Interacting safely with cyclists using Hamilton-Jacobi reachability and reinforcement learning

Aarati Noronha[1] and Jean Oh[2]

Abstract*— *In this paper, we propose a way for autonomous vehicles to interact with cyclists in a manner that is both safe and optimal. This approach combines Hamilton-Jacobi reachability and deep Q learning to promote better navigation. First we compute a value function that is the solution to a time-dependent Hamilton-Jacobi-Bellman inequality. Second, we deploy this value function to accurately quantify the safety factor of a particular state. This safety factor is then weighted and used as reward for reinforcement learning using Deep Q networks. Finally, this work is novel in the fact that it reflects the latent response between the cyclist and the vehicle while computing the value function. The efficacy of this approach is demonstrated through simulation, comparisons with human behavior and another state-of-the-art algorithm.

I. INTRODUCTION

Existing literature states that the motion of a cyclist is random, slow and unpredictable when compared to other agents encountered on roadways. This makes interactions between cyclists and autonomous vehicles particularly challenging. Recent studies have identified various problem areas pertaining to cyclists when it comes to navigation [1], [2]. They have also analysed and documented factors that might influence cyclist behaviour and perception. However, there exists a dearth amongst algorithms that are successful at following cyclists, passing cyclists and interacting with them at intersections. Most navigation algorithms tend to be overtly cautious in their interactions and do not offer any guarantees towards optimality or completeness. They also fail at quantifying the safety aspect of their interactions [3].

In recent years, a heavy emphasis has been placed on safety when it comes to solving optimal control problems. Work on Hamilton-Jacobi reachability analysis has allowed for solutions to these problems [4]. They guarantee safety by completely barring ego agents from entering user-defined failure zones. But this would come at the cost of sacrificing the algorithm's ability to reach the goal state within a reasonable time duration. Reinforcement learning algorithms such as deep Q learning have proven useful at providing suitable trade-offs between optimality and completeness for traditional optimal control problems [5].

In 2019, Jamie et al. [6] introduced a time discounted Hamilton Jacobi Safety Bellman equation used in conjunction with Q-Learning to render reinforcement learning suitable from the perspective of safety. This paper targetted systems with single agents. We built upon this approach and extended it to a system with two agents.

In summary, our main contributions are a navigation algorithm: 1) that extends the approach proposed by Fisac et al. to a system with two agents, 2) that quantitatively defines the **cyclist's comfort level** to the autonomous vehicle's state 3) that allows for an autonomous vehicle to interact with cyclists in an **optimal** manner in terms of both safety and time taken, 4) that effectively **quantifies safety**: the ability to pass a cyclist at a reasonable speed and reasonable distance.

II. PROBLEM FORMULATION

Current autonomous driving algorithms are too cautious when passing cyclists on the road resulting in sub-optimal interactions in terms of time taken. Human drivers in contrast often end up being overtly aggressive in their maneuvers. The problem formulation is the same as that defined by Fisac 19. [] except we introduce the cyclist as a second agent.

Given a start and a goal, the aim is to generate a trajectory ϵ for the vehicle that maximizes an objective function $V : R^n \times [-T, 0] \rightarrow R$ which is reflective of both safety and time taken.

$$V(x) := \sup_{u(\cdot)} \inf_{t>=0} g(\epsilon(t : x, u(\cdot), d(\cdot))) \tag{1}$$

where u is the control input of the vehicle and d is the disturbance input of the cyclist. It can be interpreted as a sequence generation problem where we are given the start and the goal and the aim is to generate an appropriate control input u at every time instant $t \in T$ given a disturbance d.

The study has three parameters of interest: passing the cyclist whilst maintaining a reasonable distance, passing the cyclist at a reasonable speed, and reaching the goal position in the minimum possible time.

III. RELATED WORK

One of the reasons for the failure to interact safely with cyclists is the lack of naturalistic driving data available. There exists an abundance of crash data that documents accidents involving cyclists [7]. But this data offers no clear insight into the state of the system before the crash, environmental factors or the presence of other agents. Few naturalistic databases but the Safety Pilot Michigan database, the inD dataset [8] and the Stanford Drone Database effectively track events involving cyclists. The Safety Pilot Michigan database is deployed for the purpose of this study.

Typically optimal control problems of the type above have been tackled in literature using Hamilton-Jacobi reachability

*This work was not supported by any organization

[1]Aarati Noronha is with the Department of Mechanical Engineering, Carnegie Mellon University, 5000 Forbes Avenue, Pittsburgh, PA 15206, USA noronha.aarati@gmail.com

[2]Jean Oh is with the Faculty of the Robotics Institute, Carnegie Mellon University, 5000 Forbes Avenue, Pittsburgh, PA 15206, USA jeanoh@nrec.ri.cmu.edu

[9]. Basically the optimal control problem is formulated as a differential game which allows for non-linear inputs and a set of reachable sets to be computed. The reachable set is the zero sub-level set of the viscosity solution of a time-dependent Hamilton-Jacobi-Bellman partial differential equation. There are different ways of formulating this problem and exacting accurate solutions but the afore-mentioned method allows for better generalizations to applications that may be discretized. One drawback of reachability analysis is its computational complexity and corresponding inability to scale well to higher dimensional systems. Also, the algorithm would end up being extremely cautious. The autonomous vehicle would have a tendency to stay in the same state repeatedly to avoid collisions at all costs. It would end up sacrificing its ability to actually reach its target state.

Reinforcement learning algorithms have proven effective in these cases involving navigation by providing suitable trade-offs between optimality and completeness. They are goal oriented wherein the agent maximizes an implicitly or explicitly defined reward function over a number of steps. Jamie Fisac et. al proposed an effective means of combining Hamilton-Jacobi Reachability and reinforcement learning in his paper in 2019. He demonstrated its success through a variety of simulated robotics tasks and its ability to scale to systems of up to 18 dimensions. It opened up an entirely different avenue in terms of safety analysis in the reinforcement learning domain. However, his approach was limited to a system with a single agents.

We drew from this baseline and set up a dual agent system involving cyclist interactions for the purpose of this study. The database deployed is the Safety Pilot Michigan database which has a large percentage of cyclist events. We formulate the cyclist-vehicle problem as a dual agent zero sum differential game. The autonomous vehicle is the control input, while the disturbance signal is the cyclist. A noteworthy addition to existing work is that we introduce a means of modelling the disturbance input as function of the cyclist's latent response to the autonomous vehicle's state. The value function is computed using reachability analysis for a three-dimensional system. It is weighted and fed to the deep Q network as a reward that represents the inherent safety factor of a particular state. The deep Q network undergoes training until convergence.

IV. Proposed approach

Through the proposed navigation algorithm, we aim to provide a safe and optimal means for vehicles to interact with cyclists. The following section delineates the dynamics of the system, safety quantification through reachability, effectively modelling disturbance input and finally deep Q learning for navigation.

A. Baseline: Reachability for safety analysis

This problem involves the formulation of three different sets of states each corresponding to separate levels of safety. The first step towards solving this problem is the definition of the collision set. The states in this set represent collisions in real life between the cyclist and the vehicle. The collision set C_0 is closed and is represented by $g : R^n \rightarrow R$

$$C_0 = \{s \in R^n | v(s) <= 0\} \tag{2}$$

where $g(s)$ can also be thought of as the minimum reward achieved over time by a trajectory starting in state x accounting only for the best possible control input at every time instant.

The next step involves computing a backward reachable set building from the collision zone. This set is the zero level sub-set of the terminal value time-dependent Hamilton-Jacobi-Bellman partial differential equation. It is representative of a set of potentially unsafe states that could eventually result in collision. The backward reachable set is therefore

$$\begin{aligned} B = \{s \in R^n | T \in [-T, 0] : \forall u(\cdot) \in U, \forall d(\cdot) \in D : \\ \epsilon(t : s, u(\cdot), d(\cdot)) \in C\} \end{aligned} \tag{3}$$

where $\epsilon(t : s, u(\cdot), d(\cdot))$ denotes a trajectory in the system at time t. The flow field $f : R_n \times A \times B \rightarrow R^n$is uniformly continuous, bounded, and Lipschitz continuous in s.

