Adapting Vehicular Planning and Communications for Optimized Driving

Student Name : Mayank Kumar Pal
July, 2020

A THESIS

SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR

THE DEGREE OF

M.Tech in Electronics and Communication Engineering
in General Category

Electronics and Communication Engineering

INDRAPRASTHA INSTITUTE OF INFORMATION TECHNOLOGY
DELHI
NEW DELHI– 110020

Thesis Committee

Dr. Sanjit K. Kaul (Chair)
Dr. Saket Anand (Co-Chair)
Dr. Gourab Ghatak
Dr. Sayan Basu Roy

Acknowledgement

I dedicate this thesis work to my parents and sister, who are my pillars of strength. I like to express my deepest gratitude to my advisor, Dr. Sanjit Krishnan Kaul, for his guidance and support. I am grateful to have an advisor like him who always helped me whenever I got stuck at some problem. He always motivated me to deal with the problem right from the basics. His efforts in my thesis are invaluable. I also like to sincerely thank my co-advisor, Dr. Saket Anand, for discussing my problems and providing his valuable inputs and steer me in the right direction. I like to thank my esteemed committee members, Dr. Gourab Ghatak and Dr. Syan Basu Roy, for agreeing to evaluate my thesis work. I am also grateful to all the members of Wireless System Lab at IIIT Delhi, who have consistently helped me with their inputs and suggestions on my work.

Abstract

Connected Autonomous Vehicles (CAVs) have for long had the attention of the intelligent transportation systems community due to their promise of improving road safety and efficiency via increased perception. CAVs broadly rely on two components: (a) wireless technologies such as DSRC, WiFi, and 5G, to enable information exchange amongst the vehicles and the roadside infrastructure, and (b) a vehicle planner that uses this information along with local information from the vehicle's sensors to find a motion plan that maximizes the vehicle's driving utility. Most existing works on vehicular planning either don't assume any communications network or neglect network constraints and costs. On the other hand, works on vehicular networks ignore motion planing. In our work, we consider motion planning that adapts to the available communications resource. Further, by associating costs with communication, we adapt the use of the network to physical on road constraints.

We consider an on-road environment that consists of an autonomous (ego) vehicle, human driven vehicles, and roadside infrastructure. The ego vehicle would like to optimize its driving utility by using information from its sensors and that obtained by querying the infrastructure over the constrained network while being cognizant of associated costs.

We formulate the above as a reinforcement learning (RL) problem. The ego vehicle would like to learn a policy function, which at every decision instant, chooses (a) a motion planning action responsible for the longitudinal and latitudinal behaviour of the ego vehicle and (b) a communications action that queries relevant information from the infrastructure. We use deep reinforcement learning to make the vehicle learn the optimal policy, in a model-free setting, using a custom made simulator that integrates traffic scenarios, communications, and reinforcement learning algorithms. We demonstrate via simulations, the ability of the ego vehicle to smartly choose communications and planning actions while achieving huge gains in driving utility from the use of communications.

Contents

List of Tables

List of Figures

List of Abbreviations

1. **5G** - 5^{th} Generation
2. **AC** - Actor Critic
3. **ACC** - Adaptive Cruise Control
4. **AV** - Autonomous Vehicle
5. **CAV** - Connected Autonomous Vehicle
6. **DSRC** - Dedicated Short Range Communication
7. **DQN** - Deep Q-learning
8. **EV** - Extended View
9. **IDM** - Intelligent Driver Model
10. **KL** - Kullback-Leibler Divergence
11. **LIDAR** - Light Detection and Ranging
12. **LV** - Local View
13. **MDP** - Markov Decision Process
14. **MOBIL** - Minimizing Overall Braking Induced by Lane-change
15. **PID** - Proportional Integral Derivative
16. **POMDP** - Partial Observable Markov Decision Process
17. **PPO** - Proximal Policy Optimization
18. **RADAR** - Radio Detection and Ranging
19. **ReLU** - Rectified Linear Unit
20. **RRT** - Rapidly-exploring Random Trees
21. **RL** - Reinforcement Learning
22. **SGD** - Stochastic Gradient Descent.
23. **TRPO** - Trust Region Policy Optimization
24. **V2I** - Vehicle to Infrastructure
25. **V2V** - Vehicle to Vehicle
26. **V2X** - Vehicle to Infrastructure/Vehicle
27. **Wi-Fi** - Wireless Fidelity

Notation Used

- $\bar{x}$, $\bar{X}$ - State estimate of x and state estimates set.
- x, X - System state and state space.
- k - Time step in episode.
- T - Length of the episode.
- Δt - Slot duration.
- U_k - Set of feasible actions at time step k. For simplicity we at times drop the time index.
- $u_k \in U_k$ - Action taken at time k that must lie in the feasible set.
- $u_k^{(l)}$ - Ego vehicle motion planning action at time k.
- $u_k^{(q)}$ - Agent communication action at time k.
- $U_k^{(l)}$ - Set of feasible motion planning actions at time k.
- $U_k^{(q)}$ - Set of feasible communications actions at time k.
- g_k - Stage cost at time k.
- α - Discount factor that discounts costs that result in the future from a current action. We require $0 \leq \gamma < 1$.
- $\hat{g_k} = \sum_{k=0}^{T} \alpha^k g_k$ - Cost-to-go starting from time k till T.
- $\mu(u|\bar{x}_k)$ - Agent policy probability mass function (PMF).
- $\mu^*(u|\bar{x}_k)$ - Optimal policy PMF, which minimizes expected sum-cost.
- $R_b^{(l)}$ - Information rate of local-view.
- $R_b^{(q)}$ - Information rate of extended-view.
- $f_k(x_k, u_k, w_k)$ - State evolution function.
- $J_\mu(x_0)$ - Expected discounted sum-cost starting in start state x_0.

Chapter 1

Introduction

Connected autonomous vehicles (CAVs) are transformative technology that are expected to improve road safety, enhance the quality of life, and improve the efficiency of transportation systems [1]. A CAV uses local sensors such as LIDARs, RADARs and cameras to perceive its surroundings, and wireless technologies such as DSRC, Wi-Fi, and 5G to allow information exchange to expand the perceived region. The information from these *local* sensors is almost instantly available with negligible sensing delay at every decision step [2]. A connected autonomous vehicle can extend its range of perception by querying information about the vehicles and the on-road artifacts that don't lie in the view of its local sensors from road-side infrastructure that may have access to the same. We will refer to the region around the ego vehicle that it can perceive using its own sensors as it *local view*. The region that it may query from the infrastructure will be referred to as the *extended view*. Figure 1.1 provides an illustration. The ego vehicle queries its extended view from the infrastructure over a constrained communications network.

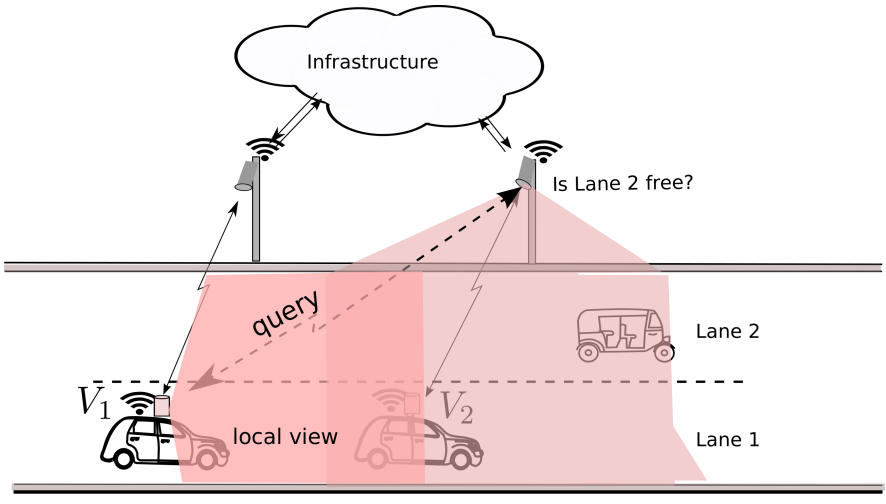


Figure 1.1: Illustration of the on road environment. A Vehicle may query the infrastructure for information that it uses to extend its local view. The infrastructure makes its own measurements and may receive measurements made by other vehicles too. The local view and the extended view are shown for the vehicle V_1.

Work that studies the impact of communication constraints on the driving utility is limited. Most of the work that considers communication and planning assumes no constraints on the communication and often treats it as a free resource. However, in practice, communication constraints do exist and there are associated costs. Specifically, transmission delays and limited bandwidth impact the ego vehicle's ability to query its extended view. This should influence the planner's motion plan and the resulting driving utility. Conversely, high-density traffic diminishes the need to communicate with other vehicles over the network. Given the costs of communicating and, more generally, to ensure an efficient use of the constrained communication resource, it becomes crucial to learn when and what regions of the extended view, if any, need to be queried from the infrastructure with the goal of optimizing the vehicle's driving utility.

Motivated by the above observations, we investigate joint selection of communication and planning actions such that the driving utility of the ego vehicle is optimized. We consider a vehicle-to-infrastructure (V2I) network architecture, where the infrastructure is assumed to always have current information of all on road artifacts in the extended view of the ego vehicle.

1.1 Contributions

Our main contributions are as follows.

- We formulate optimizing the driving utility of an autonomous vehicle that smartly leverages the network to query the infrastructure to populate its extended view as a reinforcement learning problem. At every decision instant, the RL policy uses the information obtained from the ego vehicle's sensors together with information it has received over the network to choose its next motion planning and communication action. The optimal policy is learnt for the given communication constraints and costs.

- We use deep reinforcement learning to learn the optimal policy using the proximal policy optimization [3] (PPO) algorithm by repeatedly interacting with our custom simulation environment.

