roundabouts and their distribution in United States as per [21]

USA alone, with the majority having been added over the last decade to better manage traffic within cities (see Fig. 1.1).

The majority of roundabouts in urban and rural settings do not have any traffic signals to manage the traffic flow and are commonly referred to as unsignalized or unsupervised roundabouts. As a result, a vehicle approaching the unsignalized roundabout yields to those already in the circle. Figure 1.2 presents an overview of two-lane, four-exit roundabout. Roundabouts also present a specific challenge in the complexity of driving behavior, high variance in road geometry (some have one lane while others may have multiple lanes with varying road orientations), and increased uncertainty in perception due to road geometry. It is crucial that autonomous vehicles exhibit a natural and social behavior on roundabouts for the safety and smooth flow of mixed traffic (where autonomous vehicles operate along with human-driven vehicles). Due to the need for consecutive maneuvers in a short time at roundabouts, conventional planning approaches lead to sub-optimal behavior of autonomous vehicles.

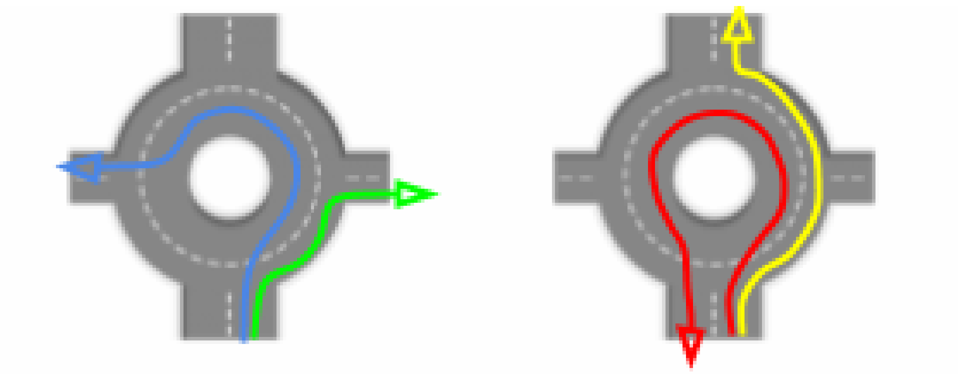

Figure 1.2: A bird's-eye view of a standard four-exit, double lane roundabout geometry. The vehicle trajectories for left turn or exit 3, right turn or exit 1, U-turn or exit 4, and going straight or exit 2 are shown in blue, green, red, and yellow, respectively.

1.3 Motion planning Hierarchy in Self Driving vehicles

A commonly adopted approach to planning the motion of an autonomous vehicle is to partition the tasks into a hierarchical structure. The system may have multiple planners and sub-routines running in a sequence or parallel and that may or may not provide feedback to each other, but all of them can be grouped into the following four-component sequential architecture:

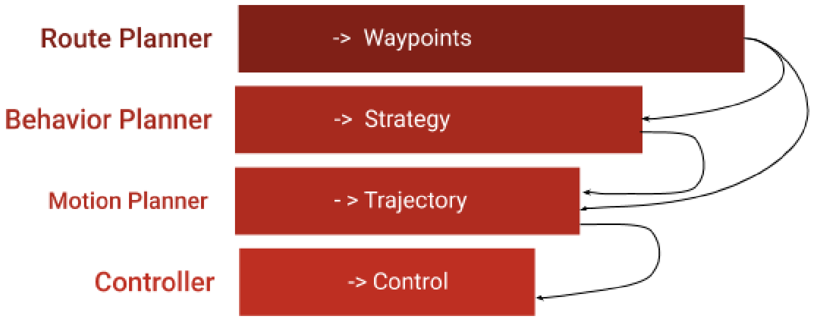

Figure 1.3: Planning hierarchy in self-driving vehicles

1. **Route Planner**:

   This planner is at the highest level of planning and selects the best route from the starting position to the destination given a road network. A standard technique is to find an optimal path given a road network modeled as a connected graph under constraints like following traffic rules. The path is generally specified as a set of waypoints to be followed by the motion planner. [1] provides an example of such a planning technique.

2. **Behavior Module:**

   The behavior module or planner acts as an intermediate layer between the route planner and motion planner. It's mainly responsible for deciding the vehicle's interaction with the other agents, obeying the traffic rules, ensuring safety, and reaching the goals optimally. The behavior can be high-level instructions like turn-left, change-lane or cruise in the lane. For example, if a car on the highway is being slowed down due to the traffic in its lane, then the behavior planner is responsible to first evaluate if a lane-change would help, and determine when to change lanes.

3. **Motion Planner:**

   The motion planner is responsible for generating a continuous trajectory to reach a given waypoint while following the strategy or behavior defined by the behavior planner within the safety and comfort constraints. It's also responsible for collision avoidance with static and dynamic objects or agents in the environment. Approaches based on search-based planning and sample-based planning are commonly used to find a feasible solution. The planner may be composed of multiple sub-routines or involve a separate trajectory optimization to make this problem computationally tractable. The work of Gu et al. [8] provides an insight into motion planning systems.

4. **Trajectory Controller:**

   The controller controls the steering and throttle to follow and track the trajectory generated by the motion planner. It's responsible for following the plans generated by the above layers within certain bounds and providing necessary feedback. Techniques such as Model Predictive Control (MPC) and pure-pursuit are commonly employed for the task of following either a trajectory or a path.

1.4 Outline of this work

The main focus of this work is the real-time decision making or behavior planning of an autonomous vehicle at roundabouts. The algorithm is responsible for selecting the appropriate maneuver, interacting and negotiating with other traffic participants with unknown intentions, and dealing with perception uncertainties similar to a human driver in the complex unsupervised roundabout scenario. Related aspects such as the required information for decision making /planning, data for learning and the algorithm output are also addressed in the following chapters.

The remainder of this thesis is structured as follows:

- *Chapter 2* summarizes existing work on behavior planning or decision making in urban scenarios along with related work on driving at roundabouts. This chapter highlights that, while numerous works have already produced advanced algorithms for simplified urban intersections, not much emphasis has been given to decision making at roundabouts.

  The subsequent sections provide a mathematical foundation for formulating the behavior planning problem as a Markov Decision Process (MDP) or Partially Observable Markov Decision Process (POMDP) along with the justification for using POMDPs.

- *Chapter 3* presents two deep reinforcement learning paradigms for decision-making at roundabouts. It explains the model architecture, training process, the simulation setup, and results for these. The presented approaches are compared with a conventional decision making strategy and tested for generalization across different environments and under imperfect perception.

- *Chapter 4* focuses on methods to generalize policies learned using deep reinforcement learning to different traffic densities. Two unique frameworks are presented that can also be extended to other driving scenarios.

- *Chapter 5* concludes this thesis by summarizing the findings and presenting future directions in the development of planning frameworks for urban driving scenarios.

- *Chapter 6 or Appendix* provides supporting material and discussion for the methods presented in *Chapter 4*.

Chapter 2

Background

2.1 Related work

Katrakazas et al. [11] present a detailed survey to classify recent research on modelling, prediction and behavior planning for intelligent vehicles. Discussion on the works that are related to our work (MDPs, unsignalized intersections, and roundabouts) is presented below. Early work on autonomous cars like that of Urmson et al. [22] used simplified slot-based approaches in the hierarchy of planning for such problems. As the decision of such approaches is based only on the current states, it's difficult to obtain a robust behavior.