In order to effectively quantify the degree of safety of each state in the backward reachable set, we compute the value function associated with each state for the afore mentioned optimal control problem. The value function $v : R^n \times [-T, 0] \rightarrow R$ is representative of the maximum penalty a trajectory accumulates over time despite the best possible control input. It is a cumulative measure of how close the trajectory comes to entering the collision set over time. The solution to this problem is given by:

$$\begin{aligned} min\Big\{g(s) - v(s,t), D_t v(s,t) + H(s, D_s v(s,t))\Big\} = 0 \\ v(s,0) = g(s) \end{aligned} \tag{4}$$

where the resulting Hamiltonian is:

$$H(s,p) = \max_{u \in U} \min_{d \in D} p^T f(s,u,d) \tag{5}$$

The complement of the union of the backward reachable set and the collision set is the set R of states that is extremely safe with no chance of collisions in the given time horizon. This set is the robust solution to the optimal control problem in question. If the vehicle enters any state in the set, it will never end up in a collision. However, this also tends to make the navigation algorithm overtly cautious. It is therefore combined with reinforcement learning to enhance optimality of the solution in terms of time taken.

$$R = (B \cup C)' \tag{6}$$

B. Our approach

Fisac et. al 2019 proposed the afore mentioned method for safety analysis for a system consisting of a single agent. We extended this work to account for systems with multiple agents. The problem is formulated as a dual agent zero sum differential game where the autonomous vehicle is the ego agent and the cyclist is the disturbance input. The system is modelled as an ordinary differential equation as follows:

$$\frac{ds}{dt} = \dot{s} = f(x,u,d) \tag{7}$$

$$\dot{s} = \frac{d}{dt}\begin{bmatrix}\Delta x\\ \Delta v\\ \Delta y\end{bmatrix} = \begin{bmatrix}0 & 1 & 0\\ 0 & 0 & 0\\ 0 & 0 & 1\end{bmatrix}\begin{bmatrix}\Delta x\\ \Delta v\\ \Delta y\end{bmatrix} + \begin{bmatrix}0\\ d-u\\ 0\end{bmatrix} \quad (8)$$

Assume $s \in R^n$. Here s is the state of the system, $u(\cdot)$ is the control input of the autonomous ego vehicle and $d(\cdot)$ is the disturbance input of the cyclist. The state x is basically an array containing the longitudinal range Δx, longitudinal range rate Δv and lateral range Δy of the cyclist with respect to autonomous vehicle.

Assume $u(\cdot) \in U(t)$ and $d(\cdot) \in D(t)$ where

$$U(t) = \{\phi : [t, 0] \to A | \phi(\cdot) \text{is measurable}\}$$
$$D(t) = \{\phi : [t, 0] \to B | \phi(\cdot) \text{is measurable}\}$$

where $A \subset R^{n_u}$ and $B \subset R^{n_d}$ are compact and $t \in [-T, 0]$ for some $T > 0$. The solution to the problem involves solving a terminal value time-dependent Hamilton-Jacobi-Bellman partial differential equation.

C. Main contribution: Modelling disturbance input as a reflection of the cyclist's comfort level

Existing literature failures to account for the fact that cyclists and pedestrians are bound to demonstrate feelings of comfort or discomfort with regards to agents around them. This novel approach highlights the fact that the disturbance input of the cyclist may not be effectively represented by just its acceleration input at every instant. We model the disturbance input of the cyclist as a function of features such as acceleration input and the autonomous vehicle's state. Typically, this disturbance set is a proper sub-set of the original disturbance set.

To effectively represent the cyclist's perception, we deploy an auto-encoder. Labelled data consisting of safe and dangerous states is fed into the auto-encoder. The segregation of data into dangerous and safe states is done on the basis of the Hamilton-Jacobi value function computed in the previous subsection. If the corresponding value function $v(x)$ is positive, there exists a solution by the controller to avoid collisions and the state is classified as safe. If the value function is negative or zero, the collision is inevitable in the future as shown in the equation below:

$$v(s) > 0 \implies \text{safe state} \quad (9)$$

$$v(s) <= 0 \implies \text{dangerous state} \quad (10)$$

The next step involved training a classifier to represent as suitable mapping. Rare event classification poses some serious challenges especially with regards to unbalanced training data. Dangerous states account for only 5 to 10 percent of the existing data. Very often classifiers trained through deep learning are unable to effectively reflect these states and are thus less than suitable for classifying rare events.

The auto-encoder approach for classification is similar to anomaly detection. In anomaly detection, we learn the pattern of a normal process. Anything that does not follow this pattern is classified as an anomaly. For a binary classification of rare events, we used auto-encoders.The process has three separate stages: encoding, decoding and characterization of the target class. We trained the auto-encoder on the data that is labelled safe. Safe states are representative of the normal or desired state of the system. The encoder learns the lower dimensional relationship that exists among these features.

The next part involved reconstruction of features. The entire data-set inclusive of both safe and dangerous states is fed to the decoder. The mean square difference between the input and the output is used for classification. If the reconstruction error is high, it means that the corresponding state was dangerous and vice-versa for safe states. The corresponding mapping w is representative of the cyclist's response to to the autonomous vehicle and is fed into the system for the computation of the afore mentioned backward reachable set.

$$w : (\Delta x, \Delta v, \Delta y, \Delta a) \implies [target, outlier] \quad (11)$$

D. Deep Q Learning for navigation

The solution to the optimal control problem above results in a set of states that are deemed extremely safe. This solution is robust, but it also has a tendency to be overtly cautious. This is undesirable in cyclists vehicle interactions because the optimality of the vehicle's trajectory in terms of time is sacrificed as a result of it being extremely safe. A suitable trade-off between time and safety needs to be reached.

A deep Q network from existing literature with experience replay is deployed. The problem is formulated as a discrete-time Markov process that aims at maximizing the cumulative payoff of a trajectory exponential discounted over time. The value function deployed is an extension of the value function computed in [4] in its discretized form. The policy or the next action is determined by the maximum output of the Q-network The loss function here is mean squared error of the predicted Q-value and the target Q-value. Learning is carried out using the gradient descent algorithm as shown in the equation 12 for $k = 1, 2, ...$ till convergence.

$$\theta_{k+1} \leftarrow \theta_k - \alpha \Delta_\theta E_{s' \sim P(s'|s,u)}[(Q_k(s,u) - target(s'))^2]|_{\theta=\theta_k} \quad (12)$$

$$target = R(s, u, s') + \gamma \max_{u'} Q_k(s', u') \quad (13)$$

V. Experimental set-up

A. Extraction of cyclist events

The database used is a naturalistic database called the Safety Pilot Michigan Database [10]. It has 34 million miles of driving data logged over four years in the Ann Arbor area. The deployment included approximately 2,800 equipped vehicles and 30 roadside equipment. The data was logged using a Mobileye sensor with a range of 98m and a WSU (Wireless Safety Unit). This approach deploys annotated data from three tables: *DataFrontTargets*, *DataWSU* and *DataLane*.

