Learning-based Vehicle Controller for Vehicle-Pedestrian Interactions

Shubham Kedia Ye-Ji Mun Kristina Miller

***Abstract*—Autonomous driving amongst pedestrians is a challenging problem. The self-driving vehicle must reach the goal efficiently while navigating through the pedestrians to avoid collision. Recent works have implemented deep reinforcement learning methods to solve this navigation problem. However, they often neglect the multi-modality of pedestrian dynamics or provide no analysis on the learned strategies for divergent pedestrian behaviors. In this work, we study the impact of various pedestrian behaviors on the effectiveness of a learned-based vehicle controller at a crosswalk scenario. We generate multiple pedestrian models with different crossing characteristics. Then, for each pedestrian type, we learn a policy using a model-free reinforcement learning algorithm. We compare the effectiveness of the policies and analyze how the crossing types influence the vehicle's collision avoidance strategy. Our empirical result shows that the policy learned with an aggressive and adversarial pedestrian performs much better across all types of pedestrians compared to the one trained with safe pedestrian that frequently yields to the vehicle.**

I. Introduction

Vehicle navigation through crowded environments is a difficult problem. There are many aspects to this problem, such as perception, motion planning, control, and intent estimation. These are all rich research topics, with many papers focusing on just one component. In this work, we focus on the control part of the problem.

There are many works that focus on this part of the problem. Some examples include formal controller synthesis, model predictive control, and reinforcement learning based controllers.

Formal controller synthesis: Formal controller synthesis aims to create controllers that are correct-by-construction. There are many works that approach the vehicle navigation problem using controller synthesis, such as FACTEST [1], FasTrack [2], SCOTS [3], LTLMoP [4], and funnel libraries [5]. However, these approaches can become difficult when the vehicle model is complicated or the pedestrian behaviour is unknown.

Model predictive control: Model predictive control (MPC) is a common approach used for vehicle navigation. In MPC, an optimization problem is solved to try and find the control inputs to a vehicle. There are many different ways to cast the optimization problem, such as linear programming [6], quadratic programming [7], and mixed integer linear programming [8]. Again, this approach can quickly become difficult when the vehicle model is complicated or the pedestrian behaviour is unknown.

Reinforcement learning based controllers: In the last few years, reinforcement learning (RL) has been dominantly explored in urban autonomous driving. For autonomous driving amongst pedestrians, several studies attempts to model the pedestrian behaviors conditioned on a hidden variable like the pedestrian's goal position or the distance from the vehicle [9], [10]. However, these approaches neglect the multi-modality of the pedestrian behaviors. On the other and, the model-free methods can automatically generate effective policy on new and unseen states from explicit trial-and-error [11].

In this work, we address the control aspect by building a model-free reinforcement learning controller and studying the effects of different pedestrian behavior on the success rate of the learned controller.

II. Problem Formulation

Preliminaries: Let $\mathbb{R}^n$ denote the set of all real n-dimensional vectors. Given a vector $x \in \mathbb{R}^m$, $m \geq n$, the projection of x onto $\mathbb{R}^n$ is denoted by $x \downarrow \mathbb{R}^n$.

Given a point $p \in \mathbb{R}^2$, let $\mathcal{V}_{l,w}(q)$ denote the physical space that a vehicle of length l and width w with state at q occupies.

A. Vehicle-pedestrian scenarios

A vehicle-pedestrian scenario occurs when a vehicle and must navigate from some initial state to a goal state while avoiding some pedestrians in a 2- or 3-dimensional workspace $\mathcal{W}$. This workspaces is the physical space that the vehicle and pedestrians occupy. Figure 1 shows a 2-dimensional vehicle-pedestrian scenario at a crosswalk.

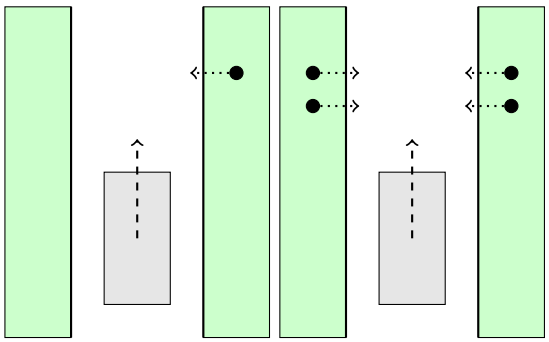

Fig. 1: Vehicle-pedestrian scenario with a vehicle depicted as a gray rectangle and pedestrians are shown as black dots. The desired direction of travel is shown with the arrows.