The work of Galceran et al. [7] involves the design of a closed-loop forward simulation of all vehicles with assigned policies using dynamics and observation models, and evaluating interactions in unsupervised intersections. The ego vehicle executes a policy from a discrete set of policies which limits possible behaviors.

Dong et al. [6] present a probabilistic graphical model-based approach for merging onto a highway. Their work successfully integrates the history of participants into the planner and doesn't require any reward function. Similarly, Wei et al. [26] present a simplified framework for intention estimation and cooperative merging, but it does not consider the history of participating agents. Both these works only tackle a subset (merging) of the problem of navigating at roundabouts, which also involves exiting and lane changes.

Hubmann et al. [10] present a POMDP framework which is applicable to any number of agents and is an online, anytime algorithm in continuous state space. They evaluated the use of heuristics to speed up computation and present results based on unsupervised intersections. They use probabilistic sampling (particle filter) guarantees for an optimal solution and do not present results for multiple maneuvers over short distances which are seen in roundabouts.

The work of Liu [15] uses road context for situation modeling, involving typical vehicles motion patterns and computes approximate solution of POMDPs using an online solver, DESPOT [19]. Although they present their results on a variety of unsupervised scenarios including a single lane roundabout, a major limitation of their work is the chosen actions, as they only consider acceleration and deceleration of the vehicle on the reference path.

Zyner et al. [29] use a supervised learning method with a recurrent neural-network for predicting the trajectories of vehicles in a roundabout. Learning a robust prediction system is a challenging task and still requires developing a complex planning module to make decisions. Zhao et al. [27] present a method for merging into roundabouts using support vector machines based on a dataset collected on a single lane roundabout. The work of Wang et al. [23] for camera-based decision making at roundabouts poses generalization challenges, as it operates on raw sensor data. Beaucorps [5] uses data collected by vehicles driven by selected humans in a simulation to learn their decision making framework. Their work is validated on the same dataset and may not reflect the true distribution of behaviors observed in roundabouts.

In many of the works listed above either the problem of consecutive decision making in short time, present at roundabouts, is not considered or the proposed approaches have limited action space and tested on a small amount of data without a discussion on their performance across different traffic densities. In this work, we tackle the problem of consecutive decision making with a larger action space involving possible lane changes, and also present methods to generalize the performance across different traffic densities.

2.2 MDPs and POMDPs

Reinforcement learning problems are often formalized as Markov Decision Processes (MDPs), described by a 4-tuple $\langle S, A, T, R \rangle$, and Partially Observable Markov Decision

Processes (POMDPs), described by $\langle S, A, Z, T, O, R \rangle$. Unlike MDPs, which assume that the states ($s_t \in S$) are fully observable, POMDPs assume that the state of a robot or a vehicle is not known. Z is a set of observations, $T(s_t, a_t, s_{t+1}) = p(s_{t+1}|s_t, a_t)$ is the probability of transitioning to state s_{t+1} when the agent takes action a_t in state s_t, $O(s_t, a_t, z_t) = p(z_t|s_t, a_t)$ is the probability of observing z_t if the agent takes action a_t and ends in state s_t. At each timestep t, an agent interacting with the environment using policy π' receives a reward $r_t \sim R(s_t, a_t)$ with the objective of maximizing the expected discounted reward R_t.

$$R_t = r_t \ + \ \gamma r_{t+1} \ + \ \gamma^2 r_{t+2} \ + \ ... \tag{2.1}$$

where $\gamma \in [0, 1]$ is the discount factor.

Watkins et al. [24] proposed Q-learning as a model-free learning approach to computing optimal policies for MDPs. The Q-value is the value of executing an action in a given state followed by an optimal policy π, as defined below:

$$Q^{\pi}(s, a) = E^{\pi}(R_t|s_t = s, a_t = a) \tag{2.2}$$

The Q-values are learned in an online, iterative fashion using:

$$Q(s, a) = Q(s, a) \ + \ \alpha(r \ + \ \gamma max_{a'} Q(s', a') \ - \ Q(s, a)) \tag{2.3}$$

While POMDPs are better suited to modeling tasks with uncertain effects and partial state observability, methods for computing their optimal policies are not very efficient. A few POMDP solvers focus on approximating a finite subset of beliefs that provide reasonable results while reducing computational complexity [13][19]. For a more detailed survey of conventional techniques for solving MDPs and POMDPs readers may refer to [20].

Mnih et al. [16] used a neural network (DQN) as a function approximator to estimate Q-values (parameterized as $Q(s, a, \theta_i)$) for MDPs by minimizing the following loss function:

$$L(\theta_i) = E_{s,a,r,s'}[\,(y_i^{target} \ - \ Q(s, a, \theta_i))^2] \tag{2.4}$$

$$(y_i^{target} = r \ + \ \gamma max_{a'} Q(s', a', \theta_i) \tag{2.5}$$

2.3 Behavior Planning using POMDPs

Modeling the behavior planning problem as a POMDP allows us to incorporate all the perception uncertainties, road context of roundabouts,and unknown intentions of varying number of traffic participants into a single problem. This also enables us to integrate planning and prediction into a single problem, as the agent learns to reason about its future. In this work, the word ego vehicle is used interchangeably with agent.

An important factor in many works related to deep reinforcement learning for complex MDPs is the assumption of full state observability that allows neural networks to use only a few samples to find optimal policies. Recently, Hausknecht [9] and Zhu et al. [28] presented slight modifications to DQN that alleviate this assumption and allow for efficient computation for POMDPs. These were only evaluated on flickering Atari games where the agent sometimes does not get any state estimates and therefore, their results represent only a subset of POMDP tasks. Chapter 3 extends these deep-reinforcement learning formulations of POMDPs to the task of navigating a roundabout.

Chapter 3

Planning using Deep Reinforcement Learning

3.1 Reinforcement learning framework

In the sections below, we describe the framework that allows us to plan efficiently using deep reinforcement learning techniques based on POMDPs, use of a state encoding that enables generalization for different roundabout geometries and a training procedure that enables intuitive and efficient training.

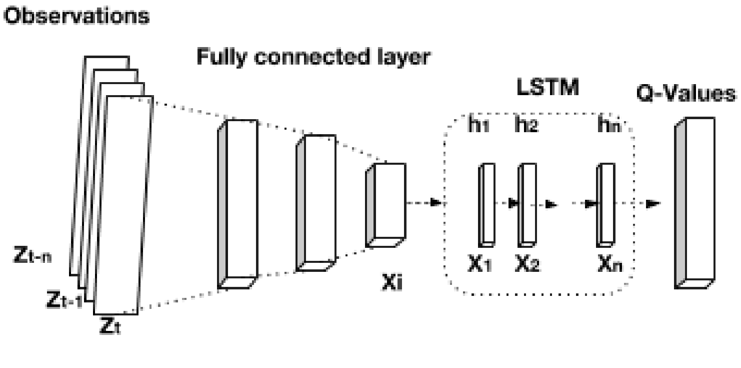

Figure 3.1: DRQN network architecture for model-free POMDP learning which only considers observation history of the agent

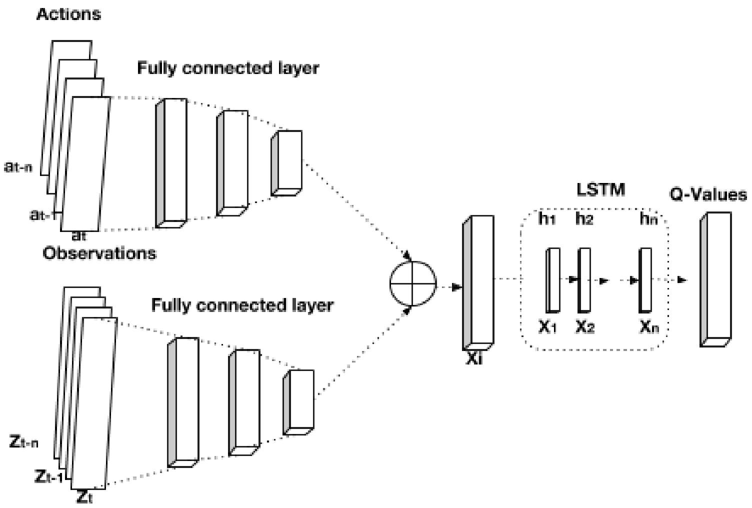

Figure 3.2: ADRQN network architecture for model-free POMDP learning. Unlike DRQN, ADRQN also considers the action history of the agent.