Typically, we define a cyclist event as a situation spanning the time the Mobileye sensor first detects a particular cyclist to the time the same cyclist exits the sensor's field of vision. The primary data used in the detection of a cyclist event is

longitudinal range, longitudinal range rate and lateral range of the ego vehicle with respect to the cyclist. Around 40539 cyclist events were detected. The next step data involved filtering out false positive events which corresponded to any of the categories listed below:

1) An event in which the maximum longitudinal range logged was over 50m. This implied no actual interaction between the cyclist and the vehicle.
2) An event that lasted less than 1 second. This implied noisy sensor data or an interaction of no significance
3) An event in which the cyclist is detected on right side of the road: This is beyond the scope of this study

B. Mapping disturbance input

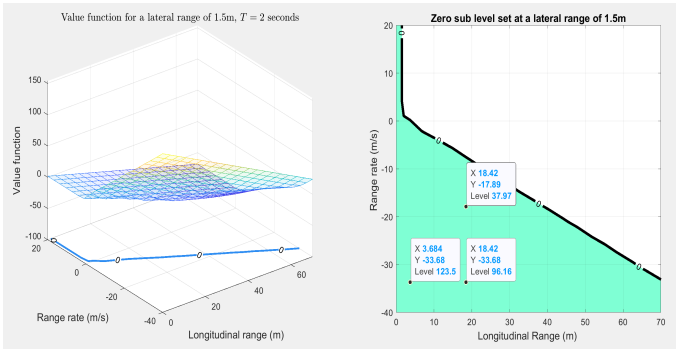
The data-set deployed to model the cyclist's comfort level is the Safety Pilot Michigan Database. Interactions of drivers with cyclists can be analysed as sequences of states and corresponding actions which progress over time. The state x of the system in this case is a tuple consisting of longitudinal range, longitudinal range rate , lateral range and acceleration input Δa of the cyclist. All the values are normalized. Studying the entire interaction over the length of its duration and effectively quantifying disturbance input is somewhat implausible. Each state is classified as either safe or dangerous based on the value function as mentioned in the previous section. The SPMD data was not sufficient to construct a suitable classifier. Synthetic states are fabricated in addition by modifying time to collision and acceleration input of the cyclist. The safe states are fed into the network during the encoding phase. During reconstruction, a mixture of safe and dangerous states is fed into the network. This results in a mapping on the basis of the magnitude of the reconstruction error.

C. Computing the backward reachable set

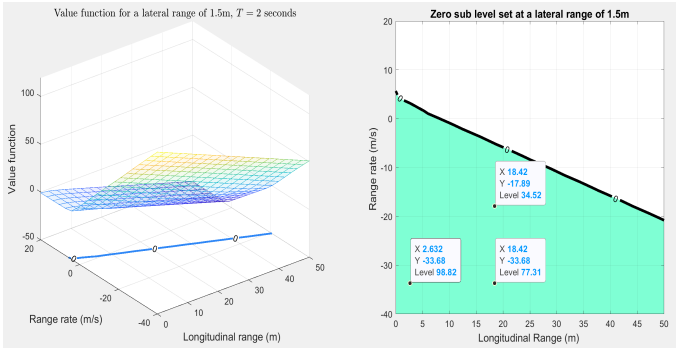
Computing a backward reachable set for every possible state and corresponding control input on behalf of the vehicle is extremely time-consuming. To reduce computational complexity,the state of the vehicle is described relative to the cyclist. The state is basically an array containing the longitudinal range Δx, longitudinal range rate Δv and lateral range Δy of the cyclist with respect to autonomous vehicle. It is pertinent to note that the the state of the system is computed from the boundary of the collision zone. The collision zone in this case is a circle of 1m radius. This allows for a one time computation of the reachable set. Ian Mitchell's level set toolbox is used as the baseline for computation of the backward reachable set. Input to the modified algorithm was the state of the system and the disturbance mapping.

D. Setting up the deep Q network

The state of the system is representative of the longitudinal range, longitudinal range rate and lateral range of the cyclist with respect to the vehicle. It also reflects lane boundaries and the distance of the vehicle from the goal position. The



(a)



(b)

Fig. 1: The value function has been plotted on the left side.For the sake of visualization, lateral range has been restricted to a value of 1.5m. On the right side is the zero sub-level set of the viscosity solution to the Hamilton-Jacobi Bellman partial differential equation. In each of the annotated boxes, X represents the longitudinal range, Y represents the longitudinal range rate and $Level$ denotes the negated value function. The aqua green portion represents the backward reachable set (a) Value function computed using our algorithm (b) Value function computed using the algorithm deployed by Fisac et al. 2019

action space of the autonomous vehicle is discretized to represent several combinations of velocities and yaw angles.

This deep Q network also allows for experience replay. It allocates memory to store experiences and observations of the past. At specific instances during training of the model, a batch of experiences is re-sampled and fed into the neural network. This ensures greater efficacy of the agent in the long run. Correlation is broken and the action values do not oscillate or diverge catastrophically. Adam optimizer is utilized.

VI. Results

To verify the proposed approach, we conducted two sets of experiments. The first set involved comparing the level set computed through our framework with the one deployed by Fisac et al. The second set of experiments involved evaluating the test trajectories qualitatively and quantitatively.

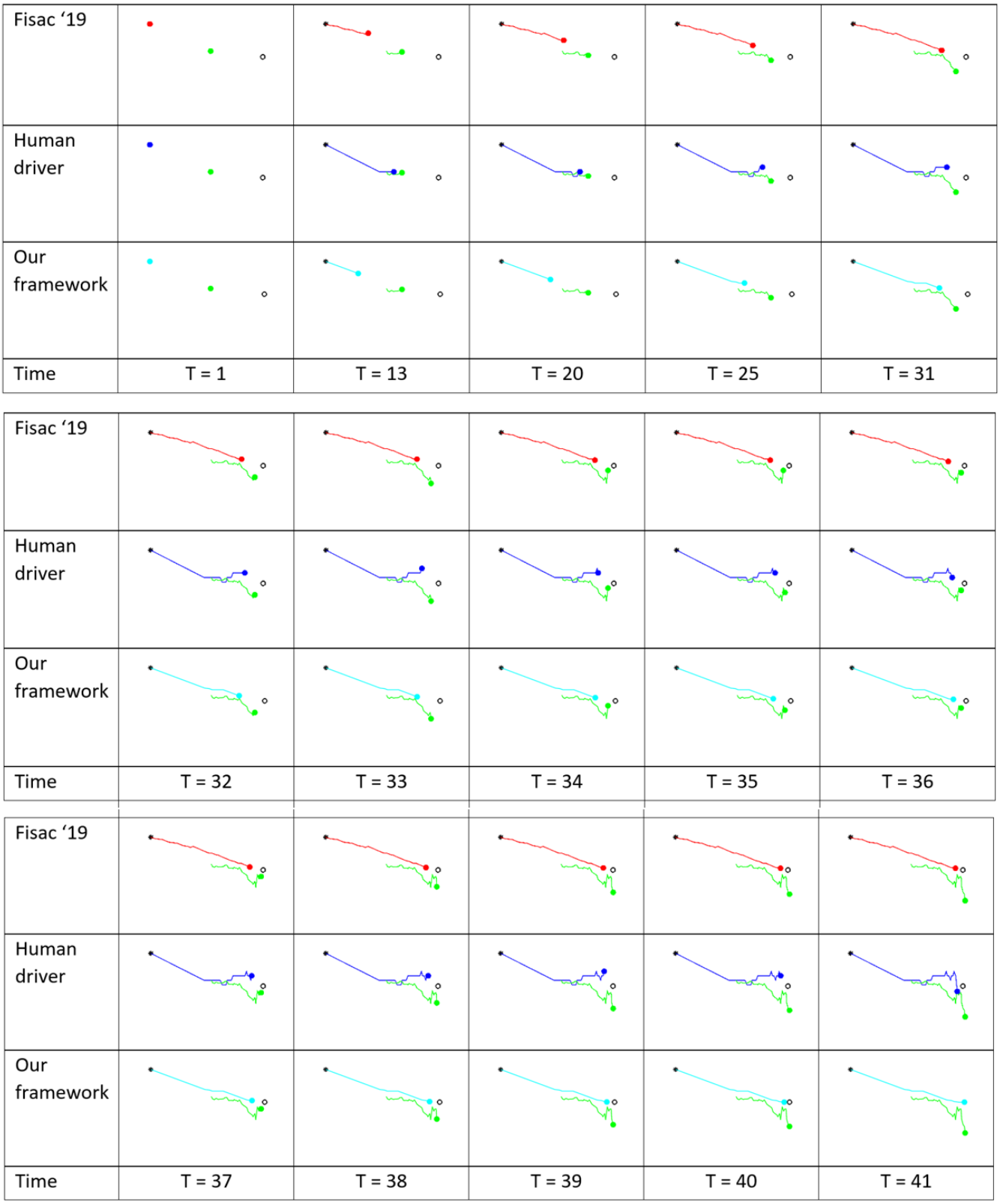


Fig. 2: A visualization of a cyclist event from the test set. A red line denotes the trajectory computed using the level set deployed by Fisac et al. A dark blue line denotes the ground truth i.e. the behavior of human drivers. A cyan blue line indicates the trajectory traversed by our framework. A green line denotes the trajectory of the cyclist. The position of each of these agents at a particular time instant is given by a dot on its trajectory in its corresponding color. The start and goal positions are denoted by a black star and black circle respectively

TABLE I: Quantitative evaluation of trajectories on the Safety Pilot Michigan Database.