- We developed a simulation environment that can simulate various traffic conditions with a single autonomous vehicle, human driven vehicles, traffic lights, and one or more lanes. We implemented the intelligent driver model for longitudinal [4] and MOBIL [5] for the latitudinal behaviour of the human driven vehicles. The simulator also enables smart querying of the extended view. It further allows easy integration with RL algorithms.

- We empirically demonstrate how the gains of using communications are impacted by assumptions about network delays and how delayed information is used by the ego vehicle's policy. We also demonstrate how traffic density impacts the the usefulness of communications. Our learnt policy demonstrates the need of communication diminishes as the traffic density increases. Further, we introduced a keep strategy which retained

the old information for non queried regions and this approach brought huge gains to the ego vehicle's driving utility. This tells us that retaining old information may help alleviate a constrained communication with the infrastructure.

1.2 Thesis Organization

The thesis is organised as follows.

- **Chapter 2** - Related Works
  - We review the current work and literature around topics such as cooperative planning, vehicular communication, and the application of reinforcement learning to these problems.
- **Chapter 3** - Model and Optimization Problem
  - We discuss in detail the model and the optimization problem.
- **Chapter 4** - Grid Based Representation
  - We discuss a grid based representation of the ego vehicle's observation that tracks the occupancy and speed of an occupant in every cell of the grid.
- **Chapter 5** - Learning An Optimal Policy using Deep Reinforcement Learning
  - This chapter discusses in detail the policy representation and the RL algorithms such as the policy gradient algorithm. We also discuss in detail the proximal policy optimization algorithm, which we used to learn the optimal policy for our experiments.
- **Chapter 6** - Simulation and Training Setup
  - This chapter discusses the details of the simulation platform that we used to simulate vehicle-to-infrastructure network and the training flow that was used to learn the optimal policy.
- **Chapter 7** - Results and Conclusions
  - This chapter discusses various simulation scenarios and provides an extensive evaluation of the ego vehicle's performance.

Chapter 2

Related Works

We summarize works that consider cooperative planning and perception for mixed autonomy settings, vehicular communications for planning, and the application of reinforcement learning to these problems.

2.1 Cooperative perceptions and planning

There exist extensive works [6], [7] and [8] that study cooperative planning and perception to improve traffic safety and flow. [7], [8] and [9] talk about increasing the perception of the ego-vehicles and its effect on improving the traffic flow. [6] and [7] combine this information to create a merged occupancy map used by the planner for path planning. The authors also conclude that critical safety systems should rely on local sensing information. The longer-term decisions such as lane change or lane-keeping can benefit from remote sensing. Authors in [8] use the Bayesian approach to include uncertainty in the perception module and communication delays to create a merged occupancy grid over which the RRT (Rapidly-exploring Random Trees) algorithm is used for path planning. However, all the schemes discuss broadcasting of information and do not consider the usefulness of the sensing information received by the ego vehicle. This naive approach of collecting data may lead to unnecessary bandwidth consumption and processing delays.

2.2 Communications and planning

DSRC (Dedicated Short Range Communication) is a wireless technology developed for automotive platforms to support fast, secure, and reliable information exchange between vehicles and infrastructure. The widely adopted implementation [10] of DSRC uses the IEEE 802.11p standard and the 5.9 GHz frequency band for low-latency, high-speed information exchange between the vehicles (V2V) or infrastructure (V2I). Works [11] and [12] talk about a network of cars that can aid each other via cooperation to predict and respond to hazards. The vehicles relay their information, such as speed and position, to other surrounding vehicles using a wireless network. The data is then used to

improve vehicle collision systems. In [11], the authors propose using a reinforcement learning-based approach to enhance the adaptive cruise control through vehicle-to-vehicle communication where the current speed and acceleration are shared with other vehicles.

Works that study the impact of communication on planning are limited. In [9], the authors study the effects of communication constraints such as interference on the maximum vehicle speed in dense networks. In [13], the authors consider a multi-agent setting and learn a policy with each agent broadcasting its local view with no cost for communicating. At execution time, however, a heuristic approach is taken, where an agent decides to communicate when it benefits the team performance. In [14], the authors take a planning-aware communication approach for decentralized coordination of multiple agents. A particle filter framework is used where each agent tracks the action distribution of every other agent and decides to communicate with one only when its local utility can improve with new information.

2.3 RL based perception and planning

Works [15] and [16] provide a framework to simulate mixed autonomy traffic control problems and integrates it with the ray-framework [17] to train deep-reinforcement learning policies easily. The learnt policy shows behaviors of stabilization and platooning, which are known to improve ring road efficiency for such traffic settings. Works [18] and [19] use RL methods such as Deep-Q-learning to train policies for a roundabout. These methods perform well in uncertain environments but fail to generalize in dynamic environments and with continuous action spaces. [20] shows that stop-and-go waves occur in high-density traffic even without any geometric or lane-changing maneuvers. These traffic waves can be dampened by controlling the speed of a few vehicles in the flow. They conducted a field experiment that consists of a circular track with 20 human-driven vehicles and a single vehicle whose speed and acceleration are controlled via a PID controller. The experiments show smoother ride, lesser braking, less fuel consumption, and higher throughput with a penetration rate of autonomous vehicles as low as 5%.

Chapter 3

Model and Optimization Problem

3.1 System model

We assume a discrete-time model with time slots indexed by $k = 0, 1, 2, \ldots T$. The slot duration Δt is assumed to be fixed for all time slots. The *decision frequency* f_d, is given by $1/\Delta t$ which describes how often the simulation environment is updated. The time evolution of the environment can modeled by a discrete-time dynamical system [21].

$$x_{k+1} = f_k(x_k, u_k, w_k),\ k = 0, 1, \ldots, \tag{3.1}$$

where x_k is the state of the system which includes the speeds and positions of all the vehicles at time k, and is not directly visible to the ego-vehicle, u_k is the action that the ego vehicle chooses at time k, and w_k captures any random disturbances in the environment and uncertainty in the next positions and speed of human driven vehicles. As a consequence of choosing the action u_k in state x_k, the system transitions to state x_{k+1} at time $k + 1$. Note that, if it is given x_k, the ego vehicle has all it needs to make its best choice of action at k.

3.2 Observation

The state of the system x_k is not directly visible to the ego vehicle. Instead, an observation $o_k = [o_k^{(l)}, o_k^{(q)}]$, is available to the ego vehicle for decision making at k. This observation is a vector of size $n \times 1$. It consist of two vectors $o_k^{(l)}$ and $o_k^{(q)}$. The former is a $n_l \times 1$ vector that is an observation of the local view made by local sensors, and the latter is a $n_q \times 1$ vector for the extended view, which is refreshed by making a query.

3.3 Action

The action $u_k = [u_k^{(l)}, u_k^{(q)}]$ consists of vectors $u_k^{(l)}$ and $u_k^{(q)}$. The vector $u_k^{(l)}$ is the motion planning action that is responsible for longitudinal and latitudinal behaviour (lane-changing) of the ego-vehicle on the road. The $u_k^{(q)}$ is a communication action that determines the query the ego-vehicle makes to the

infrastructure. The query made at time k given by $u_k^{(q)}$, populates one or more elements in $\bar{x}_{k+1}^{(q)}$.

3.4 Action constraints

Let $U_k^{(l)}$ and $U_k^{(q)}$, respectively, be the set of all feasible planning and communication related actions. The set $U_k^{(l)}$ restricts the on-road maneuvers that ego-vehicles can perform. For example, $U_k^{(l)}$ may specify the set of allowed velocities and accelerations. The set $U_k^{(q)}$ of possible communication actions restricts what an ego-vehicle can query and obtain from the infrastructure using the communication network. An example of communication-related constraints is the limit on the amount of data that can be queried from the infrastructure at time k. At any time k, u_k is feasible only if $u_k^{(l)} \in U_k(l)$ and $u_k^{(q)} \in U_k(q)$.

3.5 State Estimate

The ego vehicle may obtain an estimate $\bar{x}_k$ of the underlying state at time k by using its history of observations $o_0, o_1, \ldots, o_k$ and actions $u_0, \ldots, u_{k-1}$. The estimate $\bar{x}_k$ is used by the ego vehicle to decide on its action u_k at time k.

3.6 Cost function

At any time k, given that ego-vehicle estimates $\bar{x}_k$ and takes action u_k, it incurs a bounded stage cost $g_k(x_k, u_k(\bar{x}_k), x_{k+1})$. Let X be the set of all states and $\bar{X}$ be the set of all the state estimates. We define a stationary (independent of time k) policy μ, which is a function that maps every state estimate $\bar{x}_k \in \bar{X}$ to a feasible action u_k in $U_k^{(l)} \times U_k^{(q)}$. Let $J_\mu(x_0)$ be the finite-horizon discounted expected sum cost of policy ,when the ego-vehicle starts in state x_0 and follows the policy μ thereafter.

$$J_\mu(x_0) = E\left[\sum_{k=0}^{T} \alpha^k g_k(x_k, \mu(\bar{x}_k)|x_0\right], \tag{3.2}$$

where $0 < \alpha < 1$ is the factor that discounts future costs. The discount factor is used to set how important rewards further in time are relative to rewards that may be accrued closer to the current time. Setting a smaller value of α, makes the ego vehicle short sighted and a larger α makes it far sighted. The expectation is over the sequence of the states ego-vehicle encounters, and policy μ, given that the initial state is x_0.

3.7 Optimization problem

The optimization problem is find the optimal policy μ^* that minimizes the the expected sum cost $J_\mu(x_0)$, for all $x_0 \in X$. Let Π be the set of admissible policies.