3.1.1 Network architecture

We use two different network architectures for model-free POMDP learning and compare them to DQN [16], which is based on model-free MDP learning. Our implementation of DQN consists of 4 fully connected layers. To tackle the problem of learning from partially observable states we use an implementation similar to that of [9] in which an LSTM layer was added after DQN layers, which enables it to integrate an arbitrarily long history. We refer to this architecture as DRQN as per the original work. Thus, the presented architecture of DRQN (Fig. 3.1) estimates the function $Q(z_t, h_{t-1}|\theta)$ instead of $Q(s_t, a_t|\theta)$ (DQN). $h_t = LSTM(h_{t-1}, z_t)$ denotes the output of the LSTM layer, h_{t-1} is the output at the previous step, and θ are the parameters of the estimating function.

Given that the past actions influence past observations, we try to explicitly incorporate the full influence of the agent's history by using a network similar to [28] (Fig. 3.2). This is referred to as ADRQN here. In this architecture we use observation and action pairs and pass them through fully connected layers before concatenating their features, which are then passed trough a network similar to DRQN. For this we also modify our replay buffer to store the updated transition tuples, $(\{a_{t-n}, z_t\}, a_t, r_t, z_{t+1})$.

We experimented with the number of layers for LSTM, number of past observations, actions considered and loss functions (Huber, MSE). We present results for our best hyper-parameters, which are: 2 LSTM layers, 15 past observations and Huber loss.

3.1.2 State and observation space

We assume that the ego-vehicle has a perception system that detects and tracks other participants in its sensor field. Fig. 3.3 depicts the sensor field for which the vehicle tracks the position and velocity of a maximum of four cars, one vehicle ahead of and behind the ego vehicle in the same lane, and one ahead of and behind it in the merging lane. The parameters of sensor field used are given in Table 3.1. We also assume that the vehicle has an estimate of its current position, velocity, lanes surrounding it, and distance to the next junction.

Using the above information, the state and observation space of the ego vehicle is defined as follows:

$$S_{ego} = \langle x, y, v_x, v_y, d_j, lane_l, lane_r \rangle \tag{3.1}$$

where x, y, v_x, v_y are the position and velocity vector components along the x, y axis of the vehicle and d_j is the distance to the next junction. The coordinate transformation that aligns the vehicle position and velocity along the vehicle axis makes the vehicle state invariant to road geometry. $lane_l, lane_r$ is a binary value that is 1 if a left or right lane exists with respect to its current lane.

The state or observations of other participants is recorded as:

$$O_{car1} = \langle cp, x, y, v_x, v_y, d_j \rangle \tag{3.2}$$

where O_{car1} refers to the observed estimates of one car, cp is a binary variable that is 0 if no car is present. Similarly, for other cars that may be present in the ego vehicle's perception field $O_{car2}, O_{car3}, O_{car4}$ are computed. If no vehicle is present in one of these four slots, then the x, y coordinates are set to sensor range limits, and velocities and cp are set to 0.

Using the above, the complete state of the agent at each time step can be defined as follows:

$$s_t = \langle S_{ego}, O_{car1}, O_{car2}, O_{car3}, O_{car4} \rangle \tag{3.3}$$

As per terminology, we refer to s_t as the *state* of the agent in the case of DQN (MDP) and as the *observation* of the agent, z_t, for the POMDP formulation.

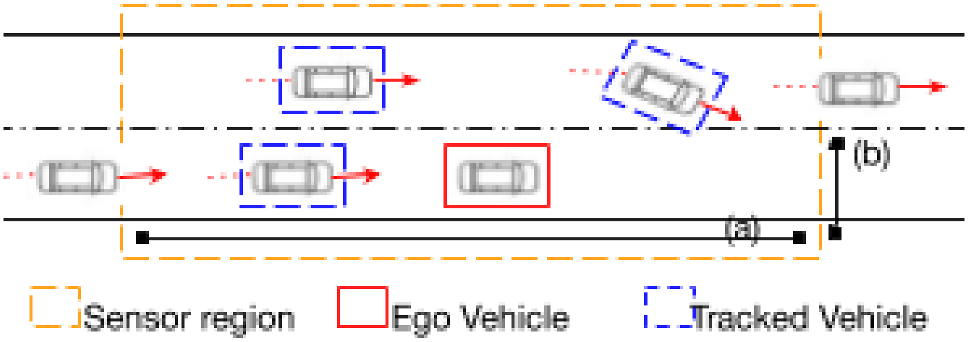

Figure 3.3: Top-view of the sensor range and vehicle tracking setup used.

3.1.3 Reward

The reward function acts as the objective for the optimization problem. For our scenario, the goal is to safely and efficiently navigate a roundabout without impeding the traffic flow. A naive approach of defining a sparse reward based or on time for successful termination would not only increase the time required for convergence, but also encourage aggressive behavior, as the agent will try to go as fast as possible. To account for traffic rules we also include a negative reward for exceeding the speed limit, lane change, and also for getting too close to a vehicle in front. The reward function is also modified to include a high negative reward for colliding with another vehicle which terminates the episode. Changing into an invalid lane is also considered a collision.

3.2 Driving stages at Roundabouts

A typical vehicle behavior in a roundabout consists of the following: merging, traversal and exit. We divide the problem into these distinct stages and learn a separate model for each of these. We define a discrete action space based on the intuition on how humans maneuver in traffic. We use a discrete action space, as behavioral planning happens on a slower time scale than motion planning or trajectory control. We also assume that the path computation and trajectory control are handled by an existing module.

3.2.1 Merging

This stage involves approaching the roundabout, making the vehicle merge into the roundabout, and yielding to the traffic already in the roundabout. As the only decision to be made at this stage is when to merge into the roundabout, we use a simplified action space:

$$a_t = \{go,\ no\ go\} \tag{3.4}$$

where, $a_t = go$ signifies that a vehicle at the junction follows the trajectory defined by the motion planner. This simplification helps in learning a more robust planner for this complex behavior.

3.2.2 Traversal

This stage consists of driving on the circular roadway inside the roundabout and may involve lane changes in case of multi-lane roundabouts. An action space similar to the merging stage will result in sub-optimal performance of the agent; therefore, the following action space is used for this stage:

$$a_t = \{acc, decelerate, change_{left}, change_{right}\ , cv\} \tag{3.5}$$

where the vehicle accelerates,decelerates at fixed rates, changes lane or continues at velocity specified by cruise control cv on its defined path. To accelerate at different rates and to obtain a more detailed behavior one can also define discrete actions with multiple predefined accelerations.