	Safety Factor	Unsafe States (%)	Time (sec)	Collisions	Goal Reached
Human driver	-2.31	39.0	253	0	1
Fisac '19	-2.18	36.8	**199**	0	1
Our Framework	**-2.01**	**31.6**	233	0	1

Bold text is indicative of the best value corresponding to a particular performance metric. The test data-set consists of 60 trajectories.

A. Evaluating the level sets and corresponding value function

In their paper published in 2019, Fisac et al. demonstrated success when it came to training a single agent using deep Q networks and reachability. We extended this framework to account for a system with two agents and computed the corresponding value functions. We used this level set as a baseline for comparison to the level set we computed by modifying the disturbance input.

The optimal control problem in question has three dimensions. But for the sake of visualization, a slice of the value function is represented in figure 1 at a lateral range of 1.25m. The zero level subset is shown in. Upon observation of the the annotated values in the two images, it is evident that the value functions computed are an effective reflections of the safety of the state.

After carefully comparing the two images, we notice an extra portion of states in the potentially unsafe region corresponding to a longitudinal range of around 2m and a range rate of over 2m/s. This extra portion is the result of the modifications we made to the disturbance input to reflect the reaction of cyclists. These states basically correspond to a scenario where the autonomous vehicle is quite close to the cyclist; but is operating at a much lower velocity than the cyclist. In theory, this would not be an unsafe interaction. But in practice, the cyclist would perceive these states as a threat merely because the autonomous vehicle is too close to it.

We can therefore say that out approach has been able to successfully reflect the perception of humans which is one of the main goals of autonomous navigation.

B. Evaluation of trajectories

We evaluated our algorithm on 60 cyclist events from the Safety Pilot Michigan Database. It is pertinent to note that these events have not been sampled during training. We use five quantitative metrics for comparison as shown in table I:

1) *Unsafe states encountered*: This is the ratio of the number of times a state in the backward reachable set is encountered to the total number of states encountered. It indicates the number of times the vehicle finds itself in a potentially unsafe region
2) *Safety factor*: This is the average Hamilton-Jacobi value function per trajectory to the total number of trajectories in the test set. A lower value indicates a trajectory is more unsafe
3) *Optimality*: This is the cumulative sum of the time taken to complete all the trajectories in the test set given each of their start positions and goal positions
4) *Collision*: This is the number of times the vehicle finds itself within a 1 m radius of the cyclist during testing
5) *Goal Reached*: This is the ratio of the number of times the vehicle reached the goal to the total number of events in the test set.

We compare our algorithm with the one deployed by J Fisac et al. in 2019 and also against the trajectories followed by human drivers from the SPMD Database. Our algorithm supersedes both human driver trajectories and Fisac'19's agent when it comes to safety. While it takes less time on an average to navigate around cyclists as compared to the ground truth, it takes slightly more time than the algorithm deployed by Fisac '19. This can be attributed to the fact that the level set computed in Fisac '19 fails to reflect the level of comfort the cyclist feels during the interaction.

A visualization of one interaction from the test set is shown in figure 2. In this particular interaction, our algorithm works best in terms of time taken, completeness and safety analysis. It reaches the goal state faster than the human driver and Fisac 19's agent. Its path is proven optimal in terms of length from the visualization. Its safety factor is also particularly high.

VII. CONCLUSIONS

This paper successfully demonstrates the use of deep Q networks in combination with the Hamilton-Jacobi Bellman value function to safely interact with cyclists. It is also able to effectively reflect a cyclist's response to the state of the autonomous vehicle in terms of its disturbance input. This can also be interpreted as a really good indicator of the cyclist's comfort. Our framework performs well in terms of time taken and ability to reach the target when tested on events from the Safety Pilot Michigan Database. Safety analysis is also effectively quantified in this work. We are currently working on extending this approach to multi-agent systems for navigation.

VIII. MATHEMATICAL PROOFS

Let $X$and $U \subset U$ be finite discretizations of the state and action spaces, $d \subset D$ and let $f : X \times U \times D \to X$ be a discrete transition function approximating the system dynamics. The proof for deep Q learning follows from the set-up in section IV-D

Here, learning rate α is 0.0001. It satisfies the following assumptions

$$\sum_k \alpha_k(s,u,d) = \infty, \quad \sum_k \alpha_k^2(s,u,d) < \infty \tag{14}$$

A. Contraction mapping

Proof: For all states $s \in X$, prove that $\mid F[Q](s) - F[Q'](s) \mid < k \parallel Q - Q' \parallel_\infty$

$\mid F[Q](s) - F[Q'](s) \mid$
$= \parallel \ (R(s,u,s') + \gamma \max_{u'} Q_k(s',u')) - (R(s,u,s') + \gamma \max_{u'} Q'_k(s',u')) \parallel$
$\leq \gamma(\max_{u'} Q_k(s',u') - \max_{u'} Q'_k(s',u'))$

Assume the first maximum is the larger one without loss of generality, assume u achieves this
$\mid F[Q](s) - F[Q'](s) \mid$
$\leq \gamma \max_{u'}(Q_k(s',u') - Q'_k(s',u'))$
$\leq \gamma \max_{u'}(Q_k(s',u') - Q'_k(s',u'))$
$\leq k \parallel Q - Q' \parallel_\infty$

B. Convergence of Q learning under our assumptions

Our assumptions are aligned with assumptions 1, 2, 3 and 5 in [11]. This basically means convergence is guaranteed in the case of deep Q learning following from the general proof of Q learning convergence for finite-state,finite-action Markov decision processes presented in [29].

.

REFERENCES

[1] P. Apasnore, K. Ismail, and A. Kassim, "Bicycle-vehicle interactions at mid-sections of mixed traffic streets: Examining passing distance and bicycle comfort perception," *Accident Analysis & Prevention*, vol. 106, pp. 141–148, 2017.

[2] J. J. LaMondia and J. C. Duthie, "Analysis of factors influencing bicycle–vehicle interactions on urban roadways by ordered probit regression," *Transportation research record*, vol. 2314, no. 1, pp. 81–88, 2012.

[3] J. Werneke, M. Dozza, and M. Karlsson, "Safety–critical events in everyday cycling–interviews with bicyclists and video annotation of safety–critical events in a naturalistic cycling study," *Transportation Research Part F: Traffic Psychology and Behaviour*, vol. 35, pp. 199–212, 2015.

[4] I. M. Mitchell, A. M. Bayen, and C. J. Tomlin, "A time-dependent hamilton-jacobi formulation of reachable sets for continuous dynamic games," *IEEE Transactions on automatic control*, vol. 50, no. 7, pp. 947–957, 2005.

[5] V. Dhiman, S. Banerjee, B. Griffin, J. M. Siskind, and J. J. Corso, "A critical investigation of deep reinforcement learning for navigation," 2018.

[6] O. Nachum, M. Norouzi, K. Xu, and D. Schuurmans, "Bridging the gap between value and policy based reinforcement learning," in *Advances in Neural Information Processing Systems*, 2017, pp. 2775–2785.