The vehicle and pedestrians only know the physical location of each other at any point in time. To make the problem

simpler, we assume that the pedestrian state is fully observable by the vehicle and disregard the sensor measurement errors. Figure 2 shows how the vehicles and pedestrians interact with each other.

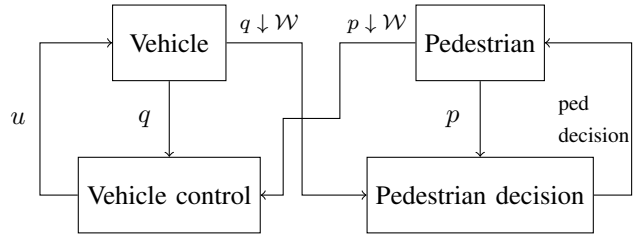

Fig. 2: The vehicle shares its physical location with the pedestrian, and vice versa. For the vehicle: Given the current state q, and the physical location of the pedestrian $p \downarrow \mathcal{W}$, a control input u is computed. For the pedestrian: Given the current pedestrian state p, and the physical location of the vehicle $q \downarrow \mathcal{W}$, the pedestrian makes a decision on what it will do next.

B. Vehicle dynamics

The vehicle is a discrete-time dynamical system of the following form.

$$q[t+1] = f_{veh}(q[t], u[t]) \tag{1}$$

Here, (i) $q[t] \in \mathcal{Q} \subseteq \mathbb{R}^n$ is the vehicle state at time $t \geq 0$, and (ii) $u[t] \in \mathcal{U} \subseteq \mathbb{R}^m$ is the vehicle input at time $t \geq 0$. Given an initial state $q[0] \in \mathcal{Q}$ and a sequence of T inputs $u = \{u[0], \ldots, u[T]\}$, a *trajectory* $\xi[t]$ of the vehicle satisfies (i) $\xi[0] = q[0]$, and (ii) $\xi[t] = f_{veh}(\xi[t], u[t])$ for all $t \in [0, T]$.

Example 1. The the vehicle in our running example is described as follows.

The vehicle state is given by $q = [x_{veh}, y_{veh}, v_y]$ where (x_{veh}, y_{veh}) is the physical location of the vehicle in $\mathcal{W} \subseteq \mathbb{R}^2$, and v_y is the velocity of the vehicle in the y_{veh} direction.

The input space for the vehicle is given by $\mathcal{U} = \{-2, -1, 0, +1, +2\}$. Here, the input is the acceleration in the y_{veh}-direction.

The vehicle dynamics are given by the following:

$$\begin{pmatrix} x_{veh}[t+1] \\ y_{veh}[t+1] \\ v_y[t+1] \end{pmatrix} = \begin{pmatrix} x_{veh}[t] \\ y_{veh}[t] \\ v_y[t] \end{pmatrix} + \begin{pmatrix} 0 \\ v_y[t] \\ u[t] \end{pmatrix} \Delta t \tag{2}$$

C. Pedestrian dynamics

The environment includes N pedestrians. The pedestrian motion model are adopted from [12]. Each pedestrian has dynamics of the following form.

$$p_i[t+1] = f_{ped}(p_i[t], g_i, z_i) \tag{3}$$

Here, (i) $p_i[t] \in \mathcal{P}_i \subseteq \mathbb{R}^s$ is the state of the pedestrian at time $t \geq 0$, (ii) g_i is the goal state of the pedestrian, (iii) $z_i \in \mathcal{Z}_i$ is the type of pedestrian. This pedestrian type will determine how the pedestrian behaves around vehicles.

Example 2. The pedestrians in our running example is described as follows.

Each pedestrian's state is $p_i = [x_{ped}, y_{ped}, v_{ped,x}, v_{ped,y}]$, where $p_i \downarrow \mathcal{W} = (x_{ped}, y_{ped})$ is the physical location of the pedestrian in $\mathcal{W} \subseteq \mathbb{R}^2$, and $v_{ped} = (v_{ped,x}, v_{ped,y})$ is the velocity vector of the pedestrian.

Each pedestrian's type is given by $z_i \in \mathcal{Z} = \{\textit{non-reactive}, \textit{safe}, \textit{aggressive}, \textit{normal}, \textit{adversarial}, \textit{genius}\}$.

A pedestrian with type $z_i = \textit{non-reactive}$ has dynamics of the following form:

$$p_i[t+1] = \begin{cases} p_i[t] & t+1 < t_0 \\ p_i[t] + \begin{pmatrix} v_{ped} \\ \mathbf{0} \end{pmatrix} \Delta t & t+1 \geq t_0 \end{cases} \tag{4}$$

Here, $v_{ped} = \frac{(g_i - p_i[0]) \downarrow \mathcal{W}}{\|(g_i - p_i[0]) \downarrow \mathcal{W}\|} \|v_{ped}\|$ where $\|v_{ped}\|$ is some constant velocity of the pedestrian, and $t_0 \geq 0$ is some random time step where the pedestrians starts moving.