The optimal policy is

$$\mu^* = \arg\min_{\mu \in \Pi} J_\mu(x_0),\ \forall\ x_0 \in X. \tag{3.3}$$

The policy μ^*, at every time step k, chooses the optimal action u_k^*, which contains an optimal *motion planning* and a *communication action*, as a function of state estimate $\bar{x}_k$, such that expected discounted sum cost (3.2) is minimized.

Chapter 4

Grid Based Representation

We use an occupancy grid to represent the ego vehicle's nearby environment. Occupancy grids [22], [23] are often used in robotics, for example, algorithms such as RRT that do path planning. An occupancy grid represents the continuous on-road space around the ego vehicle as set of discrete cells.

Each cell of the grid is either empty or occupied (has a static object or a moving object). Since the objects in the cells may be moving, we also associate a speed with each cell. If a cell is occupied by static objects, the speed is 0.

The *local view* part of the grid encodes the occupancy and speeds around the ego-vehicle measured by the local sensors. The occupancy and speeds of the cells that span the extended view can only be obtained only by querying the infrastructure. The *cell-size* dictates the number of the cells required to encode a given local or extended view. Figure 4.1 illustrates occupancy grid for a *local-view* of size $8m$ and cell resolution of $1m$. A large value of the *cell-size* implies that fewer cells are used to encode a given region leading to poor resolution [24]. In practice, the occupancy information is obtained by fusing information from sensors, which is used to generate a point-cloud map. This point-cloud map is projected to the 2D space [24], followed by filtering of noise.

4.1 Agent's Observation in grid representation

The observation o_k, which consists of vectors $o_k^{(l)}$ and $o_k^{(q)}$ at time k, is obtained from the grids. The vector $o_k^{(l)}$ consists of the occupancy and speed information of each cell that lies in the local view and can be measured by the ego vehicle's own sensors. We assume that these occupancy and speed measurements are current and noise free. The vector $o^{(q)}$ consists of occupancies and speeds corresponding to the cells that lie in the extended view. Their information can be obtained by querying the infrastructure. The occupancies and speeds of the cells that are not queried are initially assumed unknown. Later we consider the possibilities that the ego vehicle (a) retains old information for a cell that it may have obtained from an earlier query and (b) at every time step the ego vehicle marks a cell in the extended view that wasn't queried as unknown.

For all time steps, the size and resolution of the local view and extended view are kept fixed. We use a 4-byte integer and a float, respectively, to encode the

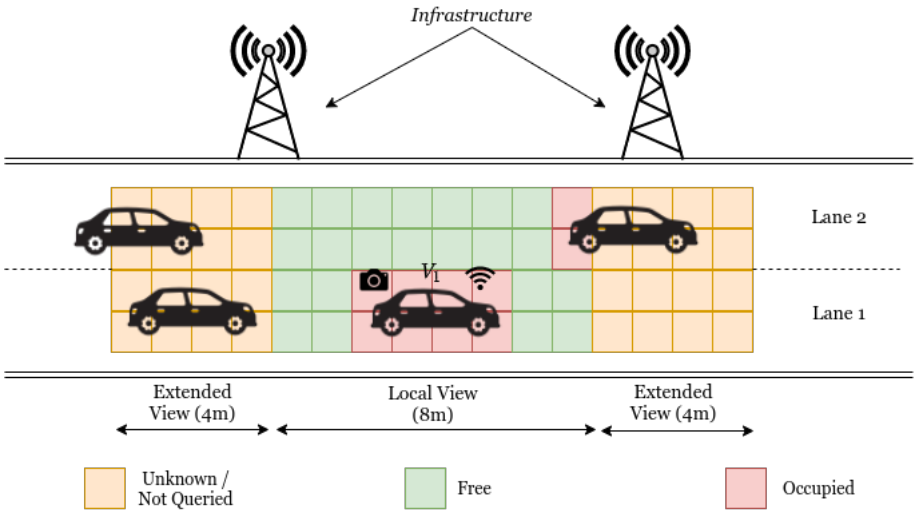


Figure 4.1: Illustration of the occupancy grid based representation for a local view of $8m$ and an extended view of $8m$. The cell resolution is $1m$. Ego-vehicle V_1 creates occupancy grids relative to its position. The actual occupancy of the local view is always available. The occupancy of the extended view is by default unknown and can be obtained by querying the infrastructure.

occupancy and speed for a cell. Let $R_b^{(l)}$ and $R_b^{(q)}$ be the amount of information generated by the local view and the extended view per second. The information rates from the views are, respectively,

$$R_b^{(l)} = f_d \times (n_l \times 8 \times 8)\,\text{bits/sec and} \tag{4.1}$$

$$R_b^{(q)} = f_d \times (n_q \times 8 \times 8)\,\text{bits/sec}, \tag{4.2}$$

where f_d is the decision frequency, which was defined in Chapter 3, n_l and n_q are the number of cells used to encode local-view and extended view occupancy and speed information, respectively.

4.2 Actions in the grid representation

Recall from Chapter 3 that the set $U_k^{(l)}$ is the set of motion planning actions the ego vehicle can perform. We allows the actions (a) accelerate, (b) decelerate, (c) do-nothing, and (d) change lane. Accelerate and decelerate actions are used, respectively, to increase and decrease the speed of the ego-vehicle. The do-nothing action keeps the speed and lane of the ego vehicle unchanged.

Recall that the set $U_k^{(q)}$ contains the communication actions that the ego vehicle may choose. We group one or more cells in the extended view together. The extended view is captured by a union of such mutually exclusive groups. The set $U_k^{(q)}$ includes the action of querying each such group and also the action

of not querying any group. The total number of actions in the set is therefore one more than the total number of groups. Note that in case a query is made, exactly one group can be queried. In our work, the groups are picked a priori and stay fixed.

The maximum number of cells any group can have is constrained by the network bandwidth. Larger number of cells in a group imply the ego vehicle has access to a larger available communication bandwidth (bits/sec or equivalently bits/time slot). On the other hand, a smaller maximum number of cells in a group implies access to a smaller communication bandwidth. Since we fix groups, we assume that the communication bandwidth constraint is fixed for all time. For example, Figure 4.2 shows the extended view of length 8m divided into four groups of mutually exclusive cells, where each group consists of 8 cells.

In case a group is queried by the ego vehicle at time k, it receives the occupancy and speed information of every cell in the group at time $k+1$. We consider the two possibilities of it receiving information that was current at k and is current at $k+1$.

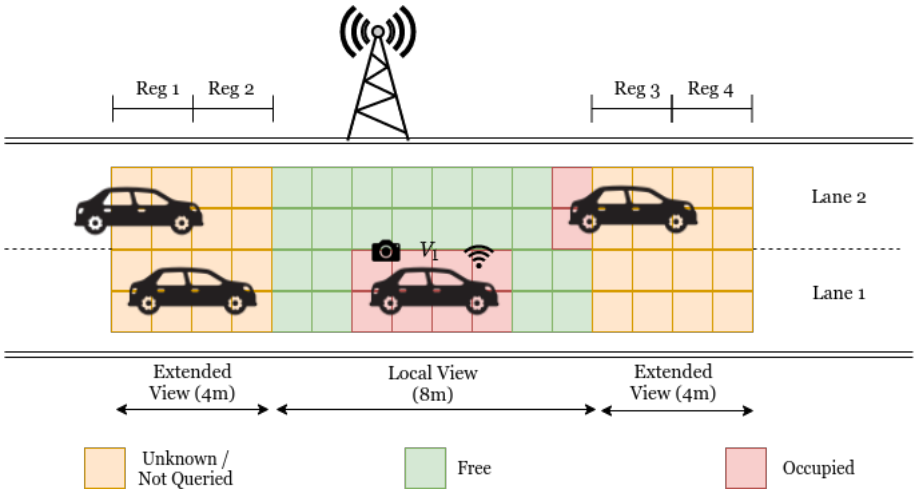


Figure 4.2: Cells in the extended view are organized as groups. We have four mutually exclusive groups named Reg 1, Reg 2, Reg 3, and Reg 4. Each has a size of 512 kilo bits (calculated using (4.1) and (4.2)).

Chapter 5

Learning an Optimal Policy

In this section, we describe our deep reinforcement learning based approach to learning a policy. We focus on model-free methods that don't require to know the state transition model and are suitable for learning policies via repeated interactions at discrete time steps with the environment in simulation.

Given that every cell in both the local and extended views may either be empty or occupied and that we must keep track of the speed associated with each cell, the state space, even for a modest number of cells can become very large. This together with the fact that at any time k, the vehicle doesn't have access to the state but only has access to the history of observations and its actions, (imperfect state information), an estimate $\bar{x}_k$ of state at k must incorporate (a subset of) the history of observations and actions. Given the resulting large number of elements in the set $\bar{X}$ of state estimates and the likely large action space $U = U^{(l)} \times U^{(q)}$, exact reinforcement learning methods, for example, tabular Q-learning, are not feasible. This motivates us to find an approximation of the optimal policy using deep reinforcement learning methods.

In reinforcement learning parlance we have a Partially Observable Markov decision process (POMDP), which is defined by the tuple $(X, \bar{X}, O, U, P, g, \rho, \alpha, T)$, where $O : X \times \bar{X} \to [0, 1]$ is the joint probability distribution of observations and states. The goal is to find the optimal policy (3.3), which minimizes the expected discounted sum cost given by (3.2).

5.1 Policy representation

We represent the agent's policy using a neural-network $\mu(u|\bar{x};\ \theta)$, parameterized by the weight vector θ. The policy function $\mu : \bar{X} \times U \to [0, 1]$, accepts an agent estimate of the state $\bar{x}_k$ and returns a distribution over all $u \in U^{(l)} \times U^{(q)}$. The $\bar{x}$ is some function of observations $o_k, o_{k-1}, ..., o_0$. In our experiments, we estimate the underlying state x_k, using last four observations, $\bar{x}_k = \{o_{k-3}, o_{k-2}, o_{k-1}, o_k\}$. We used two layered fully-connected neural network, each having 256 nodes, followed by a ReLU activation to represent the policy function. The logits obtained from the last layer are passed through a softmax layer to obtain probabilities (See Figure 5.1). An action u_k at time k, can be obtained from this policy by inputting $\bar{x}_k$ to the network and sampling from the distribution at the output.