[7] M. Dozza, "Crash risk: How cycling flow can help explain crash data," *Accident Analysis & Prevention*, vol. 105, pp. 21–29, 2017.

[8] J. Bock, R. Krajewski, T. Moers, S. Runde, L. Vater, and L. Eckstein, "The ind dataset: A drone dataset of naturalistic road user trajectories at german intersections," 2019.

[9] S. Bansal, M. Chen, S. Herbert, and C. J. Tomlin, "Hamilton-jacobi reachability: A brief overview and recent advances," in *2017 IEEE 56th Annual Conference on Decision and Control (CDC)*. IEEE, 2017, pp. 2242–2253.

[10] F. Feng, S. Bao, R. C. Hampshire, and M. Delp, "Drivers overtaking bicyclists—an examination using naturalistic driving data," *Accident Analysis & Prevention*, vol. 115, pp. 98–109, 2018.

[11] J. N. Tsitsiklis, "Asynchronous stochastic approximation and q-learning," *Machine learning*, vol. 16, no. 3, pp. 185–202, 1994.

RESEARCH REPORT - Spring 2019

Aarati Noronha
SafeAI Lab, Carnegie Mellon University
5000 Forbes Avenue, Pittsburgh, PA 15217
anoronha@andrew.cmu.edu

Abstract

This report provides a concise summary of the research conducted by Aarati Noronha during the Spring 2019 semester at Carnegie Mellon University. The research was divided into two sections; each of which are explained by a separate section of the paper. The first section focuses on using reinforcement techniques for predicting interactions between cyclists and pedestrians for autonomous vehicle path planning. The second section delineates the progress made with regards to data collection and management.

1. Navigating intersections through reinforcement learning

1.1. Introduction

Twenty percent of all accidents occur at intersections for human drivers. Intersections have been selected as a problematic zone for autonomous vehicle navigation. Particularly, interactions with pedestrians and cyclists have been sparsely documented. Primarily, there are three different ways of handling behaviour at intersections: online learning, offline learning and imitation learning.

Imitation learning techniques [1] such as conditional imitation learning or guided cost learning might work well when encountered scenarios that have already been taught through past trajectories. However, agents are unable to handle exceptions which is the case at most intersections. Online learning requires the generation of an accurate generative model. Learning generative models is a very hard task and existing methods often fail when it comes to visual quality, temporal coherence and bridging the gap between real world data complexity and synthetic data sets [6].

This report therefore focuses on an offline learning method - reinforcement learning for navigation. As Varaiya suggested [7], planning for intelligent driving is divided into four hierarchical classes: route planning, path planning, manoeuvring planning and trajectory planning. Route planning is out of the scope of this report. Instead it focuses on the latter three.

The research demonstrates results from two different training algorithms: deep Q learning and maximum entropy inverse reinforcement learning. The first one works well in small state spaces but takes an infinitely large amount of time in real world settings and can scarcely account for limited perception of the ego agent. The second one is a novel idea that is sometimes overtly cautious and computationally more expensive. But it relates better to scenarios at intersections and is ultimately more feasible.

1.2. Literature Review

David Isele et.al [4] deployed Deep Q-Networks for the specific problem of handling intersections. It surpassed a number of heuristic approaches when it came to identifying occlusion scenarios and was better able to generalize to novel domains. However, it does occasionally result in collisions which are very expensive in the real world.

In his paper in 2016, D. Zhao et al. [8] introduced an on-policy reinforcement learning method. SARSA learning combined with deep learning to solve more complicated control problems, and as a result it outperforms deep Q-learning methods in convergence rate. The key algorithm is for every update process, SARSA learning takes the next action a' for updating the current state-action value, and the quintuple (s, a, r, s', a') will be derived in sequence. I have based my approach on the latter method when implementing the deep SARSA agent.

B. Ziebart et al. [9] proposed Maximum Entropy Inverse Reinforcement Learning that resolves ambiguities by choosing distributions that have no additional preferences beyond matching feature expectations. In locally normalizing probabilistic IRL models, bias exists that paths with more branches will be assigned lower probability mass. However, maximum entropy IRL avoids this bias by assigning probability according to expected reward, higher probability for higher reward behavior. This model has been applied to model the real-world navigation and driving behaviors.

In 2010 Henry, Peter, et al. [3] use inverse reinforcement

learning (IRL) to enable robots to navigate through crowded environments with dynamics and partially observed features. The results is an imitated human pedestrian behavior given an environment. It is an extension of maximum entropy inverse reinforcement learning with more difficult scenarios. They use a crowd motion simulator that generate realistic motion patterns to evaluate their work. The two approaches mentioned above are used in combination with mean models of features generated through Gaussian regression to enable better performance of the maximum entropy agent.

1.3. Dataset deployed

The dataset in question is the Stanford drone dataset [5]. It consists of eight unique scenes and six different types of agents. The number of videos in each scene and the percentage of each agent in each scene is reported in 1. An excellent feature of the dataset is that it has a large percentage of pedestrians and cyclists.

This project has focused its learning algorithms on the roundabout instance called 'deathCircle' on the Stanford campus. The ground truth of the dataset is shown in 2. The 'deathCircle' spans over 200m in both directions in a horizontal plane. Training data includes 1307 cyclists, 831 pedestrians and 109 cars from 5 videos. Other agents are ignored.

Scenes	Videos	Bicyclist	Pedestrian	Skateboarder	Cart	Car	Bus
gates	9	51.94	43.36	2.55	0.29	1.08	0.78
little	4	56.04	42.46	0.67	0	0.17	0.67
nexus	12	4.22	64.02	0.60	0.40	29.51	1.25
coupa	4	18.89	80.61	0.17	0.17	0.17	0
bookstore	7	32.89	63.94	1.63	0.34	0.83	0.37
deathCircle	5	56.30	33.13	2.33	3.10	4.71	0.42
quad	4	12.50	87.50	0	0	0	0
hyang	15	27.68	70.01	1.29	0.43	0.50	0.09

Figure 1: Stanford drone dataset statistics

1.4. Deep Q Learning agent (DQN)agent

1.4.1 Designing the basic algorithm

The neural network in question has 3 layers. It is designed to input an array of 4 values containing the state of the system. The state of the system is basically the feed received from sensors at 4 points of the agent i.e. at the front, rear, left and right. The neural network has 2 fully connected hidden layers each consisting of 24 nodes with activation of the form relu. The output layer has 9 nodes each representing a different action which are all defined within the reward function. The action space includes 9 actions: forward, back, left, right, forward and left, forward and right, back and left, back and right. The training process involves feeding input and output pairs into the neural network in order to understand and predict rewards based on the data received from the sensors. After the training process is completed, the model is able to effectively predict the reward of the current state. The algorithm was executed for 300 episodes and the gist is delineated in figures 3 and 4.