A pedestrian with type $z_i \neq \textit{non-reactive}$ has dynamics which are [13] of the following form:

$$p_i[t+1] = p_i[t] + \begin{pmatrix} \mathbf{0} \\ F_{des}[t] + F_{veh}[t] \end{pmatrix} \tag{5}$$

Here, $F_{des}[t]$ is the force of the destination, and $F_{veh}[t]$ is the force of the vehicle.

$$\begin{aligned} F_{des} &= k_{des} \left(v_{ped} - \|v_0\| \frac{(g_i - p_i[t]) \downarrow \mathcal{W}}{\sqrt{\|(g_i - p_i[t]) \downarrow \mathcal{W}\|^2 + \sigma^2}} \right) \\ F_{veh} &= A e^{(-b\|(\xi[t] \downarrow \mathcal{W}) - (p_i[t] \downarrow \mathcal{W})\|)} ((\xi[t] \downarrow \mathcal{W}) - (p_i[t] \downarrow \mathcal{W})) \end{aligned} \tag{6}$$

Here, $\xi[t]$ is the state of the vehicle at time t, v_0 is the initial velocity of the pedestrian, and A, b, σ, and k_{des} are parameters defining pedestrian behavior that are determined by the pedestrian type. All different pedestrian types are shown in Table I.

TABLE I: Pedestrian types

Ped type	A	b	k_{des}	σ	**Behavior**
non-reactive	n/a	n/a	n/a	n/a	Vehicle has no effect on behavior
safe	80	0.4	0.7	10	Mostly cares about avoiding the vehicle
aggressive	50	1.8	1.1	10	Mostly cares about reaching the destination
normal	150	0.7	1	10	Cares equally about reaching the destination and avoiding the vehicle
genius	180	0.3	1.4	10	Similar to *normal*, but can make "better" decisions about reaching the destination or avoiding the vehicle
adversarial	50	1.8	1.1	10	Similar to *aggressive*, but starts with a higher velocity

D. Problem statement

Given a vehicle with initial state $q[0]$ and n pedestrians with inital states $p_i[0]$, and goal states g_i, $i \in \{1, \dots, n\}$ find a sequence of inputs $u[t]$ for the vehicle such that $p_i[t] \notin \mathcal{V}_{l,w}(\xi[t])$ for all $t \in [0, T]$ and $i \in \{1, \dots, n\}$.

Example 3. The scenario in our case study is seen in Figure 1, and is described as follows. The physical space occupied by the vehicle is $\mathcal{V}_{6,4}(q)$. The pedestrians are illustrated as dots, and we disregard their volumes for simplicity. The pedestrians start on one side of the street and aim to cross to the other side. We conduct experiments on both a single pedestrian environment and a multi-pedestrian environment. The multi-pedestrian (shown on the right in Figure 1.

In our scenario, all units are meters (unless otherwise stated). The initial state of the vehicle $q[0]$ is chosen such that $(x_{veh}[0], y_{veh}[0]) \in \{(12, 8.9), (12, 15.9)\}$, and $v_y[0] \in [1, 2]$ m/s. It drives in a straight line longitudinal path towards the goal position at $(12, 33)$.

In a single pedestrian scenario, the pedestrian moves along a latitudinal path from $(6.5, 30)$ to $(16, 30)$. In the multi-pedestrian setting, the pedestrians have initial and goal positions chosen from the set $\{(6.5, 29), (6.5, 30), (16, 29), (16, 30)\}$, such that each pedestrian has a different initial and goal position, and for any pedestrian the initial position is not equal to the goal position.

E. Baseline solution (Heuristic controller)

A heuristic controller is used as the baseline solution to this problem. Here, there are three parts of the pedestrian crossing scenario. When the pedestrian is either in the starting area or in the road, and the pedestrian is in front of the vehicle, then the vehicle should come to a stop. When the pedestrian reaches a safe goal area G were $g_i \in G$, the vehicle can continue. An example of this can be seen in Figure 3.

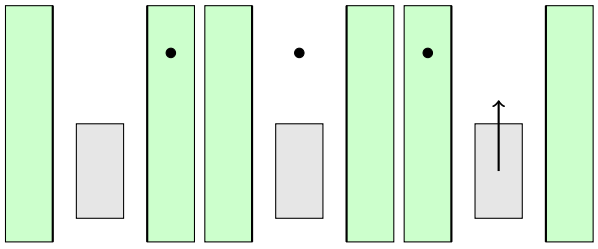

Fig. 3: Example pedestrian crossing scenario. The vehicle is shown using a gray box, and the pedestrian is shown as a black dot. The pedestrian waiting area is the bottom green box and the pedestrian goal area is the top green box. When the pedestrian is ahead of the vehicle and in the waiting area or the road, then the vehicle should come to a stop. When the pedestrian reaches the goal area, then the vehicle should start to move again.