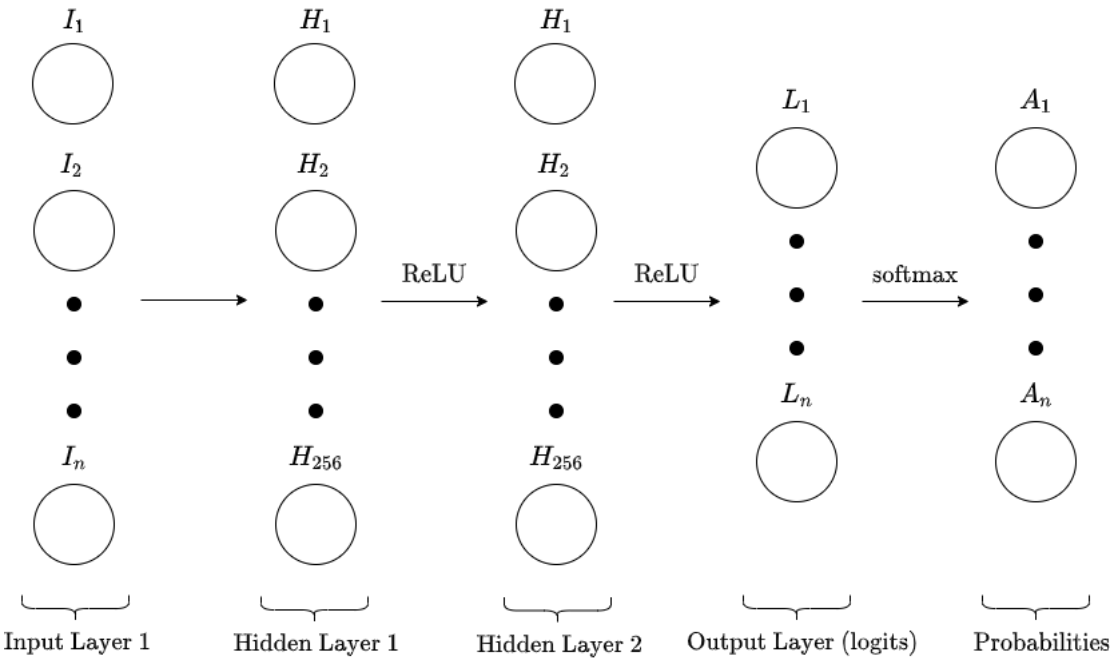


Figure 5.1: The policy function accepts $\bar{x}$ estimate vector as the input to the policy. The input layer is passed through a two layered deep (Hidden Layer 1 and Hidden Layer 2) neural network each of having 256 nodes followed by ReLU activation. The Output layer of the network outputs logit for each action, which are then passed through a softmax layer to obtain action probabilities.

5.2 Policy based learning

We used policy based learning methods. One of such method is the REINFORCE [25] family of algorithms. Standard REINFORCE updates the policy parameter θ in the direction of $\nabla_\theta \left[\log \mu(u_k|\bar{x};\ \theta)(\sum_{k=0}^{T} g_k)\right]$, which is an estimate of the $\nabla_\theta \left[E\left[\sum_{k=0}^{T} g_k(x_k, \mu_k(\bar{x}_k, \theta), w_k)\right]\right]$. This method of updating parameters θ suffers from the issue of high variance in the estimate of gradient. It is possible to reduce the variance of this estimate while keeping the estimate unbiased by subtracting a learned function of $\bar{x}_k$, called as the baseline. The resulting gradient becomes $\nabla_\theta \left[\log \mu(u_k|\bar{x};\ \theta)(\sum_{k=0}^{T} g_k - b_k(\bar{x}_k)\right]$.

A learned estimate of the state-value function $V^\mu(\bar{x})$ is commonly used as the baseline. That is $b_k(\bar{x}_k) = V^\mu(\bar{x})$, where $V^\mu(\bar{x})$ estimates the expected discounted sum cost starting in $\bar{x}$. The approximate state-value function when used as a baseline can be seen as the estimate of the *advantage* of action u_k in $\bar{x}_k$. The *advantage* is defined as $A(u_k, \bar{x}_k) = Q^\mu(u_k, \bar{x}_k) - V^\mu(\bar{x}_k)$, which measures the relative usefulness of action u_k. This approach can be viewed as the actor-critic architecture where the policy μ is the actor and the baseline b_k is the critic [26].

Policy gradient methods alternate between interacting with the environment to collect samples and using those samples to estimate the gradient to update the

policy parameters. However, it is seen in practice that multiple steps of updates using the estimate of the gradients above often leads to destructively large policy updates [3], which may cause the policy to diverge or get stuck amongst poor policies (local minima). In TRPO [27], the authors used a surrogate objective function which penalizes the large policy updates. The penalized version is given by

$$\min_{\theta} \left[\frac{\mu_\theta(\mu_k|\bar{x}_k)}{\mu_{\theta_{\text{old}}}(\mu_k|\bar{x}_k)} A_k - \beta \, \text{KL}[\mu_{\theta_{\text{old}}}(.|\bar{x}_k), \mu_\theta(.|\bar{x}_k)] \right], \tag{5.1}$$

where A_k is the advantage estimate for action $u_k = \mu(\bar{x}_k)$, θ_{old} is the vector of parameters before update and β is KL penalty.

In [3], the authors provide a method of computing penalty coefficient β, based upon the KL-divergence d between updated and old-policy.

$$d = \mathbb{E}_k \left[\text{KL}[\mu_{\theta_{\text{old}}}(.|\bar{x}_k), \mu_\theta(.|\bar{x}_k)] \right]. \tag{5.2}$$

The β is updated as follows.

- If $d < d_{\text{targ}}/1.5, \beta \leftarrow \beta/2$
- If $d > 1.5 d_{\text{targ}}, \beta \leftarrow \beta \times 2$

With this scheme, policy updates which are significantly different than d_{targ}, are rare. Further, β is a hyperparameter but not very important as algorithm automatically adjusts it.

5.3 Learning the Optimal Policy

We use PPO style updates (5.1) and (5.2) to learn a policy which minimizes the sum cost defined in (3.2). The learning process starts by initializing the parameters of the policy and value function estimator to θ_0 and ϕ_0 respectively. The learning proceeds in an episodic manner. An episode begins by calling RESET() which resets the state of the simulator to x_0 and returns an estimate of it, $\bar{x}_0$. For an estimate of time k, $\bar{x}_k$, the agent infers a joint-action by running the policy and sampling an action $u_k \sim \mu(\bar{x}_k, |; \theta_k)$. The action is then sent to the simulator via STEP(), which executes it and returns the stage-cost $g_k(x_k, u_k, x_{t+1})$, next estimate $\bar{x}_{k+1}$ and episode termination status (bool). The episode termination status indicates whether a collision of the ego-vehicle with another vehicle ended the episode. In such case, the episode is terminated and the new episode must be started by calling RESET(). We collect N such episodes each of T time steps, and for each interaction, we store the current estimate, reward and current estimate of value function which are used to calculate the policy gradient for policy update.

For each collected interaction, we calculate the cost to go, $\hat{g}_k = \sum_{t=0}^{T} \gamma^t g_k$ followed by the advantage estimate $\hat{A}_k = \hat{g}_k - V(\bar{x}_k)_{\phi_k}$, using the current value function V_{ϕ_k}. We update the parameters for the policy and value function as given in equations (5.3) and (5.4) for K epochs, where e is current epoch, θ_e

is the parameter after e epoch. The KL divergence of μ_e and μ_{e-1}, calculates the distance between the old and the new distribution. A larger value indicates that a big update took place and the same is penalized, thereby restricting large policy updates. This PPO based approach is summarized in Algorithm 1.

Algorithm 1 Learning optimal policy μ, using PPO

Initialize: initial policy parameters θ_0, initial value function parameters ϕ_0.
for training episode $e = 0, 1, ...N$ **do**
  Receive initial estimate $\bar{x}_0$, from simulator using **RESET()**.
  for k=1 to T **do**
    Select *motion* planning action $u_k^{(l)}$, by running the policy $\mu(\bar{x}_k|;\ \theta_k)$.
    Select *communication* action $u_k^{(q)}$, by running the policy $\mu(\bar{x}_k|;\ \theta_k)$.
    $u_k \leftarrow \left[u_k^{(l)}, u_k^{(q)}\right]$;
    Execute action u_k, and observe next-estimate $\bar{x}_{k+1}$, stage-cost g_k and episode terminal status *done* using **STEP()**.
    $\bar{x}_{k+1} = \bar{x}_k$
  end for
end for
Compute cost-to-go $\hat{g}_k$, for the collected trajectories.
Compute advantage estimates $\hat{A}_k$, using current Value-function V_{ϕ_k}.
for $e = 1, 2, ...$ to K **do**
  Update the policy parameters by minimizing the PPO-objective for K epochs.

$$\theta_{k+1} = \arg\min_{\theta} \frac{1}{|N|T} \sum^{N} \sum^{T} \left[\frac{\mu_{\theta_{e-1}}(u_k|\bar{x}_k)}{\mu_{\theta_e}(u_k|\bar{x}_k)} \hat{A}_k - \beta\ \mathrm{KL}[\mu_{\theta_e}(.|\bar{x}_k), \mu_{\theta_{e-1}}(.|\bar{x}_k)] \right] \tag{5.3}$$

  Fit the value function by regression on mean-squared error via gradient descent algorithm.

$$\phi_{k+1} = \arg\min_{\phi} \frac{1}{|N|T} \sum^{N} \sum^{T} \left(V_{\phi}(\bar{x}_k) - \hat{g}_k\right)^2 \tag{5.4}$$

end for

Chapter 6

Simulation and Learning Setup

We created an open-source Python framework [GitHub] using *PyGame* and *Numpy* libraries that provides a utility to solve the problem of simulating and learning motion planning and querying policies for mixed traffic settings. The user can access the *environment* using Python APIs which provide an interface to initialize, reset and step through the simulation. The simulation also provides a mechanism of collecting observations, applying actions and defining costs.