1.4.2 Computing the loss function for the network

The loss function is of the mean square error form and Adam optimizer is utilized. Consider the agent in a given state s. It can perform action a, receive reward r and move to the next state s. The value function as a result of performing a given action is denoted by $Q(s, a)$. The Q-value of all possible actions for a given state are considered and the maximum is selected. It is then multiplied by a discount factor γ and added to the reward in the current state to get the target value. The loss function as shown in equation 1 is computed as the difference between the target value and the Q-value predicted by the model.

$$Loss = (r + \gamma Q(s, a') - Q(s, a))^2 \tag{1}$$

1.4.3 Computing the reward function and deciding an action

A parameter called exploration rate is defined. In the initial stages of training, when the value of the exploration rate is high, the agent in a given state selects actions randomly based on this parameter. Once it learns through several interactions with the environment, it picks the action with the largest Q-value. The exploration rate is updated after every 50 episodes and becomes 99.5% of the previously exploited exploration rate value. This way, the agent gets to explore different sets of states and rewards and hence learn the surrounding environment in a better and more complete way. The reward is basically is a function of the aggressiveness or passiveness of approach of the agents in the vicinity. In other words, $\Delta d/\Delta v$ is taken into consideration. The reward is delineated in 2. Success is defined if the goal or target position is attained is attained. Collision is defined if the agent is within 0.5m of another agent.

$$r = \begin{cases} 100, & \text{if } success \\ -100 & \text{if } collision \\ \sum_{i=1,..4} 5\frac{\Delta d_i}{\Delta v_i} & \text{otherwise} \end{cases} \tag{2}$$

1.4.4 Introducing experience replay

Neural networks do not have a memory and they tend to replace older experiences with newer ones. Every action

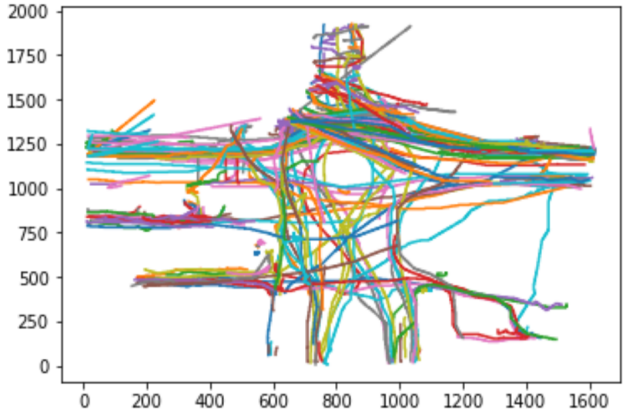


Figure 2: Ground truth over death circle

Algorithm 1: Deep Q-learning with Experience Sampling
Initialize replay memory D to capacity N
Initialize action-value function Q with random weights
for episode = 1, **do**
Initialize state s
  for step = 1, **do**
  with probability = Ɛ
    select a random action a
  otherwise
    select $a = max_a$ Q(s, a′)
  Execute action a, observe reward r and state s'
  Set s′= s
  Store transition s, a, r, s' in D
  Sample random minibatch of transitions $s, a, r, s'$from D
  Compute reward from equation 2
  Compute loss from equation, perform a gradient descent step
  end for
end for

Figure 3: Learning algorithm for deep Q learning with experience replay

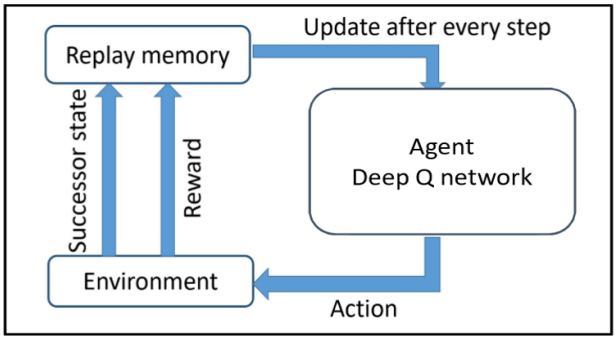


Figure 4: Gist of Deep Q learning

affects the next state. This outputs a sequence of experience tuples which can be highly correlated. If the network is trained in sequential order, the agent is at risk of being influenced by the effect of this correlation. This does not favor optimum performance during training. This algorithm allocates a certain amount of memory to store experiences and observations of the past. At specific instances during training of the model, a batch of experiences is sampled and fed into the neural network. This ensures greater efficacy of the agent in the long run. Correlation is broken and the action values do not oscillate or diverge catastrophically. In training simulations, the batch size was 64.

1.5. Maximum entropy inverse reinforcement learning agent

1.5.1 Highlights of Original framework

The original maximum entropy framework assumes trajectories in the existing data set are reasonable and uses them as input demonstrations in order to teach an ego agent ideal behaviour maneuvers. It generates a mean reward at the end of training and a policy is calculated on the basis of this reward. Another assumption of the conventional framework is that it assumes that the ego agent has complete knowledge of the features of the entire environment. Also, the features in the state space are constant for a given trajectory and across different trajectories.

This model implies that a path or trajectory τ through states s_i and actions a_{ij} has a cost which is a linear combi-

nation of features parameterized by θ as shown:

$$cost(\tau) = \sum_{a_{i,j} \in \tau} \theta \cdot f_{a_{i,j}} \tag{3}$$

For a fixed start and end point, this results in a maximum entropy distribution over paths shown in equation 4.

$$P(\tau|\theta) = \frac{1}{Z(\theta)} e^{\sum_{a_{ij} \in \tau} -\theta \cdot f_{a_{i,j}}} \tag{4}$$

1.5.2 Modified Learning algorithm

This scenario differs from the original algorithm because the features are not constant. They vary within a path and also between paths. Secondly, the ego agent has limited perceptual range of his surrounding. He has knowledge of the states around him to the extent allowed by his sensory system in the case of a pedestrian or sensors as in the case of a car.

In this case we update the features in the ego agent's perceptive field and perform gradient descent for every time step as shown in equation 5. The other features beyond the scope of the agent's detection are set to mean model values computed in section 1.5.4.

$$\nabla F = \mathbf{f}^t - \sum_{a_{ij} \in H} D_j^t f_{a_{ij}}^t \tag{5}$$

Where D_j is the expected state visitation frequency of all actions entering the state during the given time step. Calculating state visitation frequency using the method in [3] is computationally expensive. It has been modified such that a matrix is formed using values from the density mean model. This matrix is updated with values using equation6 every time a new state is encountered. It is a function of the policy Q_{ij} computed through regular value iteration and the transition probability p_{ij}. It is conditioned on weight vector θ as follows:

$$D_j^t = \sum_{a_{ij} \in H} D_i^t Q_{ij}^t p_{ij} \tag{6}$$

The weights are updated as follows where γ is the discount factor:

$$\theta_{n+1} = \theta_n e^{-\gamma \nabla F} \tag{7}$$

A reward is computed for every time increment H and a policy is calculated on the basis of this reward. The ego agent is made to follow this policy. The algorithm is delineated in figure 5

For a given trajectory, the maximum entropy distribution is as follows where T is the duration for which the trajectory lasts:

$$P(\tau|\theta) = \frac{1}{Z(\theta)} e^{\sum_T \sum_{0<h<H} -\theta \cdot f_{a_{i,j}}} \tag{8}$$

1.5.3 Feature Estimation

The environment which spans over 200m is represented by a grid state space with m states in total. Actions describe transitions between these states. Features are basically functions of all actions entering the given state. In other words, the feature vector for a given state is representative of the neighboring states.

It is assumed that all maneuvers of a virtual agent are planned on the basis of the number of agents in the vicinity as well as the passiveness or aggressiveness of their approach. In such a scenario, time to collision or the ratio of relative distance to relative velocity would have been considered an ideal feature metric. However, in order to learn better rewards, this approach keeps distance static and considers relative velocity and density of neighboring states.

Consider n possible actions into a given state or in other words, n neighboring states. The dynamic features for each state are divided into $3n + 1$ bins. The real valued density is designated n bins and is a whole number representing the number of people in each neighboring state. In addition, the velocity component in the x and y direction are designated n bins each. Relative velocity is computed for each of the neighboring states. The velocity features also successfully capture orientation in this manner. Finally, a single bin has been introduced at the end which is a function of distance from the target position of the virtual agent. Intuitively speaking, a smaller target distance means a smaller weight learned and equivalently a smaller reward for the given state.

1.5.4 Gaussian representation of mean model features

The original feature matrix is constructed based on values from mean density and velocity models. As mentioned earlier, the state space is two-dimensional grid across which the virtual agent travels. The underlying assumption is that the Stanford drone data-set is a record of trajectories that are considered ideal. In other words, zero collisions occurred during the recorded duration. For each time instant in a recorded trajectory, there exists a position (x, y) with a corresponding density and velocity component.