3.2.3 Exit

This stage involves making a right turn that leads the vehicle out of the roundabout. This may involve yielding to other traffic participants like pedestrians, bicyclists, etc. We use the same action space as the one defined for the traversal stage, as the vehicle may be required to make a lane change before exiting.

$$a_t = \{acc, decelerate, change_{left}, change_{right}\ , cv\} \tag{3.6}$$

3.3 Simulation setup

For simulating the roundabout scenario the Simulation of Urban Mobility (SUMO) traffic simulator [2] was used along with TraCI [25] for interfacing. Fig. 3.4 depicts a bird's-eye view of the double-lane roundabout scenario with four exits. SUMO is not suited for rendering large-curvature roads, as it defines them as multiple linear segments; therefore, we chose to ignore the curvature of the road. Also, the curvature of the road is handled by the motion and trajectory planning modules and our task is to obtain robust consecutive behaviors.

The roundabouts are connected to long approach roads with give-way lines at the junctions. Each road within and approaching the roundabout has a suggested speed limit that can be exceeded by the ego vehicle as well as other vehicles. The simulator is setup to allow collisions. The simulation parameters for the selected roundabout and the vehicle properties are presented in Table 3.1. These values are consistent with the recommendation of US DoT for rural and urban double-lane roundabouts [18]. The ego vehicle follows the policies computed using our algorithm, whereas other participants use the default car following model (Kraus) of SUMO.

Table 3.1: Simulation scenario and Vehicle parameters

Inner island radius	22.5 m	Vehicle Length	4.5m
Lane Width	3.5 m	Vehicle Width	1.6m
Velocity Limit	11.2 m/s(25mph)	Max Acceleration/Deceleration	2 m/s^2
Sensor range parameter 'a'	150m	Sensor range parameter 'b'	50m

3.4 Training process

Similar to their work of Lin [14] and that of [16] we also store previous samples in a replay buffer set up to a fixed size and train on uniformly sampled mini-batches from

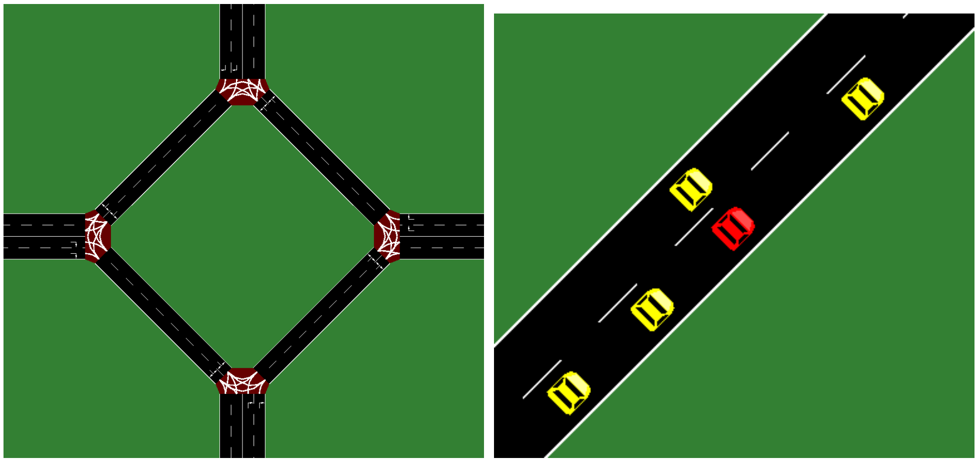

Figure 3.4: A simulation setup of a roundabout generated using SUMO [2]. (Left) depicts the four-exit roundabout with double lanes with the intersection network. (Right) The ego-vehicle is highlighted in red and the yellow vehicles represent other vehicles in the roundabout.

this replay buffer. Additionally, we also create a separate, target network $Q(s, a, \theta')$ that provides updates to the main network $Q(s, a, \theta)$ for learning stability. Both the networks are identical except that the target network is updated at each iteration and its weights are copied to the main network at every 1,000 iterations or time-steps. Finally, RMSProp, an adaptive learning rate method, is used.

During training, we use a curriculum learning approach, wherein each model is first trained without any other interacting vehicle so that it learns the most optimal policy and later with other vehicles with random initialization. In later stages, an additional bonus reward is given to merging and traversal if they lead to successful exit to enable long-term consistent behaviors. The rewards were tuned using initial tests such that probability of crash is highly minimized and an expected human-like behavior is obtained.

3.5 Experimental Results

We evaluate the models on the training simulation setup and others with variations in observability and roundabout geometry. Since we consider the models to be trained without the need for further exploration we set $\epsilon = 0$ for the ϵ-greedy approach. We replay each scenario based on the learned model 100 times to obtain an average of the evaluation metrics that are presented in the sections below. We present results for the complete scenario - merging, traversal,and exit.

Unlike other reinforcement learning tasks, using cumulative rewards as an evaluation metric is of little significance for autonomous-driving cases. An agent exhibiting fast and aggressive behavior may obtain higher rewards than others. Therefore, we use other metrics like time to traverse the roundabout, collision rate and distance to other vehicles. Similar to [15], we use two different metrics for minimum distance, small gap (less than 5m) and large gap (5-7m), that are computed for each replay and averaged. The values of of small or large gaps are presented as the ratio of time the ego vehicle is within the specified range from the leading vehicle to the total travel time. The collision rate are reported as a percentage between [0,1] across the number of trials. During our work we also assume that other participants have a 50% chance of deviating from their driver model, thus representing a difficult scenario. As there are no real world data available to calibrate the uncertainty in driver models a high value of 50 % is used to reflect a worst case scenario. A discussion on driver models is presented in the Appendix (6.1) at the end of this document.

3.5.1 Standard test results

We test the behavior on the same scenario in which the vehicle was trained. The number of participating vehicles and their routes are fixed for a given experiment but their initial positions is varied randomly. Table 3.2 presents results for one interacting vehicle which is within the roundabout and Table 3.3 summarizes the results for 5 interacting vehicles with ego vehicle going straight (exit 2). We also tested a simple rule-based planner to evaluate the need for complex behavioral planners. It uses Time to Collision metrics for merging and then an adaptive cruise control maintains the speed in the given lane.

Table 3.2 demonstrates that simple planners and our proposed methods have similar performance in the case of a single interacting vehicle. However, our models significantly outperform the rule-based planner with a lower collision rate and decreased travel time in the case of multiple vehicles, as highlighted in Table 3.3. Particularly, we also observe that the two POMDP-based architectures have less collisions and

Table 3.2: Single Interacting Vehicle

Algorithm	Time to traverse (s)	Small Gap	Large Gap	Collision Rate
Rule-based	27.1	0.17	0.56	**0**
DQN	27.1	0.10	0.64	0.01
DRQN	27.3	0.26	0.47	**0**
ADRQN	27.3	0.19	0.53	**0**

Table 3.3: Multiple Interacting Vehicles

Algorithm	Time to traverse(s)	Small Gap	Large Gap	Collision Rate
Rule-based	33.1	0.62	0.26	0.31
DQN	31.8	0.74	0.16	0.26
DRQN	30.7	0.65	0.31	**0.22**
ADRQN	30.9	0.81	0.09	0.23

lower travel time than DQN, which formulates this problem as a fully observable MDP. Both POMDP formulations have similar scores so it can't be determined if encoding actions explicitly has a significant effect on collision rate and travel time. This could be due to the fact for a given sequence of observations there exists a deterministic policy, and a network could infer that for a given problem. ADRQN also has a higher ratio of small gap to large gap as compared to DRQN and others which illustrates that the system prefers to follow a leading vehicle more closely than others which might result in the slightly higher collision rate observed.