The controller used to control the vehicle velocity is a bang-bang controller. The heuristic controller algorithm is shown in Algorithm 1.

Algorithm 1: Heuristic controller

Input: current vehicle state $q = [x_{veh}, y_{veh}, v_y]$, pedestrian state $p = [x_{ped}, y_{ped}, v_{ped,x}, v_{ped,y}]$, vehicle goal state $d = (d_x, d_y)$, constant velocity $\bar{v}$, pedestrian goal area $G \subset \mathcal{W}$

Output: $u = a_y$

$\hat{n}_{v2d} \leftarrow \frac{(d - q\downarrow\mathcal{W})}{\|(d - q\downarrow\mathcal{W})\|}$

$\hat{n}_{v2p} \leftarrow \frac{(p\downarrow\mathcal{W} - q\downarrow\mathcal{W})}{\|(p\downarrow\mathcal{W} - q\downarrow\mathcal{W})\|}$

if $\hat{n}_{v2d} \cdot \hat{n}_{v2p} > 0 \ \wedge \ p\downarrow\mathcal{W} \in G$ **then**

  $v_{ref} \leftarrow 0$

else

  $v_{ref} \leftarrow \bar{v}$

end

$u \leftarrow 2\text{sgn}(v_{ref} - v_y)$

This heuristic controller has some flaws. For example, the vehicle does not know when the pedestrian starts to move towards the goal, so it will wait until the pedestrian reaches the goal. However, there may be time for the vehicle to move to its goal position while it is waiting. Therefore, it is beneficial to learn how to navigate around pedestrians so the vehicle can reach the goal quickly.

III. Learning-based Controller framework

We use model-based deep RL approach to study how the vehicle's learned policy is influenced by different pedestrian types, illustrated in Table I. The pedestrian type is determined at the beginning of each episode and does not change throughout the episode. For the multi-pedestrian setting, the characteristic of each pedestrian is randomly selected from safe, aggressive, and normal type. This section outlines the network structure and the implementation details of our learning-based controller. We also assume that the control inputs have an immediate effect on the state of the system. Based on these assumptions, this section outlines the network structure and the implementation details of our learning-based controller.

A. Technical Background

a) Reinforcement Learning: In a RL framework, at time t an agent observes the state of the environment s_t and executes a policy $\pi(a_t|s_t)$ to take the action a_t in the current state. Then, the agent is given a reward $r_t = R(s_t, a_t)$ as a consequence of its action. Using these immediate rewards, the goal of RL is to choose the sequence of actions that maximize the expected future reward, also known as the value V, as follows:

$$\pi^* = \arg\max_{\pi} V^{\pi}(s) = \mathbb{E}\left[\sum_{t=0}^{T} \gamma^{t-1} r_t | s_0 = s, \pi\right] \quad (7)$$

where $\gamma \in [0, 1)$ is a discounting factor, which is a weighting factor to scale down the rewards in the distant future.

b) Actor-Critic: An actor-critic algorithm consists of two neural networks: an actor and a critic [14]. The critic approximates the value function $V(s; w)$ with parameters w while the actor updates the policy parameter $\theta \sim \pi(a|s; \cdot)$ in the direction suggested by the critic. Given the approximated value by the critic network, the advantage of taking action a_t at state s_t is defined as

$$A(s_t, a_t) = r(s_t, a_t) + V(s_{t+1}; w) - V(s_t; w). \quad (8)$$

Then, the actor network is optimized by taking gradient steps with respect to the policy gradients and the advantage value:

$$\nabla J(\theta) \approx \sum_{t=0}^{T-1} \nabla_\theta \log_{\pi_\theta}(a_t, s_t) A(s_t, a_t). \quad (9)$$

As a result, the policy parameters are updated as

$$\theta \leftarrow \theta + \alpha \nabla J(\theta). \quad (10)$$

where α is a learning step size. Also, the critic parameters are updated as

$$w \leftarrow w + \beta \nabla A(s_t, a_t) \quad (11)$$

where α is another learning step size.