Our framework consists of the two major components, the v2i simulator and the Trainer. Figure 6.1 demonstrates the interaction between the v2i simulator and the RLlib library to train a RL based policy. The environment provides all the algorithms and scripts with all necessary information such as observations, actions, stage costs to learn a policy. The v2i simulator is used to simulate the autonomous and human-driver behaviour. Furthermore, the simulator provides a means of simulating the infrastructure, capable of exchanging information with a ego-vehicle via query. In our work, we make use of existing RISELab's RLlib [17] library to implement various RL algorithms. The Trainer uses RLlib library to train an RL agent to learn a policy by interacting with the v2i simulator.

Furthermore, the simulator is compatible with OpenAI's Gym [28] which allows the easy integration of our simulator with other various open-source RL algorithm implementations. RLlib is an open-source library supporting the implementation, training and evaluation of reinforcement learning algorithms. It was created to address the issue of scaling the training process for reinforcement-learning tasks. It supports distributed computation over CPUs and GPUs. The libraries include highly-parallelizable versions of policy gradient algorithms such as TRPO and PPO, out of which PPO is used as the default choice in our experiments.

Central focus in the design of our simulator was the ease of modifying the setup to support mixed-traffic, different traffic densities, vehicle parameters, different sizes of local and extended view and different grid resolutions. We also allow to simulate traffic lights to add perturbations to the speed of vehicles which otherwise is deterministic due to the nature of driving models. Furthermore, the setup allows multiple-lanes, varied vehicles sizes. Once trained, policies can be evaluated on the simulated scenarios in a straightforward manner.

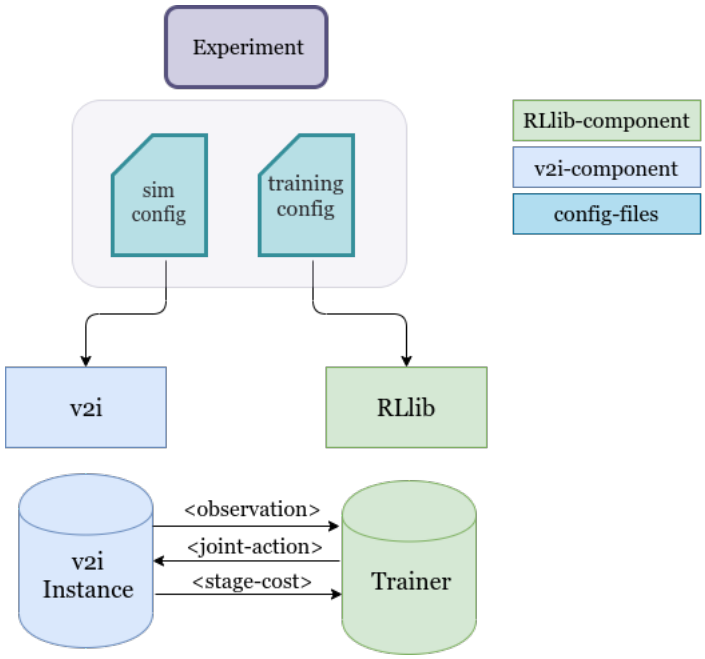


Figure 6.1: Simulator provides components to programmatically control the simulation and training process. An experiment requires two files (a) sim-config and (b) training-config. The v2i simulator accepts a sim-config, which specifies the simulation parameters and returns an instance which can be used to retrieve observation and stage-costs and apply actions. The RLlib accepts a training-config, which defines learning parameters and returns an trainer instance, which interacts with the v2i-instance to train an RL policy.

6.1 v2i simulator

Now, we will briefly describe the internals of the **v2i** simulator. Figure 6.2 shows few different simulation scenarios which can be created using v2i. As described in Figure (6.1), the simulator accepts a *sim-config* file, which defines all the simulation parameters and returns an instance of gym-based environment. Table 6.1 lists the simulation parameters which can be configured by the user to simulate different traffic settings.

1. *Observation* : The simulator keeps track of the position and velocities of each vehicle in the scenario. At every time step, simulator generates an occupancy and velocity grid relative to the position of the ego vehicle for the area defined as the local and the extended view. The perception range of the local sensors is defined using the *local-view* parameter, the size of the extended view is defined using the *extended-reg* and the resolution of the grid is controlled using the *cell-size* parameter. As the ego vehicle and other vehicles move, at every time step, new grids are generated. The cells corresponding to the local view are available to the agent at every time step. However, the cells corresponding to the extended-view are accessed by making an query to the infrastructure.

Simulator Parameter	Purpose
decision-frequency	Time step duration in seconds.
max-speed	Sets a maximum limit on the speed of all the vehicles.
local-view	Size of local view in metres.
total-view	Size of local view and extended view
extended-reg	Size of extended view in metres.
reg-size	Size of each communication region. Should completely divide extended-reg.
cell-size	Resolution of grids in metres.
enable-tf	Enable/Disable traffic lights
density	Sets lane-wise traffic density

Table 6.1: Different simulation parameters supported by v2i simulator and their usage. The parameters can be changed to simulate different scenarios.

2. *Longitudinal controller* : As described in Chapter (3), the valid longitudinal actions are (a) accelerate, (b) decelerate and (c) do-nothing. The simulator uses two different controllers, one for human-driven vehicles and another one for ego-vehicle(s). The longitudinal motion of the human-driven vehicles is simulated using the Intelligent Driver Model (IDM) [4], which is a continuous-time car-following model developed for modeling freeway and urban traffic. The acceleration for each human-driven vehicle is calculated using

$$\dot{v}_k^i = a\left[1 - \left(\frac{v_k^i}{v_0}\right)^{\delta} - \left(\frac{s_k^{*i}(v_k^i, \Delta v_k^i)}{s_k^i}\right)\right], \tag{6.1}$$

where

$$s_k^{*i}(v_k^i, \Delta v_k^i) = s_0 + \max\left[0, \left(v_k^i T_{\text{headway}} + \frac{v_k^i \Delta v_k^i}{2\sqrt{ab}}\right)\right], \tag{6.2}$$

where v_k^i is the speed of the vehicle i, at time k and a is the maximum allowed acceleration, which we set to 0.73 m/s^2. v_0 is the maximum allowed speed for vehicles, which can be adjusted by setting the *max-speed* simulation parameter. s_0 is the minimum gap between vehicles, which is set to $2m$. b is the comfortable deceleration rate, which is set to 1.67 m/s^2. Δv is the speed difference between vehicle i and the vehicle i_{lead} (vehicle just in-front of vehicle i). s_k^i is bumper to bumper distance between vehicle i and i_{lead}. T is the headway time in seconds. The choice of the IDM model parameters is guided by the empirical experiments presented in the [29], which studies several German freeways for a variety of traffic settings such as lane closing and intersections. The acceleration at time k, for human-driven vehicle i, given by $\dot{v}_k^i$ is calculated using (6.1). The acceleration for the ego-vehicle is controlled using an RL policy. The policy can select from three different acceleration rates $\{0.73, 0.0, -1.67\}$

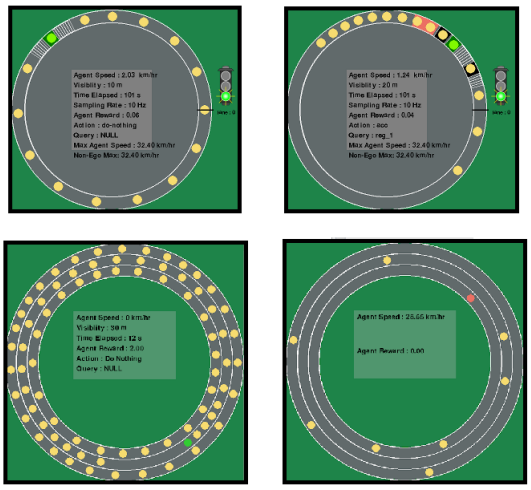


Figure 6.2: Mixed-traffic setting in v2i simulator. Vehicles are represented using circles. Ego-vehicles are colored in green and Human-driven vehicles in yellow. **Clockwise from top left**: single-lane with traffic light and only local view; single-lane with traffic light and extended view (Red portion of the grid denotes the non-queried regions); Multiple lane, no traffic lights and low traffic density scenario; Multiple lane, no traffic lights and high traffic density scenario;

m/s^2. 0.73 corresponds to accelerate, 0.0 corresponds to do-nothing and -1.67 corresponds to decelerate. Given acceleration, the new position and velocity of the vehicle i, is calculated using

$$s^i_{k+1} = s^i_k + \left(v^i_k \Delta t\right) + \frac{1}{2}\dot{v}^i_k \Delta t^2;\ v^i_{k+1} = v^i_k + \dot{v}^i_k \Delta t \tag{6.3}$$

3. *Lateral controller* : As described in Section (4), the valid lateral action is (a) lane-change. Similar to longitudinal control, the simulator uses two controllers. The lane-change for human-driven vehicle is modelled using the MOBIL [5] that models the driver's lane-change behaviour as a function of vehicle's acceleration and bumper-to-bumper distance. The lane-change decision for a vehicle is given by

$$(\dot{v}^i_{k+1} - \dot{v}^i_k) + p\left(\dot{v}^n_{k+1} - \dot{v}^n_{k+1} + \dot{v}^o_{k+1} - \dot{v}^o_k\right) > \Delta a_{\text{th}}, \tag{6.4}$$

where $\dot{v}^i_k$, $\dot{v}^n_k$ and $\dot{v}^o_k$ are the accelerations calculated using the IDM model at time k for vehicle i, its follower n and successor o respectively. The politeness factor $p \in [0, 1]$ controls the aggressive of the drivers. Setting $p = 0$, makes driver egoistic and doesn't consider whether follower and successor will have any disadvantage if it executes the lane-change. Δa_{th} is the *changing threshold* at which lane-change is executed. We set $p = 0.5$ and $\Delta a_{\text{th}} = 0.2\ m/s^2$ for our simulations. The lane-change for the ego-vehicle is controlled using an RL policy. The simulator tries to execute the lane-change if policy outputs a lane-change maneuver. A successful lane-change puts the ego vehicle into the desired lane. When there are

more than two lanes, action space of policy has two lateral actions (a)left lane-change and (b)right lane-change instead of one. An unsuccessful lane-change denotes the collision of ego-vehicle with another vehicle and results in termination of the episode.