To formulate a density model, this report looks to Gaussian mixture models in combination with the expectation maximization technique. Several gaussian kernels are fitted across positions in the state space as a means of identifying and distinguishing dense and sparse regions. The mean and covariance matrix of each mixture model are arbitrarily initialized, and the log likelihood is computed. Posterior probabilities are calculated, and initial parameters and corresponding log likelihood are re-estimated. The steps are repeated until convergence occurs. The number of Gaussian components allowed to exist is evaluated by minimizing

Input: Ground truth of trajectories from an 84 min video at the death circle intersection
Start point and goal of virtual agent
Compute mean models of density, velocity in x and velocity in y positions at the intersection
Compute initial feature matrix based on mean models - f
While target not reached **do**:
For time = 1, ... t - H, t **do**:
Initialize weight vector θ
While not converged **do**:
Update estimates of locally observable features in original feature matrix - **f**
For actions $a \in t + H$, compute expected state visitation frequency - D_a
Compute the gradient $\nabla F^t = \mathbf{f}^t - \sum D_a f^t$
Update the weights
end while
Compute reward $r = \mathbf{f} \cdot \theta$
Compute policy for virtual agent, perform action with maximum reward
end for
end while

Figure 5: Max entropy inverse reinforcement learning

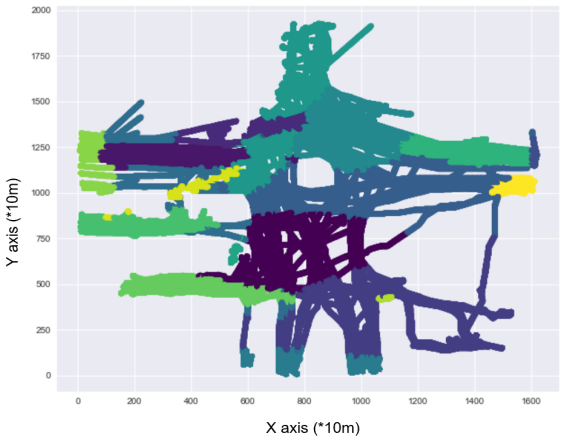


Figure 6: Density model for identifying dense and sparse regions across the state space

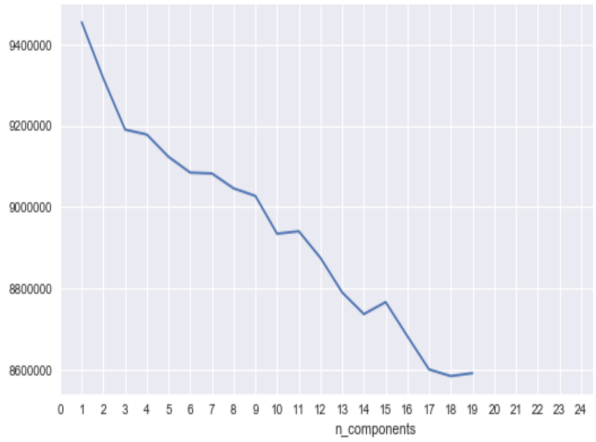


Figure 7: Akaike Information criterion for cluster number selection

information lost per model through the Akaike Information Criteria. This is shown in figure 6 To compute mean density at a particular point (x_1, y_1), the value at (x_1, y_1) in the density model above is divided by the visitation frequency of that state. This value is rounded off to the nearest whole number to get mean density.

To formulate mean velocity models as shown in figures 8 and 9 , velocity from the pre-recorded data at several positions is divided into its x and y components. Separate Gaussian kernels representing mean velocity in x and y directions are learnt. The kernels are a combination of inverse radial bias functions that represent medium irregularities and white noise. The hyperparameters are tuned to maximize negative log likelihood until convergence occurs. Given a set of observations at a given instance, this approach subtracts observed features from mean model features to get a set of deviations from the mean. A new Gaussian kernel is constructed that is a smooth representation of these deviations. The deviated values are then added to the original feature model which represents the updated feature matrix . This method gives extra weightage to the features within the observable range of the agent at a given time instant.

1.6. Experiments

1.6.1 Comparison between the two agents

Deep Q learning (DQN) and max entropy inverse reinforcement learning (MAX-ENT) have been used to train a virtual agent in a simplified environment setting. In this

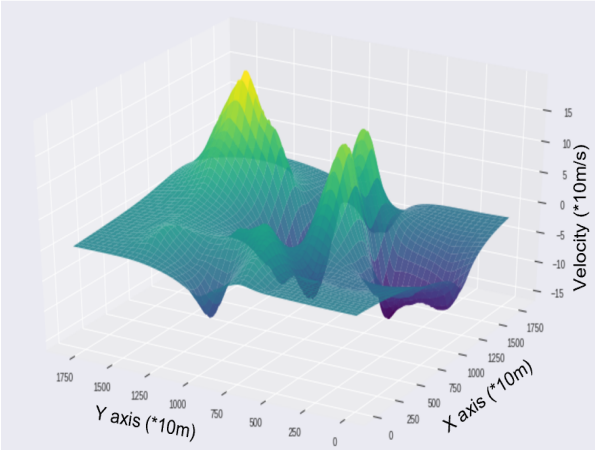


Figure 8: Velocity mean model in X direction across state space: X and Y axis (metres)*10, Z axis is mean velocity (metres*10/sec)

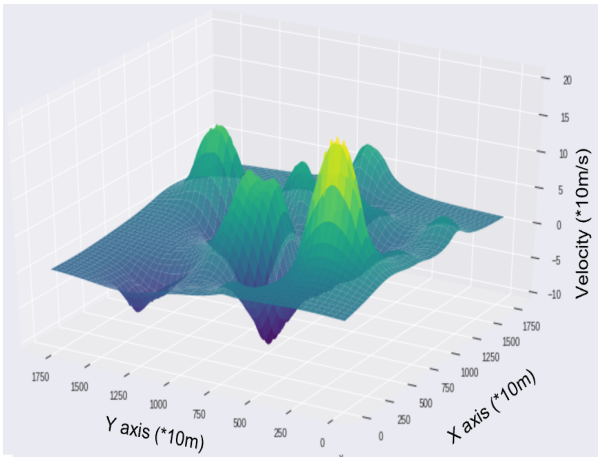


Figure 9: Velocity mean model in Y direction across state space: X and Y axis (metres*10), Z axis is mean velocity (metres*10/sec)

case, the virtual agent is a pedestrian but the approach can be extended to cars and cyclists as well. The other agents in play in the environment are extracted from the ground truth of the Stanford drone dataset. The start and end positions of both virtual agents are fixed. A comparison is performed between the two agents. It is imperative to note that the environment is its most primitive for the experiment at hand.

For the sake of convenient comparison, the state space has been discretized into an 8 by 8 matrix. The action space includes only 5 actions: forward, back, left, right and stay. With regards to the training time, the DQN agent took a large amount of time to train before demonstrating a near optimum path. This means that a DQN agent must be taught its behavior through extensive simulation before being allowed to navigate on the road in real time. Even so, it is unlikely to have encountered every scenario during training and is destined to collide and fail in case of anomalies.

Reward generation is very different for both agents and is shown in figures 10 and fig:rewardIRL. Trajectory followed by each agent is also different and is shown in figures 12 and 13

1.6.2 Fine tuning the maximum entropy agent

The star represents the ego agent in . The simulation environment is a replica of the death Circle intersection in Stanford. The other agents in play denoted by colored dots are from a 22 minute video recorded at the intersection. The action space has 9 actions: stay, forward, back, left, right, forward and right, forward and left, back and right, back and left. If the agent chooses to leave the current state, it will move 1.5m every 1.2 seconds. The virtual agent is made to navigate the environment. The trajectory taken by it is shown in blue in figure 14 It avoids all collisions. It learns optimum rewards every time instant. However, it faces problems on occasion when translating the learned rewards to a successful policy and may not always focus on reaching the target position.

1.7. Conclusion

The research successfully compared direct and inverse reinforcement learning techniques. It demonstrated the efficacy of the latter over the former for successful navigation at traffic intersections in real time with dynamic features. The former technique requires extensive training which is extremely time consuming. It is also not equipped to handle any deviations from the behavior it was trained on. Reinforcement learning through deep Q learning is therefore deemed unfit for navigating intersections.