B. Reward Function Tuning

The aim of the vehicle is to learn a policy such that it avoids collision with the pedestrian and at the same time reach the goal efficiently. To this end, the reward is assigned for three terminal conditions: reaching goal position, collision and timeout. If the vehicle's state position ends up at the goal position, a positive reward is given. This award encourages the vehicle to move towards the goal throughout the episode. On the contrary, if the vehicle's action results in having the pedestrian within its rectangular region, a penalty is imposed for colliding. Lastly, the vehicle is punished for not reaching the goal position within the time limit $T = 50s$ in order to restrain the vehicle from unnecessary stop or slow movement leading to unlimited length of episode. Our reward is summarized as:

$$r_t = \begin{cases} 3 & \text{reaching goal} \\ -1 & \text{collision} \\ -1 & \text{timeout} \\ 0 & \text{else} \end{cases} \quad (12)$$

C. Network Architecture and Training

We adopt a model-free actor-critic method to learn the driving policy and the value function using the immediate rewards in Equation (12). Unlike the model-based approach, the model-free approach does not need to explicitly predict the pedestrian's next state nor estimate the pedestrian intended goal positions [11]. Instead, the vehicle policy can directly choose an action given the observable current state. The observable state at time t consists of the vehicle and pedestrian states: $s_t = [q[t], p_{rel}[t]]$ where $p_{rel}(t) = p[t] - q[t]$ is the pedestrian's relative position to the vehicle's. Note that the closest pedestrian state is taken into account in the multi-pedestrian setting. In our implementation, the actor and critic are three-layer multi-layer perceptrons [15] with the layer size $[2, 128, 32]$. Each layer is followed by ReLU activation function [16]. Both the actor critic takes the current observable state $s[t]$ as an input. Then, the actor has a additional softmax layer [17] to output action probabilities for the five discrete control input in $\mathcal{U}$, and the critic outputs the value of the corresponding state with the final linear layer. As our input state is relatively easy to embed, our simple network is enough to learn interactive vehicle controller for our simple scenario, which will be demonstrated in section IV.

To train the actor-critic network, we use Proximal Policy Optimization (PPO), a model-free policy gradient algorithm [18]. PPO is one of the policy gradient methods that employs multiple epochs of stochastic gradient descent for each policy update. To accelerate and stabilize training, we run twelve instances of the environment in parallel for collecting the agent's experiences. At each policy update, 30 steps of six episodes are used.

IV. Experiments and results

The scenario that we performed experiments on is the one described in Example 3. The pedestrians are initialized at some random velocity and allowed to move after some random time steps. The random initialization ensures that the agent has sufficient exploration in the state-space and the policy trained is robust enough for different initial states. Also, to understand the generalizability of the trained policies, evaluation is performed on all the types. To compare the performance of presented model-free RL based controller, identical experiments are performed in the baseline solution (heuristic controller). Finally, a multi-pedestrian environment is created to understand the scalability of the approach for multiple agents.

We created the environments using OpenAI Gym [19], and the training framework for the vehicle policy was implemented in PyTorch [20]. The policy is trained in a multi-environment setting, where eight parallel independent episodes are ran simultaneously to update the weights in critic network and actor network. The trained policy is then evaluated for 9216 randomly generated episodes to get a statistical convergence of the evaluation metrics. The following sub-sections provide a more detailed description of the training environments and evaluation results.

A. Heuristic controller

The results observed on the experiments with heuristic controller are summarized in Table II. Here, a success implies that the vehicle is able to avoid the pedestrians and reaches the desired goal. The failures are categorized into three groups: front collision, side collision, and timeouts. Front collision occurs when pedestrian comes in contact with front surface of the vehicle. This is the most undesirable type of failure, since it implies the policy does not employ corrective measures to avoid the collision. Side collision occurs when pedestrian comes in contact with side surface of the vehicle. Timeouts

are stalled progress, which is either due to vehicle moving backwards or oscillate front and back within the time limit.

The employed heuristic controller has shown satisfactory results for non-interactive pedestrian with success rate of 88%. The front collisions are 0% and there are no timeouts. On the contrary, the episode lengths are fairly long suggesting a conservative solution. Also, the results for reactive pedestrians are quite lacking, with aggressive pedestrians having success rate of only 30%. The presented heuristic controller is not specialized to deal with different reactive pedestrians. It may be possible to come up with a solution for a particular reactive pedestrian sub-class. But, it's difficult to generalize for different pedestrian classes. In this case, it worked well for the non-reactive class. Though, the observed results are for presented heuristic controller, there may be many successful solutions in academia and industry which are handcrafted to deal with vehicle pedestrian interactions. Currently, the study on the heuristic controllers is beyond the scope of this work. The presented heuristic solution is just to provide a baseline comparison for the developed learning-based controller.