3.5.2 Generalization results

Imperfect state estimation

After being trained with perfect observations, where the agent receives all estimates of the vehicles in its perception field, we test the learned policies in settings with observation probability of 0.8; i.e. at each time step there is a 0.2 probability that a car present in the agent's perception field is dropped, setting $cp = 0$. The results are shown in Table 3.4. We observe that POMDP (ADRQN) formulations perform better even with imperfect state estimation that may arise due to errors in perception pipelines. The ability to account for missing information provides a large advantage for the behavioral planners that do not use a separate prediction module.

Table 3.4: Imperfect State estimation (going straight)

	Perfect estimations	Imperfect estimation
Algorithm	Collision Rate	Collision Rate
Rule-based	0.31	0.36
DQN	0.26	0.34
ADRQN	0.23	**0.27**

Different roundabout geometries

Roundabouts vary a lot from one place to another, so it is crucial for the learning based planners to account for these variations. The presented formulation encodes the states and observations that aren't dependent on these variations. To test the generalization ability we test the learned planners for different exits for a different geometry with a single lane and 3 exits.

Table 3.5: Different roundabout geometry (going straight)

	Double-lane, 4 exit (baseline)	Double-lane, 3 exit	Single-lane, 3 exit
Algorithm	Collision Rate	Collision Rate	Collision Rate
Rule-based	0.31	0.31	**0.19**
DQN	0.26	0.25	0.34
ADRQN	0.23	0.18	0.29

As shown in Table 3.5, the performance achieved by the POMDP-based technique does not vary drastically as the number of roundabout exits changes. Table 3.5 also illustrates that the rule-based planner outperforms others in the case of a single-lane roundabout as there's no scope for lane change. Whereas, the policies learnt on double-lane roundabout do not transfer directly to a single-lane scenario.

Chapter 4

Generalizing behaviors across different traffic densities

4.1 Overview

The previous work, similar to traditional reinforcement-learning approaches (especially those based on MDPs) assumes that the environment model is always fixed. This is not a realistic assumption in many real-world applications such as the case of self-driving vehicles. For instance, the traffic scenario in major urban centers can vary significantly depending on the time of the day, and effective algorithms need to adapt to those changes.

This problem of 'non-stationary' environments is difficult to solve if sufficient data or environment dynamics are not known. Our approach of model free-learning using POMDP formulation alleviates some aspects of the non-stationary environments by assuming it as a part of partially observable states. In early 2000, research showed that better performance can be achieved using hidden-state formulations for unknown environments over POMDP formulations if there exists some regularity in the way environment dynamics change. This chapter extends that work for driving behaviors across different traffic densities. In section 2 we introduce a new method to tackle this problem.

4.2 Hidden modes for traffic density

In this section, we formulate the problem of generalization across different traffic scenarios using the formulation first presented by Choi et al. [4] using a formal non-stationary environment model (Fig. 4.1) with repeated dynamics. Their proposed model decomposes the non-stationary environment into multiple stationary environments called modes. Each mode is an MDP with distinct dynamics, requiring different policies, and an agent can only be a part of one mode at any time. A key feature of their formulation was that transition between different modes is not dependent on the agent's action and the mode dynamics are slower than the agent dynamics. The following properties of our scenario make it suitable to use the proposed model:

Their proposed model had the following properties as described in their original work:

1. **A Finite Number of Environment Modes**
2. **Small Number of Modes**: the number of modes is much fewer than the number of states.
3. **Infrequent Mode Transitions**: a mode is likely to persist for some time before switching to another one.
4. **Partially Observable Modes**
5. **Modes Evolving as a Markov Process**: mode transitions are stochastic events and are independent of the agent's response.

4.2.1 Learning policies for hidden modes

Based on the first three properties, a simple framework of generalization using multiple modes is proposed where different behavior planning policies are independently learned for different predefined traffic densities or modes. The approaches presented in the previous sections are used to learn policies for different densities at a given roundabout. As the hidden mode approach does not impose any constraints on the type of stationary environment described by each of the modes, which can be an MDP or POMDP, this can be extended to scenarios other than roundabouts.

Unlike the work of Choi et al. [4], we do not make an assumption about the

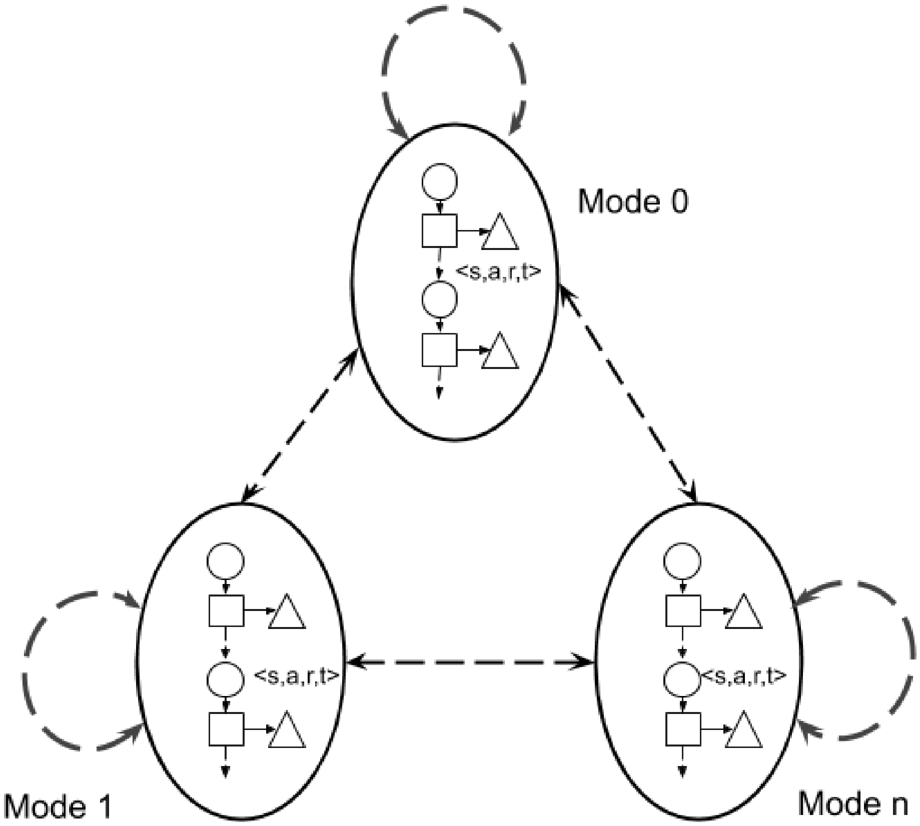

Figure 4.1: A Hidden-Mode representation with n modes. Each mode is a MDP or POMDP

transition dynamics from one mode to another. As the mode is not directly observable, the current mode is estimated by another module referred to as the *change-point detector*. The following section describes two methods for change-point detection.

4.2.2 Change-point detection

A key component of the hidden-mode approach is to identify the mode that the agent is in at time t. While Choi et al. formulated this as a model-based MDP learning problem, the following section proposes two methods that do not make this assumption. Here it is assumed that the number of modes is known in advance.