4. *Communication controller* : As discussed in Section (4), the communication actions are created by dividing the extended region into a set of mutually exclusive groups of cells. The size of group of cells can be configured by setting the *reg-size* simulation parameter. The reg-size should completely divide the defined extended region. Based upon the defined parameters, the continuous reg-sized communication regions are created. The RL policy decides/learns which region to query given the agent's state estimate. In practice, reg-size represents the bandwidth or limit on the size of the query which can transmitted using a communications network. Additionally, the set of communication actions contains a *Null* region, which doesn't query any region and can save communication bandwidth whenever necessary.

5. *Stage-costs* : To make the ego vehicle learn an optimal policy using deep RL we must set the stage cost (see Equation 3.2) appropriately. The stage-cost can be divided into two sub-costs, namely (a) planner and (b) communication costs. For planner, we use current speed of the ego-vehicle as the reward and normalize it by the maximum allowed speed. To discourage unnecessary lane-changes, we penalize the agent for making a lane-change. As a penalty, we subtract 0.1 from the planner reward. Communication cost is imposed by adding an additional cost of 1.0 to the planner cost, if any region is queried by the agent. For Null query, no cost is added to the planner reward to motivate the planner to save bandwidth whenever it can.

6.2 Trainer

Now, we will briefly describe the internals of **Trainer**. As described above, the trainer is used to train a RL policy and is built on top of the RLlib library. The trainer encapsulates the additional work of setting up the RLlib to train a policy. Similar to v2i simulator, the trainer accepts a *training-config* (see Figure 6.1) to control the training process. Table 6.2 lists the parameters which are common across deep RL algorithms. The algorithm specific hyper-parameters can be tuned by setting the algorithm using *run* parameter and then supplying the algorithm specific parameters to over-write the default values.

The *lr* is the learning rate of the optimizer which controls the size of the parameters update of the policy (actor) and the critic if any. The *number-workers* and *num-gpus* parameters are used to scale up the RL training process. The former is used to control the number of parallel instances of the environment and the later controls the number of GPUs to use for training policy and critic. Higher value of *number-workers* results in better gradient estimate and thereby

Trainer parameter	**Purpose**
run	Deep RL algorithm to use for training.
lr	learning rate for SGD update.
number-workers	Number of parallel instances to run.
horizon	Maximum length of the episode.
grad-clip	The threshold for gradient clipping
num-gpus	Number of GPUs to use for training.
enable-lstm	Enable the use recurrent policies.
fcnet-hiddens	A list containing the size of each FC-layer.
fcnet-activations	Activation function.

Table 6.2: Common training parameters. The parameters can be set to control the various aspects of the RL training process.

results in faster learning and stable policy or critic updates. *enable-lstm* allows the agent to use the lstm layer to model the temporal dependencies. *fcnet-hiddens* and *fcnet-activations* controls the architecture of the policy and critic. The former is a list specifying the size of each fully connected layer and the latter defines the non-linearity to be used between these layers.

Chapter 7

Evaluation and Results

We now demonstrate the efficacy of the learnt policy in optimizing the driving utility under communication constraints. We create different scenarios via the simulation setup discussed in Chapter 6 to capture different settings like transmission delay, limited bandwidth, traffic density and perturbations via the means of traffic lights. We will compare the following scenarios. For each of them, the *decision frequency* and *cell-size* is set to 2.5Hz and 1m respectively.

1. Only Local View (LV10m) - This scenario simulates an ego-vehicle with a local view only. The ego-vehicle has no access to communications. The plan is adapted based upon the local view information only. The size of the local view is considered to be 10m, measured from the center of the ego-vehicle in both directions. We assumed zero sensing delay and therefore, the local-view information is available instantly.

2. Only local view (LV40m) - This scenario is similar to LV10m. However, the size of local view is set to 40m. This scenario acts as a baseline against which we the other schemes.

3. Communication with zero transmission delay (EV40m Inst.) - This scenario simulates an ego-vehicle with a local-view of size 10m and an extended view of size 30m relative to the ego vehicle's position in both directions.

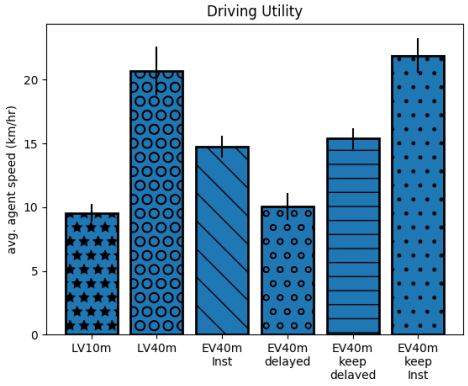
   As per equation (4.2), The rate at which data is generated in the extended view for this scenario is 9.6kb per slot per second. The extended view is divided into two regions, each of size 30m starting from left. Therefore, a single query can query 4.8kb per slot per second out of available 9.6kb per slot per second information with zero transmission delay, i.e. a query made at time t for a region, returns at $t + 1$ the occupancies and speeds of the queried region at time $t + 1$. Hence, the communication network is constrained to query only 4.8 kilo bits of information per slot per second. However, the query returns the latest information.

4. Communication with transmission delay (EV40m delayed) - This scenario simulates the transmission delay in the query. This is similar to the (3), except that a query takes a delay of one decision step, i.e. a query made

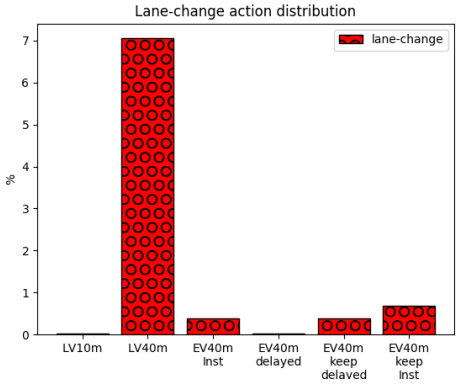
at time t, returns at $t = 1$ the information of the queried region that was current at time t.

5. Communication with zero transmission delay and retain old information (EV40m keep Inst.). Scenarios (3) and (4) mark the non-queried regions as Unknown in the occupancy grids. However, one can argue that we can retain the old information for such regions and let the policy use it. This scenario is similar to EV40m Inst, except that it retains old information for non-queried regions.

6. Communication with transmission delay and retain old information (EV40m keep delayed) - Adds transmission delay to the information returned in the query. Otherwise similar to EV40m keep Inst.



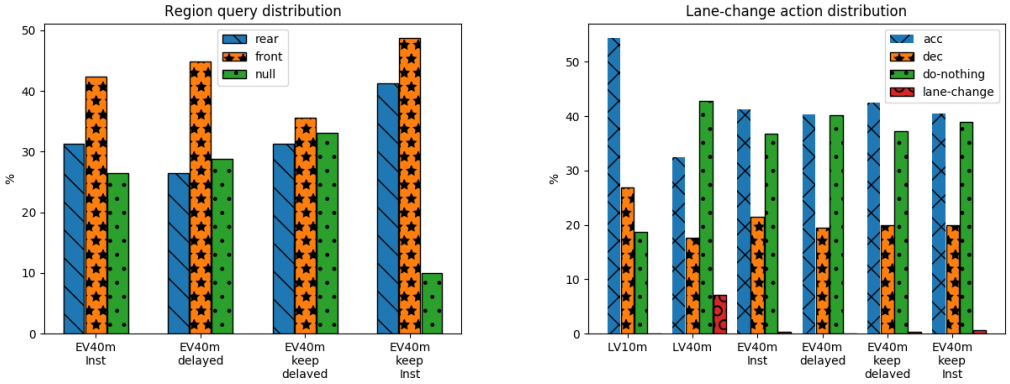
(a) Agent's average speed without perturbations introduced using traffic lights.



(b) Lane change action distribution.

Figure 7.1: Learned policy performance on various scenarios obtained by averaging over 100 independent runs. Figure 7.1a curve summarizes the driving utility obtained by the agents for various scenarios discussed above. Figure 7.1b summarizes the percentage of number of lane changes executed by the agent.

Note that only in EV40m Inst, EV40m delayed, EV40m keep delayed and EV40m keep Inst, the ego vehicle policy jointly chooses planning and communication actions. For LV10 and LV40m the ego vehicle chooses only planning action. For each scenario, we train an RL agent using the PPO algorithm until the algorithm empirically converges. To test the performance of the agent as shown in figures (7.2) and (7.1), we run the learned policy for 100 episodes, each of 2200 time-steps to collect the data. The data is used to analyze the agent's behavior. During the training time, we ensure the behavior of the vehicles doesn't achieve steady state by adding perturbations by the means of introducing traffic lights.