TABLE II: Heuristic controller

Ped type	Success (%)	Front col (%)	Side col (%)	Episode length (s)
non-reactive	88.1	0	11.9	21.7
aggressive	31.8	34.5	33.6	42.2
safe	80.2	11.7	8.1	40.7
normal	49.8	31.7	18.4	40.7

B. Non-reactive pedestrian environment

The non-reactive pedestrian environment consists of the vehicle and a single pedestrian. The pedestrian is initialized with a constant velocity of 0.5 m/s and it starts moving on the crosswalk at a random time stamp of 0-5s. The initial velocity of the vehicle is also initialized randomly in the interval of 1-2 m/s to facilitate exploration. The objective of the pedestrian and autonomous vehicle is to reach their respective goals as defined in the problem statement. Table III summarizes the results of the evaluation for non-reactive pedestrian environment.

TABLE III: Non-reactive pedestrian evaluation summary

Motion constraint	Success (%)	Front col (%)	Side col (%)	Episode length (s)
None	80.4	18.51	1.1	14
Forward motion only	94.7	5.2	0.1	27

In this scenario, success rate observed is 80% while the timeout rate is 0.02%. Hence, to some extent the vehicle is able to avoid pedestrians and reach the goal. The percentage of front collision is higher than the rest of failure modes. So, the policy trained does not always perform effective braking based on the pedestrian dynamics. In fact, based on the observation of interactions between vehicle and pedestrians the trained policy is conservative. This is also evident from the high episode length of 14s. Ideally, based on the vehicle dynamics, its possible to reach the goal within 4s (if it accelerates to maximum from its starting position). Also, in many episodes, the vehicle was observed to move backwards and sometimes oscillate back and forth. This behaviour is not realistic in real life. To test the effect of this behaviour, another experiment was performed with velocity constrained to forward direction. The results are summarized in Table III.

Here, the success rate of 95% is observed but the trained policy is conservative. This is evident from the episode size of 27s (nearly doubled). Since, the vehicle cannot move backward, it will eventually reach the goal even it select actions with deceleration values more frequently. Also, the pedestrian is non-reactive and moves with constant velocity. A slow progress of the vehicle allows it to cross without any collisions. Note when the trained policy from previous unconstrained scenario was evaluated on this environment, the observed results are quite similar to the evaluation in unconstrained environment.

C. Reactive pedestrian environment

The reactive pedestrian environment has the similar setting as in the nonreactive environment. But the pedestrian dynamics are modeled as second order dynamical system which simulates different behavioural attributes. Since, the velocity is not fixed to a constant, a random initialization is done in range of 0-0.5 m/s. Also, the setting to start moving on crosswalk at random time stamp of 0-5s and random initialization of agent's velocity is kept the same. The multi-environment training and evaluation scheme are also retained from non-reactive pedestrian environment.

To understand the effect of pedestrian reactivity on trained policies, experiments are performed on three main reactive sub-classes: aggressive, safe, and normal. In each training environment, a pedestrian is initialized with a fixed behavioural attribute and the policy is trained for that behaviour setting. The trained policy is then evaluated for all three sub-classes to understand the generalizibility of the controller. Hence, the evaluation scheme has the following structure, evaluation metric (success rate, collision, etc.) of policy trained on environment (aggressive,safe, and normal) evaluated on all three reactive sub-classes. Table IV summarizes the results of the experiment. Note that timeouts are observed as zeros for all the evaluation environments and hence not reported separately.

The success rate on evaluation of safe pedestrian sub-class is the highest and for aggressive pedestrian is the least. This is observed for the trained policy from all three environments. This was expected, since safe pedestrians has high reactivity to the vehicle, and so even with a inferior policy the collisions can be avoided. The same argument holds for aggressive pedestrians where a more superior policy is required to increase the success rate. The episode length as observed from Table IV is similar in the range 6-8s for all the evaluation environments. This is 50% lesser than than

TABLE IV: Reactive pedestrians controller

Test env	Training env	Success (%)	Front col (%)	Side col (%)	Epi length (s)
agg	agg	79	17	4	7.2
	safe	72	22	6	7.1
	normal	69	24.5	6	6.7
safe	agg	88	11	0.4	7.7
	safe	90	9	0.4	6.8
	normal	87	12	1	6.6
normal	agg	80	19	0.3	7.3
	safe	77	23	0.2	7
	normal	74	26	0.6	6.6

that in the non-reactive case (14s). This denotes the policies trained are less conservative and the vehicle is able to learn cooperative interactions to perform optimally. On comparison of policies trained across different training environments, it is observed that the policy trained on aggressive environment performs the best in most of the evaluation scenarios. This is evident from Table IV where the success rates are higher for aggressive training environment. This presents a hypothesis whether the success rate can be further increased when policies are trained in more aggressive setting (named adversarial). To observe this effect, additional experiments are performed with two more sub classes: adversarial and genius. Figures 4 and 5 summarizes the results.