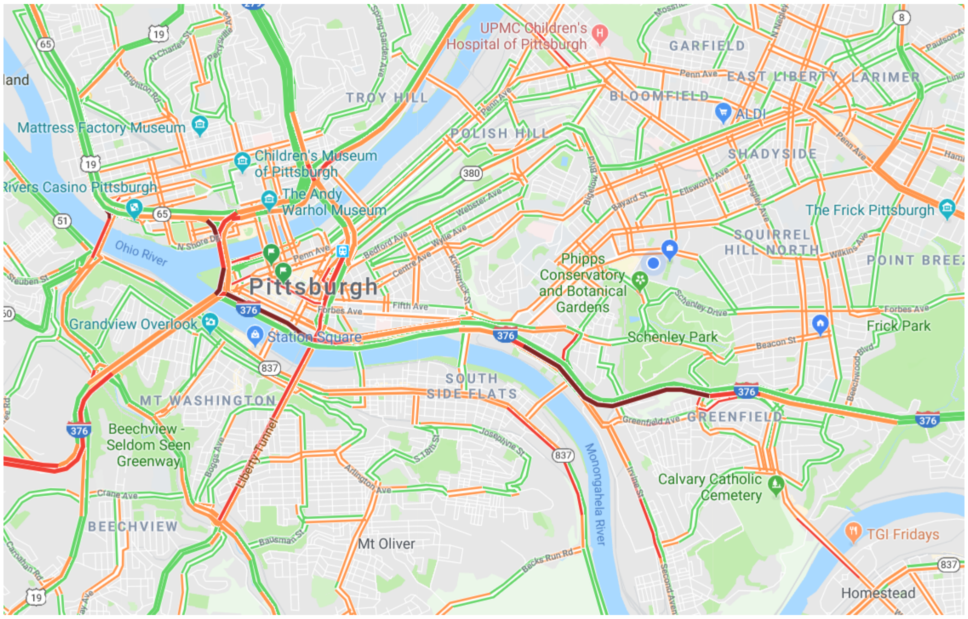

Figure 4.2: A bird's-eye view of expected traffic zones around Pittsburgh, PA during morning hours. Yellow, orange, red, deep red represent the traffic severity in increasing order.

Using external information (HM-e)

There are many readily available sources other than the state or observation space of the agent/self-driving car that help characterize the traffic densities on-road. For example, Google Maps (Fig. 4.2) are widely used to inform drivers about existing traffic conditions and to plan routes. They divide the road traffic into small finite sets based on historical travel time data sourced from a large number of vehicles. By leveraging such external systems additional computation can be avoided and more consistent characterization of traffic scenario can be obtained as they can provide information beyond the perception range of the car. A *hidden-mode* framework, *HM-e*, that uses these external sources is developed where an external agent similar to Google Maps is implemented. This external system divides the driving scenario into n modes and the policy corresponding to that mode is executed. The external system does not vary the mode on a given edge of the network, thus satisfying the criterion of non-stationary environment dynamics changing at a slower rate than the agent dynamics.

Supervised learning (HM-s)

There may be cases in which an agent does not have access to external sources for detecting change-points. In such scenarios, the agent has to estimate the mode from its perception system or by using its observation and action histories. A simple classification network can be used that estimates the mode based on the observation z and action a from time $t-k$ to t. For this supervised learning framework, a network is first trained using the learned controllers with perfect knowledge. The learned change-point estimator is then used to estimate mode and deploy the corresponding model. This framework is referred to as *HM-s*.

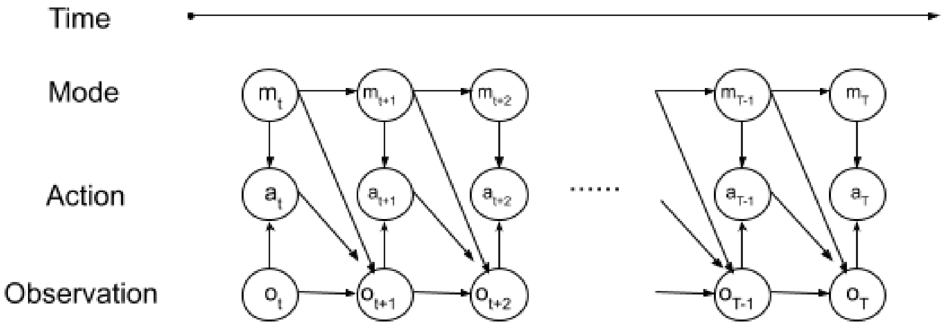

Figure 4.3: A graphical representation of a model-based MDP formulation where arcs describe the dependencies between nodes and each node represents either a mode, state or action.

4.3 Traffic Conditioned Models for behavior planning

Using multiple models for hidden-modes is a simplifying abstraction of the real world and it has its limitations. The methods presented in this section relax the assumption of non-stationary environment modes being discrete and proposes a new framework that uses a single model for generalization.

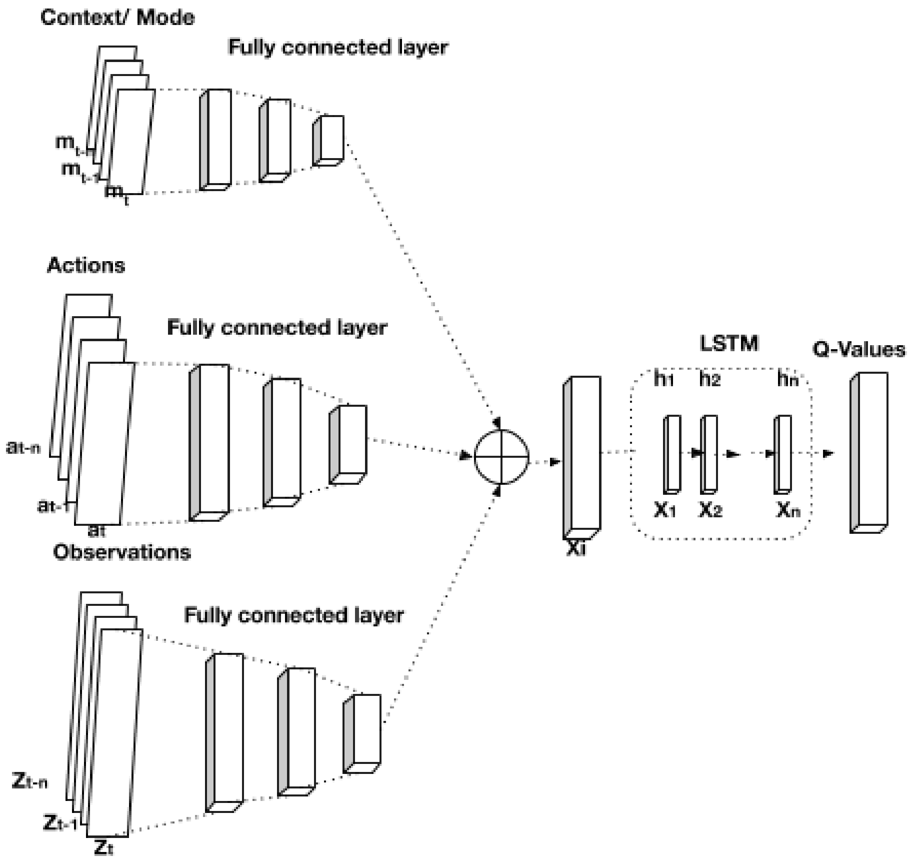

Figure 4.4: Traffic-Conditioned model framework for generalization across different traffic scenarios. Unlike a hidden-mode framework, this uses a single model where the non-stationary environment mode is encoded as a mode vector. The mode vector can be either a discrete or continuous variable.