(a) Query actions distribution.

(b) Planning actions distribution

Figure 7.2: Agent's planning and query actions distribution when perturbations are enabled. Figure 7.2a shows the percentage of the regions queried and Figure 7.2b shows the planning action percentage taken by the agent.

7.1 Understanding gains from the increased perception

Figure 7.1a, shows the average driving utility (speed) of the ego-vehicle for different scenarios we detailed above. Not surprisingly, LV40m demonstrate the advantage of having a larger perception over LV10m . The agent in LV40m was able to maintain a higher average speed compared to other scenarios. On the other hand, given the very limited view that the ego vehicle has in LV10m, the average speed of the agent suffered a lot. Observe that the differences between the average speed of the agents. A smaller perception, forces the agent to stick to smaller speed to avoid collisions. Further, from Figure 7.2b, it is evident that the larger perception in LV40m allows for better planning as in LV40m, the agent has a higher percentage of do-nothing and a lower percentage of acceleration and deceleration actions compared to LV10m that has a very high percentage of acceleration and deceleration actions. Also, the significant percentage of lane-changes in LV40m shows the advantage of better lane changes due to increased perception.

7.2 Understanding gains from the addition of communications

LV10m and LV40m demonstrate the advantage of increased perception. However, in practice, the range of an on-vehicle sensors and other on-road occlusions restrict how much an ego-vehicle can perceive. From figure 7.1a, one can observe the advantage of having communication. EV40m Inst. shows the higher average speed of the ego-vehicle with the same size of local view as of LV10m. However, EV40m Inst. is not comparable to an LV40m which has the same size of total-view. This is because of the communication constraint imposed on the EV40m agent, which limits the amount of information that can be queried at once to either the extended view infront of the vehicle or rear of the vehicle. EV40m delayed introduces the one-time step delay in the query that

further deteriorates the driving utility of the agent significantly and making it only marginally better than LV10m. EV40 keep delayed and EV40m keep Inst introduces the scheme of keeping the old occupancies and velocities for EV40m delayed and EV40m Inst. respectively. Figure (7.1a), shows the advantage of keeping the old information as this helps the ego-vehicle to improve its average driving utility. Keeping the old information allows the agent to learn to re-use the old information, which might be queried by the agent previously, for current decision making.

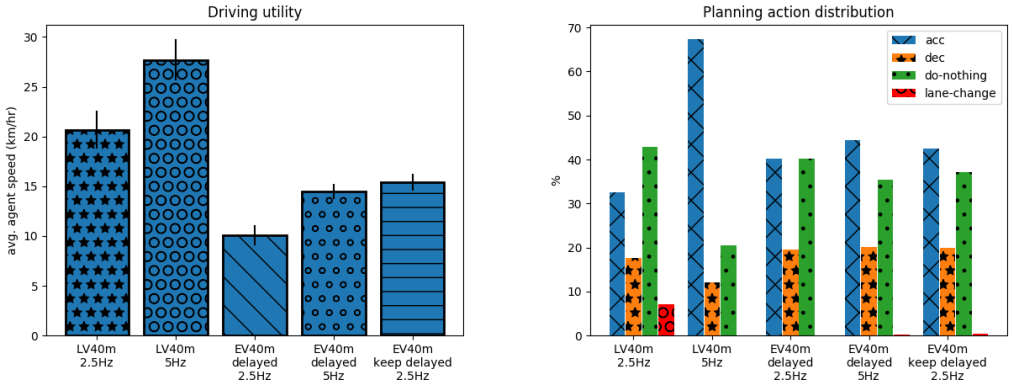
7.3 Adapting motion planning actions to communication constraints

From figure 7.1a and 7.2b, it is clear that in LV40m the ego-vehicle has the highest average speed. Further, the planning distribution of the LV40m demonstrates better path planning as we see a higher percentage of do-nothing action and much less acceleration and deceleration than rest. For all communication-based scenarios (EV), the planning distribution shows a similar trend like LV40m that clearly shows the advantage of communications in terms of better path planning. Access to communications allows the agent to query information that helps the ego-vehicle to plan well in advance. Furthermore, LV10m shows the worse planning of all as its percentage of do-nothing is considerably lower and has a higher percentage of acceleration and deceleration actions. In Figure 7.1b, EV40m Inst. shows the addition of communication makes the agent much less willing to do lane-change. However, keeping the old information (EV40m keep delayed and EV40m keep Inst) for the non-queried regions improves the lane-change capability of the ego vehicle marginally.

7.4 Adapting communications to a constrained communications network

Figure (7.2a), shows the distribution of the queries made by the ego-vehicle for various scenarios. Note that, the query distribution is valid only for scenarios that involve communications. The figure shows that, in all scenarios, the ego-vehicle is slightly biased towards querying the front region than the rear. This is likely to be explained by the fact that any vehicle that may get added to the view will be added towards the front. Hence, querying the front region allows the ego-vehicle to plan ahead. Further, the figure shows that scenarios that don't keep the old information (EV40m Inst. and EV40m delayed) are significantly biased towards querying the front region of the extended view. Also, EV40m delayed is slightly more biased towards the querying front region than EV40m Inst. Keeping the old information allows the agent to query other non-queried regions, as the information for the queried regions can likely be trusted some time steps. EV40m keep delayed and EV40m keep Inst. shows the significant decrease in the bias towards querying the front region.

7.5 Understanding the effect of decision frequency on driving utility.



(a) Agent's average speed. (b) Agent's planning actions distribution.

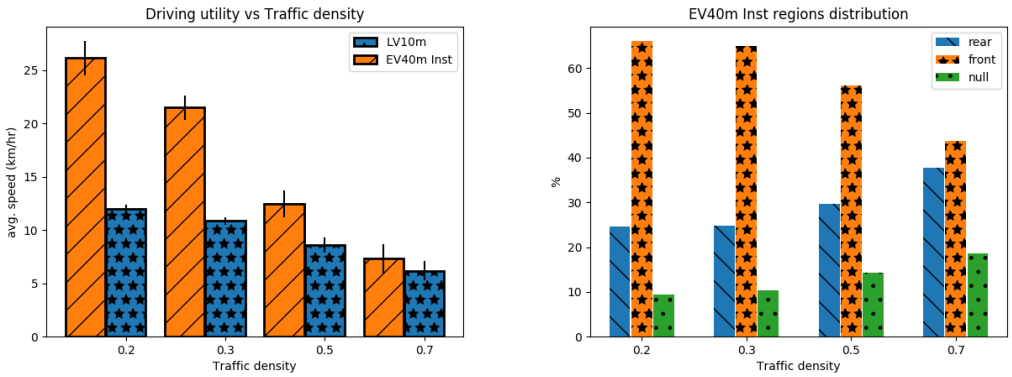
Figure 7.3: Performance of the ego-vehicle for different set of decision frequencies. Figure 7.3a summarizes the agent's average speed with 5Hz and 2.5Hz decision frequency for various scenarios and Figure 7.3b summarizes the actions distribution of the corresponding scenarios.

Figure 7.3, summarizes the average driving utility and planning actions distribution of ego-vehicle with 2.5Hz and 5Hz decision frequencies. From figure (7.3a), it is clear that LV40m 5Hz and EV40m delayed 5Hz has much higher average driving utility than their 2.5Hz counterparts. This shows the clear advantage of having high decision frequency that allows ego-vehicle to improve its driving utility which in our case is maximizing the ego vehicle's speed. Further, comparing EV40m delayed 5Hz and EV40m keep delayed shows that higher decision frequencies can nullify the need of keeping the old information. This can also be thought as the higher decision frequency corresponds to faster updates and if the rate at which underlying information changes is slower than the decision frequency than there is little value in keeping old information.

From Figure (7.3b), LV40m 5Hz and EV40m delayed 5Hz shows significant spike in the acceleration action which directly corresponds to the improved gains in average speed in figure (7.3a). Higher decision frequency allows the ego-vehicle to get the updates at a faster rate which in turn can be used by the ego-vehicle to plan better.

7.6 Understanding the effect of traffic density on motion and communication planning.

To understand the effect of traffic density on agent's behaviour, we train a policy where for each episode the traffic density is sampled uniformly from set $(0, 1]$. Figure (7.4a), summarizes the performance of the learned policy by evaluating the agent on bunch traffic densities with (EV40m Inst) and without (LV10m) communications. It is clear from the figure that as the traffic density increases, the gain from the communication towards driving utility diminishes.



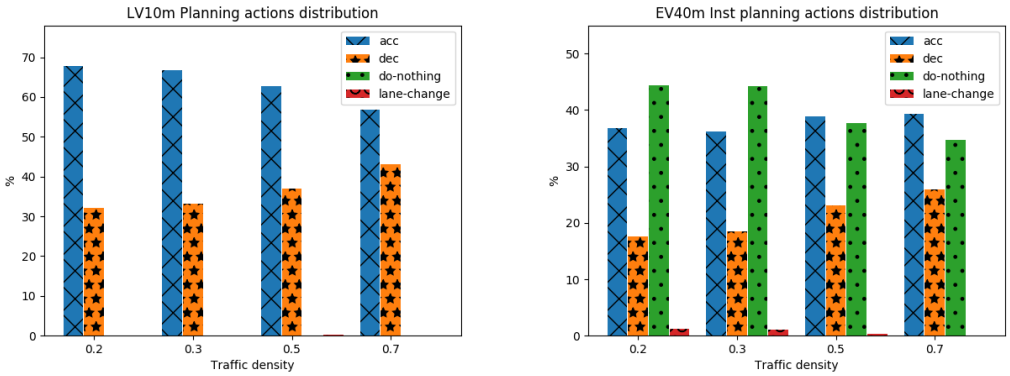
(a) Agent avg. speed for different traffic densities.