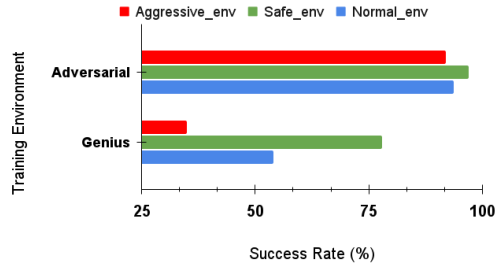

Fig. 4: The success rate of policies trained on mentioned reactive training environments (Y axis). The legends represents the evaluation environment.

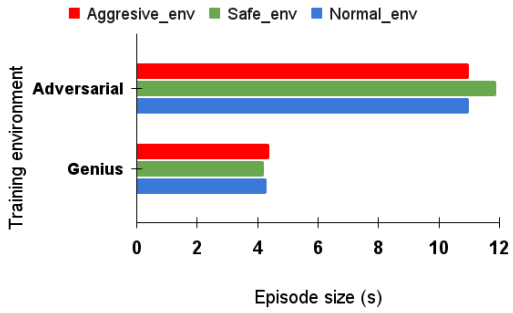

Fig. 5: The episode length of policies trained on mentioned reactive training environments (Y axis). The legends represents the evaluation environment.

The success rate of the policy trained in adversarial environment is much higher than in the genius. In fact, in all the evaluation sub-classes, it is consistently greater than 90%. It signifies the robustness of the trained policy. On the other hand, the policy trained in genius environment has much lower success rate, and the variance between the three sub-classes is also huge. This is expected, since safe pedestrians can still avoid collision by yielding to the vehicle, whereas aggressive pedestrian mostly cares about reaching the destination and can collide if the vehicle doesn't stop or slow down. This is also evident from the episode length plotted in Figure 5. The genius pedestrian training environment has all episode length of nearly 4s which is also the minimum time to reach the goal. It means that the vehicle learns a policy to accelerate as fast as possible towards the goal. During training time, the genius pedestrian can skillfully avoid the collisions and hence, vehicle does not learn any useful cooperative actions such as brake completely or slow down. This negatively impacted the performance on aggressive, safe, and normal pedestrians.

D. Mutiple pedestrian environment

The pedestrians are initialized with a random type from the set of: aggressive, safe, normal, and genius. The initialization of other states, such as initial velocity of vehicle, initial velocity of pedestrians, and random time stamp for pedestrian to start crossing, are kept similar to the previous experiments. The evaluation scheme was also retained from the previous experiments along with the policy trained from adversarial environment, which has shown best performance from previous experiments. Since, the current learning framework takes inputs of only a single pedestrian state, a naive approach of using the states of the nearest pedestrian for generating action sequences was attempted. The results of the experiment are summarized on Table V.

TABLE V: Multi-pedestrian environment evaluation summary

Success (%)	Front col (%)	Side col (%)	Episode length (s)
84.3	13.5	2.1	11.6

The applied approach of using nearest pedestrian states worked reasonably well with the success rate of 84%. The average episode length is not too long, and vehicle was able to avoid collisions by cooperative interactions with the pedestrian. To improve the success rate, one more experiment was performed, where the first layer of multi-layer perception network was expanded to take inputs of all the pedestrian states. This was done for both actor and critic network. But during the training step, no convergence was observed using this approach. At every step of policy and value iteration, the episodes ended at maximum time limit of 50s with no terminal states (success or collision). This implies the deficiency of the present network architecture to scale up for training with multiple pedestrians states.

V. Conclusion

In the presented study, we explore how a learning-based vehicle controller is influenced by the pedestrian's reactive behaviors. The RL framework employed, We employ one

of the model-free algorithms, an actor-critic with proximal policy optimization, to learn the safe vehicle policy, and it has shown promising results compelling to handle diverse pedestrian dynamics. The experiments are conducted on different pedestrian behavioural classes and sub-classes. It was empirically demonstrated that the reactive pedestrians enrich the set of explored actions (i.e. yielding and speeding up) and hence help to train more effective policies. It was also observed that training with more adversarial pedestrians generates superior policies compared to skillful or genius pedestrians. A comparison to a baseline solution of a heuristic controller was also performed. It was observed the presented RL-based controller could perform better than baseline solution when pedestrians were reactive. Lastly, the proposed framework was evaluated on a multi-pedestrian environment. There was some degree of success by employing the states of nearest pedestrian to generate actions. Though, when states of all pedestrians are integrated, the network architecture seems lacking to train successful policies.