4.3.1 Discrete modes (TCM-d)

The work of Chen et al. [3] uses the explicit encoding of kinematic features of an agent to compute policies that generalize better when agent parameters are varied. Using a similar approach, a *Traffic-Conditioned* model is proposed that uses a single model for all modes of the non-stationary environment by encoding the modes along with the state or observation space. In this formulation, the agent policies are dependent on its observation z_t, past action a_{t-1}, and mode of the non-stationary environment. For discrete modes, a change-point detection method using an external source is used and referred to as *TCM-d*. Fig. 4.4 presents a graphical representation of this approach.

4.3.2 Continuous modes (TCM-c)

One may argue that the driving environment is continuous and representing it as discrete modes for tractability restricts the representation power significantly. The hidden-mode approach is severely impacted by increasing the number of modes, as the number of different models to be learned increases linearly as the number of modes is increased. Also, as the number of modes increases, the hidden-mode method suffers from oscillations at the boundaries of the mode transitions. While the previous approach could not leverage a continuous mode due to the reasons presented above, *Traffic-Conditioned Models* can exploit a continuous non-stationary environment using appropriate state augmentation.

For this, a major challenge is defining a continuously varying feature that is representative of the non-stationary environment, unlike symbols that are assigned to different discrete modes. Taking inspiration from the way the above modes were defined, it is proposed that *ratio of the average velocity of vehicles in perception range for a given time k to target speed* is a useful mapping of the non-stationary environment dynamics. By using this mapping one can also avoid using an explicit change point detection module.

Similar to the *Traffic-Conditioned Models- Discrete* approach, a single model is used for generalizing traffic behaviors by using χ, defined below, as the mode. The presented approach is referred to as *TCM-c.*

$$m_t = \chi_t = \frac{avg.\ velocity_{t-k\ldots t}}{target\ speed_t}$$

4.4 Experimental Results

We use the same setup as the one presented in Chapter 3 for this section as well. The action space is partitioned into different driving stages, with the same state and reward function for an accurate comparison. We only test for going straight with 15 interacting vehicles around the ego vehicle.

Different traffic densities are created by regulating the speed of other participants with random spacing so the ego vehicle can merge. For training, two different traffic densities are considered. We regulate the speeds of the vehicles to 5mph for *Density 1*, thereby slowing the traffic considerably from the target speed of 25mph. For *Density 2*, the forward vehicles also move at the target speed of 25mph, thereby causing no bottleneck.

For evaluation, we consider an additional traffic density that is in the middle of the two that were considered for training to investigate the drawbacks of discrete modes. For this *eval Density 3*, the forward vehicles move at 15mph.

To learn policies for different modes of HM-s and HM-e we use ADRQN, whereas for TC-d and TC-c we use the network architecture presented above, which is similar to ADRQN. We also compare these frameworks against an ADRQN policy learned on one traffic density.

Table 4.1: Generalization performance across different traffic densities

	Density 1	Density 1	Density 2	Density 2	eval Density 3	eval Density 3
Algorithm	Time to traverse (s)	Collision Rate	Time to traverse (s)	Collision Rate	Time to traverse (s)	Collision Rate
ADRQN	30.9	**0.23**	84.6	0.57	58.1	0.79
HM-e	31.1	**0.23**	89.3	**0.05**	54.8	0.29
HM-s	33.6	0.30	86.8	0.21	55.9	0.58
TC-d	30.9	**0.23**	91.7	0.06	55.1	0.31
TC-c	32.6	0.24	90.1	0.06	59.6	**0.17**

Table 4.1 demonstrates the advantages of a framework that accounts for the non-stationary environment as compared to a single policy learned on a specific traffic density. We observe that HM-e, TC-d, and TC-c have a comparable performance on traffic densities that they were trained on and exceed that of ADRQN. However, for the *eval Density 3*, which is at the midpoint of the other two densities, methods based on discretized modes and use of external information have significantly higher collision rate than the TC-c, which does not make these assumptions. While TC-c takes a bit more time than others, it has the lowest collision rate for the unknown case. Higher collision rates for HM-s could be due to violation of the independent data assumption that is made during supervised learning of the change-point detector.

Chapter 5

Conclusions

5.1 Summary and discussion

In this work, we investigate behavioral planners that are capable of navigating an unsignalized roundabout safely and efficiently. Handling of the combined problem of merging, traversal, and exiting in a single framework demonstrates the system's ability of long-term reasoning where one wrong maneuver can affect its future. Our choice of state-space encoding, use of different POMDP formulation also allows them to be robust to perception uncertainty and geometry variations. We use a deep-learning framework with LSTMs in Chapter 3 that enables more efficient computation than approximate POMDP solvers. We also observe that the process is similar to conventional DQN, but we obtain better performance by accounting for state-history through recurrent elements. Encoding additional information about the agent's past actions improves the performance marginally.

Finally, we formulate the problem of generalizing across different traffic densities as a hidden-mode reinforcement learning problem that can be seen as a subset of a POMDP formulation. The presented framework of learning multiple models for different modes of a non-stationary environment and use of change-point detection modules is intuitive and leads to better results. In addition to this, our proposed Traffic-Conditioned framework results in decreased computational complexity by using a single policy and can accommodate a continuously varying non-stationary environment avoiding the need of handcrafted discretization. While the use of discrete modes with external information for change-point detection results in more optimal policies, they require carefully handcrafted division of the environment into discrete modes and can result in lower performance at the boundaries of modes.

We have demonstrated the effectiveness of our approaches, in several cases and in comparison to other methods for robust behavior planning at roundabouts.

5.2 Future work

Our current approaches are applicable to scenarios other than roundabouts but suffer from the same drawbacks as other deep reinforcement learning formulations, which is the need for handcrafted reward functions. This work can be improved by methods that learn the rewards using real-world data or methods that can learn effectively from sparse reward functions.

For generalization, the hidden-mode formulation can also be viewed as a hierarchical learning problem where one MDP/POMDP framework selects the mode while the other learns the driving behavior given the mode.

Chapter 6

Appendix

6.1 Driver models

The car-following driver model in SUMO is based on the work by Krau [12]. This discrete-time, continuous space model considers the braking distance as a safety criterion and regulates the speed. The final speed of the car is a minimum of:

1. the maximum speed that vehicle can drive on the road
2. speed based on the maximum acceleration of the vehicle
3. a safe speed that allows braking within a given distance from the car ahead

The implementation further assumes that the driver is not perfect in holding its desired speed as suggested by Ranjitkar et al. [17]. This imperfection is modeled as a stochastic acceleration/deceleration and is uniformly distributed around the vehicle's maximum acceleration.

The model parameters like *minGap* and τ, time headway, are set to the minimum value so that collisions can take place. The driver imperfection distribution which is parameterized by percentage standard deviation of expected value, σ, can be estimated from real-world data for a given scenario, but the lack of such datasets for roundabouts is a limiting factor. The results presented in the preceding chapters are based on the large value of σ, thereby representing a more stochastic scenario. The results below present the variation in performance as the driver-imperfection is changed.