The latter technique deploying maximum entropy inverse reinforcement learning eliminates the need for training and is equipped to handle anomalies in the behavior of other road users. It can also be deployed at every time instant to calculate policies in real time. This approach has also deemed the original method of computing expected state visitation frequency computationally inefficient for large state spaces and has replaced it with a much faster approach.

1.8. Future work

The technique in this report can be extended successfully to any type of virtual agent from a cyclist to a car. However, it is pertinent to note that the mean models used for constructing the feature matrix may not always best represent the agents that were in minority during the recorded trajectories. It is not very successful in reaching the target position. Future work will delve into representing the

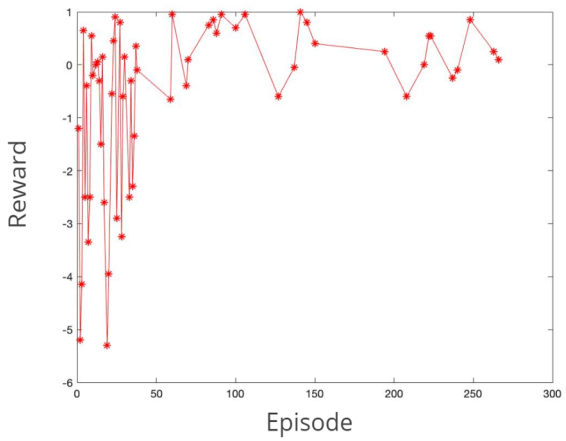


Figure 10: Reward of the DQN Agent over time during training

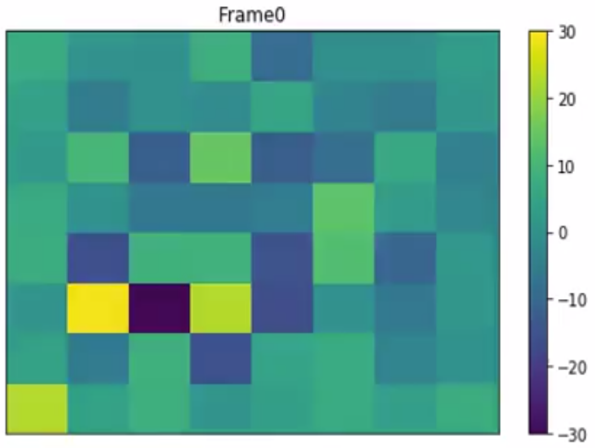


Figure 11: Reward of the MAX-ENT Agent for the first frame across the state space

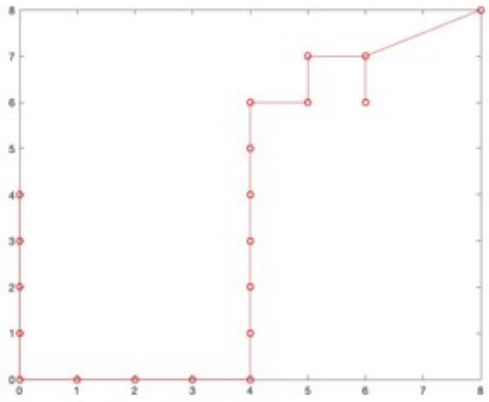


Figure 12: Trajectory of the DQN Agent

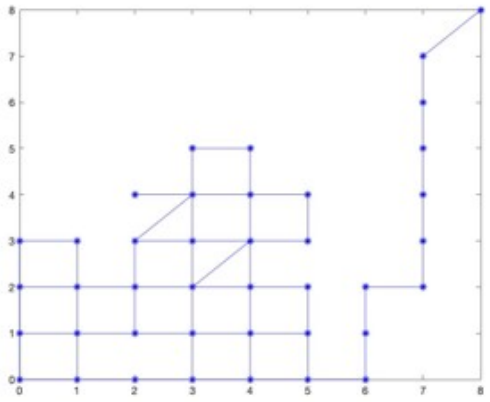


Figure 13: Trajectory of the MAX-ENT Agent

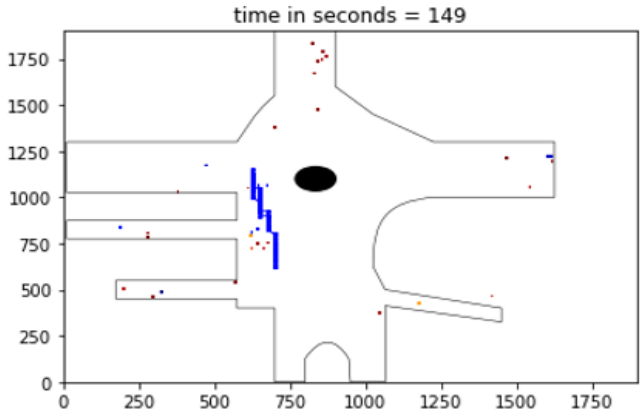


Figure 14: Trajectory taken by IRL agent after fine tuning

target position better in the feature matrix so that the learning algorithm gives it greater importance whilst computing weights. Minor changes can also be included to account for velocity in the given action space.

2. Progress in Database management

This section provides a brief and concise summary of my work in mySQL this semester. After successfully completing a course in Udemy, I proceeded to install darknet on the main computer at the instruction of Jia Cheng. I also downloaded the KITTI dataset and uploaded it to the mySQL Workbench. Modifications had to be made to existing python code for conversion off data into acceptable SQL format. This code was KITTI dataset centric. I also generalized the python program so that it could deal with other datasets. Currently I am in possession of data from the KITTI dataset , Safety Pilot Model Deployment Database

and the Stanford Drone Dataset.

3. Work in Progress and Future directions

This section highlights future work which includes implementing Bayesian recurrrent neural networks to generate a set of control points of distance and velocity for pedestrians and cyclists. It also involves the implementation of a cyclist motion model that identifies trajectory patterns and discrete transitions at several intersections using Dirichlet Process Gaussian Proceeses specified by Yutaon Han [2] in his 2019 paper.

References

[1] F. Codevilla, M. Miiller, A. López, V. Koltun, and A. Dosovitskiy. End-to-end driving via conditional imitation learning. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1–9. IEEE, 2018.

[2] Y. Han, R. Tse, and M. Campbell. Pedestrian motion model using non-parametric trajectory clustering and discrete transition points. *IEEE Robotics and Automation Letters*, 2019.

[3] P. Henry, C. Vollmer, B. Ferris, and D. Fox. Learning to navigate through crowded environments. In *2010 IEEE International Conference on Robotics and Automation*, pages 981–986. IEEE, 2010.

[4] D. Isele, R. Rahimi, A. Cosgun, K. Subramanian, and K. Fujimura. Navigating occluded intersections with autonomous vehicles using deep reinforcement learning. *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2034–2039, 2018.

[5] A. Robicquet, A. Sadeghian, A. Alahi, and S. Savarese. Learning social etiquette: Human trajectory understanding in crowded scenes. In *European conference on computer vision*, pages 549–565. Springer, 2016.

[6] T. Unterthiner, S. van Steenkiste, K. Kurach, R. Marinier, M. Michalski, and S. Gelly. Towards accurate generative models of video: A new metric & challenges. *arXiv preprint arXiv:1812.01717*, 2018.

[7] P. Varaiya. Smart cars on smart roads: problems of control. *IEEE Transactions on automatic control*, 38(2):195–207, 1993.

[8] D. Zhao, H. Wang, K. Shao, and Y. Zhu. Deep reinforcement learning with experience replay based on sarsa. In *2016 IEEE Symposium Series on Computational Intelligence (SSCI)*, pages 1–6. IEEE, 2016.

[9] B. D. Ziebart, A. L. Maas, J. A. Bagnell, and A. K. Dey. Maximum entropy inverse reinforcement learning. In *Aaai*, volume 8, pages 1433–1438. Chicago, IL, USA, 2008.