(b) Agent query distribution for different traffic densities.

Figure 7.4: Performance of the policy trained on the varied traffic densities. All the performance metrics were obtained by running the 15 independent runs each of 2200 time steps, for each traffic densities using the trained policy. Figure 7.4a summarizes the agent driving utility with(LV10m) and without extended view (EV40m) and Figure 7.4b curve summarizes the region query distribution for EV40m for different traffic densities.

The decrease in the gain is likely due to the fact that the vehicles are so tightly packed that having access to more information doesn't help the agent as it blocked between the slow moving vehicles. Further, from the figure 7.4b, we can observe that when policy is trained for varied densities, the ego-vehicle at lower traffic densities queries the front region more often than read. Also, the number of time agent choose not to query is much lower for lower traffic densities. However, as the density increases, the agent frequency of choosing null query increases as querying doesn't help to improve driving utility.

From Figure 7.5a, we can observe that ego vehicle spent almost all of its time adjusting its speed (high percentage of acceleration and deceleration). Also, the ego vehicle rarely chooses do-nothing and lane change action which demonstrates the agent inability to plan better. The very low percentage of lane changes and do-nothing action is likely due to the smaller perception the ego vehicle has. For lower traffic densities, the ego vehicle choose acceleration more often than deceleration. However, the trend reversed as traffic density increases. From Figure 7.5b, we can observe almost equal amount and high percentage of acceleration and do-nothing actions, which demonstrates the EV40m Inst agent's ability to plan better which is likely due to the increased perception due to communications. Further, we can also observe the agent learns to do lane changes as compared to no rare lane changes in LV10m. AS traffic densities increases, the percentage of lane change also diminishes, which is likely due to the decrease in the opportunity to do successful lane changes due to high traffic density.



(a) Agent planning distribution without communications.

(b) Agent planning distribution with communications.

Figure 7.5: Planning action distribution of the learned policy on varied traffic density. Figure 7.5a summarizes the agent planning actions distribution without communication and Figure 7.5b summarizes the same with communication for different traffic densities.

Bibliography

[1] D. Elliott, W. Keen, and L. Miao, "Recent advances in connected and automated vehicles," *Journal of Traffic and Transportation Engineering (English Edition)*, vol. 6, no. 2, pp. 109 – 131, 2019. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S2095756418302289

[2] Q. Chen, S. Tang, J. Hochstetler, J. Guo, Y. Li, J. Xiong, Q. Yang, and S. Fu, "Low-latency high-level data sharing for connected and autonomous vehicular networks," *2019 IEEE International Conference on Industrial Internet (ICII)*, Nov 2019. [Online]. Available: http://dx.doi.org/10.1109/ICII.2019.00055

[3] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *CoRR*, vol. abs/1707.06347, 2017. [Online]. Available: http://arxiv.org/abs/1707.06347

[4] O. Derbel, T. Peter, H. Zebiri, B. Mourllion, and M. Basset, "Modified intelligent driver model for driver safety and traffic stability improvement," *IFAC Proceedings Volumes*, vol. 46, no. 21, pp. 744 – 749, 2013, 7th IFAC Symposium on Advances in Automotive Control. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1474667016384634

[5] A. Kesting, M. Treiber, and D. Helbing, "General lane-changing model mobil for car-following models," *Transportation Research Record*, vol. 1999, no. 1, pp. 86–94, 2007. [Online]. Available: https://doi.org/10.3141/1999-10

[6] S. Kim, Z. J. Chong, B. Qin, X. Shen, Z. Cheng, W. Liu, and M. H. Ang, "Cooperative perception for autonomous vehicle control on the road: Motivation and experimental results," in *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2013, pp. 5059–5066.

[7] S. Kim, B. Qin, Z. J. Chong, X. Shen, W. Liu, M. H. Ang, E. Frazzoli, and D. Rus, "Multivehicle cooperative driving using cooperative perception: Design and experimental validation," *IEEE Transactions on Intelligent Transportation Systems*, vol. 16, no. 2, pp. 663–680, 2015.

[8] W. Liu, S. W. Kim, Z. J. Chong, X. T. Shen, and M. H. Ang, "Motion planning using cooperative perception on urban road," in *2013 6th IEEE Conference on Robotics, Automation and Mechatronics (RAM)*, 2013, pp. 130–137.

[9] R. Talak, S. Karaman, and E. Modiano, "Speed limits in autonomous vehicular networks due to communication constraints," in *2016 IEEE 55th Conference on Decision and Control (CDC)*, 2016, pp. 4998–5003.

[10] Y. J. Li, "An overview of the dsrc/wave technology," in *Quality, Reliability, Security and Robustness in Heterogeneous Networks*, X. Zhang and D. Qiao, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 544–558.

[11] C. Desjardins and B. Chaib-draa, "Cooperative adaptive cruise control: A reinforcement learning approach," *IEEE Transactions on Intelligent Transportation Systems*, vol. 12, no. 4, pp. 1248–1260, 2011.

[12] E. Moradi-Pari, A. Tahmasbi-Sarvestani, and Y. P. Fallah, "A hybrid systems approach to modeling real-time situation-awareness component of networked crash avoidance systems," *IEEE Systems Journal*, vol. 10, no. 1, pp. 169–178, 2016.

[13] M. Roth, R. Simmons, and M. Veloso, "Reasoning about joint beliefs for execution-time communication decisions," in *Proceedings of the Fourth International Joint Conference on Autonomous Agents and Multiagent Systems*, ser. AAMAS '05. New York, NY, USA: Association for Computing Machinery, 2005, p. 786–793. [Online]. Available: https://doi.org/10.1145/1082473.1082593

[14] G. Best, M. Forrai, R. R. Mettu, and R. Fitch, "Planning-aware communication for decentralised multi-robot coordination," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, 2018, pp. 1050–1057.

[15] C. Wu, A. Kreidieh, K. Parvate, E. Vinitsky, and A. M. Bayen, "Flow: Architecture and benchmarking for reinforcement learning in traffic control," *CoRR*, vol. abs/1710.05465, 2017. [Online]. Available: http://arxiv.org/abs/1710.05465

[16] C. Wu, A. Kreidieh, E. Vinitsky, and A. M. Bayen, "Emergent behaviors in mixed-autonomy traffic," in *Proceedings of the 1st Annual Conference on Robot Learning*, ser. Proceedings of Machine Learning Research, S. Levine, V. Vanhoucke, and K. Goldberg, Eds., vol. 78. PMLR, 13–15 Nov 2017, pp. 398–407. [Online]. Available: http://proceedings.mlr.press/v78/wu17a.html

[17] E. Liang, R. Liaw, R. Nishihara, P. Moritz, R. Fox, J. Gonzalez, K. Goldberg, and I. Stoica, "Ray rllib: A composable and scalable reinforcement learning library," *CoRR*, vol. abs/1712.09381, 2017. [Online]. Available: http://arxiv.org/abs/1712.09381

[18] S. Shalev-Shwartz, N. Ben-Zrihem, A. Cohen, and A. Shashua, "Long-term planning by short-term prediction," *CoRR*, vol. abs/1602.01580, 2016. [Online]. Available: http://arxiv.org/abs/1602.01580

[19] A. Yamaguchi and C. G. Atkeson, "Neural networks and differential dynamic programming for reinforcement learning problems," in *2016 IEEE International Conference on Robotics and Automation (ICRA)*, 2016, pp. 5434–5441.

[20] R. E. Stern, S. Cui, M. L. D. Monache, R. Bhadani, M. Bunting, M. Churchill, N. Hamilton, R. Haulcy, H. Pohlmann, F. Wu, B. Piccoli, B. Seibold, J. Sprinkle, and D. B. Work, "Dissipation of stop-and-go waves via control of autonomous vehicles: Field experiments," *CoRR*, vol. abs/1705.01693, 2017. [Online]. Available: http://arxiv.org/abs/1705.01693

[21] D. P. Bertsekas, *Dynamic programming and optimal control*, vol. 1, no. 2.

[22] A. Elfes, "Using occupancy grids for mobile robot perception and navigation," *Computer*, vol. 22, no. 6, pp. 46–57, 1989.

[23] A. Elfes, "Occupancy grids: A stochastic spatial representation for active robot perception," *CoRR*, vol. abs/1304.1098, 2013. [Online]. Available: http://arxiv.org/abs/1304.1098

[24] Lindong Guo, Ming Yang, Bing Wang, and Chunxiang Wang, "Occupancy grid based urban localization using weighted point cloud," in *2016 IEEE 19th International Conference on Intelligent Transportation Systems (ITSC)*, 2016, pp. 60–65.

[25] R. S. Sutton, D. McAllester, S. Singh, and Y. Mansour, "Policy gradient methods for reinforcement learning with function approximation," in *Proceedings of the 12th International Conference on Neural Information Processing Systems*, ser. NIPS'99. Cambridge, MA, USA: MIT Press, 1999, p. 1057–1063.

[26] T. Degris, P. M. Pilarski, and R. S. Sutton, "Model-free reinforcement learning with continuous action in practice," in *2012 American Control Conference (ACC)*, 2012, pp. 2177–2182.

[27] J. Schulman, S. Levine, P. Moritz, M. I. Jordan, and P. Abbeel, "Trust region policy optimization," *CoRR*, vol. abs/1502.05477, 2015. [Online]. Available: http://arxiv.org/abs/1502.05477

[28] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba, "Openai gym," 2016.

[29] M. Treiber, A. Hennecke, and D. Helbing, "Congested traffic states in empirical observations and microscopic simulations," *Physical Review E*, vol. 62, no. 2, p. 1805–1824, Aug 2000. [Online]. Available: http://dx.doi.org/10.1103/PhysRevE.62.1805