Table 6.1: Effect of changing driver imperfection for multiple vehicle interaction scenario

Imperfection	$\sigma = 0.1$		$\sigma = 0.3$		$\sigma = 0.5$	
Algorithm	Time to traverse(s)	Collision Rate	Time to traverse(s)	Collision Rate	Time to traverse(s)	Collision Rate
Rule-based	32.1	0.09	33.4	0.12	33.1	0.31
DRQN	30.2	0.06	29.6	0.11	30.7	**0.22**
ADRQN	31.3	**0.04**	31.9	**0.08**	30.9	0.23

Table 6.1 demonstrates that the proposed framework for DRQN and ADRQN have better performance than rule-based planners across different driver-imperfection values. Even at very low variance in driver models ($\sigma = 0.1$) the rule-based planner not only takes longer to navigate but has more than double the collisions over simple RL-based planners.

Bibliography

[1] Hannah Bast, Daniel Delling, Andrew Goldberg, Matthias Müller-Hannemann, Thomas Pajor, Peter Sanders, Dorothea Wagner, and Renato F Werneck. Route planning in transportation networks. In *Algorithm engineering*, pages 19–80. Springer, 2016. 1

[2] Michael Behrisch, Laura Bieker, Jakob Erdmann, and Daniel Krajzewicz. Sumo–simulation of urban mobility: an overview. In *Proceedings of SIMUL 2011, The Third International Conference on Advances in System Simulation*. ThinkMind, 2011. (document), 3.3, 3.4

[3] Tao Chen, Adithyavairavan Murali, and Abhinav Gupta. Hardware conditioned policies for multi-robot transfer learning. In *Advances in Neural Information Processing Systems*, pages 9333–9344, 2018. 4.3.1

[4] Samuel PM Choi, Dit-Yan Yeung, and Nevin L Zhang. Hidden-mode markov decision processes for nonstationary sequential decision making. In *Sequence Learning*, pages 264–287. Springer, 2000. 4.2, 4.2.1

[5] Pierre De Beaucorps, Thomas Streubel, Anne Verroust-Blondet, Fawzi Nashashibi, Benazouz Bradai, and Paulo Resende. Decision-making for automated vehicles at intersections adapting human-like behavior. In *2017 IEEE Intelligent Vehicles Symposium (IV)*, pages 212–217. IEEE, 2017. 2.1

[6] Chiyu Dong, John M Dolan, and Bakhtiar Litkouhi. Intention estimation for ramp merging control in autonomous driving. In *2017 IEEE Intelligent Vehicles Symposium (IV)*, pages 1584–1589. IEEE, 2017. 2.1

[7] Enric Galceran, Alexander G Cunningham, Ryan M Eustice, and Edwin Olson. Multipolicy decision-making for autonomous driving via changepoint-based behavior prediction. In *Robotics: Science and Systems*, volume 1, 2015. 2.1

[8] Tianyu Gu and John M Dolan. On-road motion planning for autonomous vehicles. In *International Conference on Intelligent Robotics and Applications*, pages 588–597. Springer, 2012. 3

[9] Matthew Hausknecht and Peter Stone. Deep recurrent q-learning for partially observable mdps, 2015. 2.3, 3.1.1

[10] Constantin Hubmann, Marvin Becker, Daniel Althoff, David Lenz, and Christoph Stiller. Decision making for autonomous driving considering interaction and uncertain prediction of surrounding vehicles. In *2017 IEEE Intelligent Vehicles Symposium (IV)*, pages 1671–1678. IEEE, 2017. 2.1

[11] Christos Katrakazas, Mohammed Quddus, Wen-Hua Chen, and Lipika Deka. Real-time motion planning methods for autonomous on-road driving: State-of-the-art and future research directions. *Transportation Research Part C: Emerging Technologies*, 60:416–442, 2015. 2.1

[12] S Krau. Microscopic modeling of traffic flow: Investigation of collision free vehicle dynamics, 1997. 6.1

[13] Xin Li, William KW Cheung, Jiming Liu, and Zhili Wu. A novel orthogonal nmf-based belief compression for pomdps. In *Proceedings of the 24th international conference on Machine learning*, pages 537–544. ACM, 2007. 2.2

[14] Long-Ji Lin. Reinforcement learning for robots using neural networks. Technical report, Carnegie-Mellon Univ Pittsburgh PA School of Computer Science, 1993. 3.4

[15] Wei Liu, Seong-Woo Kim, Scott Pendleton, and Marcelo H Ang. Situation-aware decision making for autonomous driving on urban road using online pomdp. In *2015 IEEE Intelligent Vehicles Symposium (IV)*, pages 1126–1133. IEEE, 2015. 2.1, 3.5

[16] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013. 2.2, 3.1.1, 3.4

[17] Prakash Ranjitkar, Takashi Nakatsuji, and Akira Kawamua. Car-following models: an experiment based benchmarking. *Journal of the Eastern Asia Society for Transportation Studies*, 6:1582–1596, 2005. 6.1

[18] BW Robinson, L Rodegerdts, Wade Scarborough, W Kittelson, R Troutbeck, Werner Brilon, Lothar Bondzio, Ken Courage, Michael Kyte, John Mason, et al. Roundabouts: An informational guide. federal highway administration. *Turner-Fairbank Highway Research Center*, 2000. 3.3

[19] Adhiraj Somani, Nan Ye, David Hsu, and Wee Sun Lee. Despot: Online pomdp planning with regularization. In *Advances in neural information processing systems*, pages 1772–1780, 2013. 2.1, 2.2

[20] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018. 2.2

[21] Eric A. Taub. As Americans figure out the roundabout, it spreads across the U.S. *New York Times*, 2015. (document), 1.2, 1.1

[22] Chris Urmson, Joshua Anhalt, Drew Bagnell, Christopher Baker, Robert Bittner, MN Clark, John Dolan, Dave Duggins, Tugrul Galatali, Chris Geyer, et al. Autonomous driving in urban environments: Boss and the urban challenge. *Journal of Field Robotics*, 25(8):425–466, 2008. 2.1

[23] Weichao Wang, Qinggang Meng, and Paul Wai Hing Chung. Camera based decision making at roundabouts for autonomous vehicles. In *2018 15th International Conference on Control, Automation, Robotics and Vision (ICARCV)*, pages 1460–1465. IEEE, 2018. 2.1

[24] Christopher JCH Watkins and Peter Dayan. Q-learning. *Machine learning*, 8 (3-4):279–292, 1992. 2.2

[25] Axel Wegener, Michał Piórkowski, Maxim Raya, Horst Hellbrück, Stefan Fischer, and Jean-Pierre Hubaux. Traci: an interface for coupling road traffic and network simulators. In *Proceedings of the 11th communications and networking simulation symposium*, pages 155–163. ACM, 2008. 3.3

[26] Junqing Wei, John M Dolan, and Bakhtiar Litkouhi. Autonomous vehicle social behavior for highway entrance ramp management. In *2013 IEEE Intelligent Vehicles Symposium (IV)*, pages 201–207. IEEE, 2013. 2.1

[27] Min Zhao, David Kathner, Meike Jipp, D Soffker, and Karsten Lemmer. Modeling driver behavior at roundabouts: Results from a field study. In *2017 IEEE Intelligent Vehicles Symposium (IV)*, pages 908–913. IEEE, 2017. 2.1

[28] Pengfei Zhu, Xin Li, Pascal Poupart, and Guanghui Miao. On improving deep reinforcement learning for pomdps. *arXiv preprint arXiv:1804.06309*, 2018. 2.3, 3.1.1

[29] Alex Zyner, Stewart Worrall, and Eduardo Nebot. A recurrent neural network solution for predicting driver intention at unsignalized intersections. *IEEE Robotics and Automation Letters*, 3(3):1759–1764, 2018. 2.1