In future work, the network architecture can be upgraded for training in multi-agent environment. Currently, our controller network is not sufficient to handle interactions with multiple pedestrians so that we considered only the closest pedestrian for the policy learning. However, this can be improved by replacing the network with a deeper, advanced network architecture. The pedestrian behavioural models employed can also be upgraded to account for hidden variables which affects pedestrian actions such as following the nearest neighbour, and interactive coordination among pedestrians.

REFERENCES

[1] C. Fan, K. Miller, and S. Mitra, "Fast and guaranteed safe controller synthesis for nonlinear vehicle models," in *International Conference on Computer Aided Verification*, pp. 629–652, Springer, 2020.

[2] S. L. Herbert, M. Chen, S. Han, S. Bansal, J. F. Fisac, and C. J. Tomlin, "Fastrack: A modular framework for fast and guaranteed safe motion planning," in *2017 IEEE 56th Annual Conference on Decision and Control (CDC)*, pp. 1517–1522, IEEE, 2017.

[3] M. Rungger and M. Zamani, "Scots: A tool for the synthesis of symbolic controllers," in *Proceedings of the 19th international conference on hybrid systems: Computation and control*, pp. 99–104, 2016.

[4] C. Finucane, G. Jing, and H. Kress-Gazit, "Ltlmop: Experimenting with language, temporal logic and robot control," in *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 1988–1993, IEEE, 2010.

[5] A. Majumdar and R. Tedrake, "Funnel libraries for real-time robust feedback motion planning," *The International Journal of Robotics Research*, vol. 36, no. 8, pp. 947–982, 2017.

[6] A. Bemporad, F. Borrelli, M. Morari, *et al.*, "Model predictive control based on linear programming˜ the explicit solution," *IEEE transactions on automatic control*, vol. 47, no. 12, pp. 1974–1985, 2002.

[7] M. M. G. Ardakani, B. Olofsson, A. Robertsson, and R. Johansson, "Model predictive control for real-time point-to-point trajectory generation," *IEEE Transactions on Automation Science and Engineering*, vol. 16, no. 2, pp. 972–983, 2018.

[8] V. Raman, A. Donzé, M. Maasoumy, R. M. Murray, A. Sangiovanni-Vincentelli, and S. A. Seshia, "Model predictive control with signal temporal logic specifications," in *53rd IEEE Conference on Decision and Control*, pp. 81–87, IEEE, 2014.

[9] H. Bai, S. Cai, N. Ye, D. Hsu, and W. S. Lee, "Intention-aware online pomdp planning for autonomous driving in a crowd," in *2015 ieee international conference on robotics and automation (icra)*, pp. 454–460, IEEE, 2015.

[10] Y. Luo, P. Cai, A. Bera, D. Hsu, W. S. Lee, and D. Manocha, "Porca: Modeling and planning for autonomous driving among many pedestrians," *IEEE Robotics and Automation Letters*, vol. 3, no. 4, pp. 3418–3425, 2018.

[11] N. Deshpande and A. Spalanzani, "Deep reinforcement learning based vehicle navigation amongst pedestrians using a grid-based state representation," in *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*, pp. 2081–2086, IEEE, 2019.

[12] D. Yang, K. Redmill, and Ü. Özgüner, "A multi-state social force based framework for vehicle-pedestrian interaction in uncontrolled pedestrian crossing scenarios," in *2020 IEEE Intelligent Vehicles Symposium (IV)*, pp. 1807–1812, IEEE, 2020.

[13] D. Yang, Ü. Özgüner, and K. Redmill, "A social force based pedestrian motion model considering multi-pedestrian interaction with a vehicle," *ACM Transactions on Spatial Algorithms and Systems (TSAS)*, vol. 6, no. 2, pp. 1–27, 2020.

[14] V. R. Konda and J. N. Tsitsiklis, "Actor-critic algorithms," in *Advances in neural information processing systems*, pp. 1008–1014, 2000.

[15] S. Haykin, *Neural networks: a comprehensive foundation*. Prentice Hall PTR, 1994.

[16] A. F. Agarap, "Deep learning using rectified linear units (relu)," *arXiv preprint arXiv:1803.08375*, 2018.

[17] I. Goodfellow, Y. Bengio, and A. Courville, "Softmax units for multinoulli output distributions. deep learning," 2018.

[18] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*, 2017.

[19] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba, "Openai gym," *arXiv preprint arXiv:1606.01540*, 2016.

[20] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer, "Automatic differentiation in pytorch," 2017.