Human-centric Autonomous Driving in an AV-Pedestrian Interactive Environment Using SVO

Luca Crosato
Northumbria University
Newcastle Upon Tyne, UK
luca.crosato@northumbria.ac.uk

Chongfeng Wei
Northumbria University
Newcastle Upon Tyne, UK
chongfeng.wei@northumbria.ac.uk

Edmond S. L. Ho
Northumbria University
Newcastle Upon Tyne, UK
e.ho@northumbria.ac.uk

Hubert P. H. Shum
Durham University
Durham, UK
hubert.shum@durham.ac.uk

***Abstract*—As Autonomous Vehicles (AV) are becoming a reality, the design of efficient motion control algorithms will have to deal with the unpredictable and interactive nature of other road users. Current AV motion planning algorithms suffer from the freezing robot problem, as they often tend to overestimate collision risks. To tackle this problem and design AV that behave human-like, we integrate a concept from Psychology called Social Value Orientation into the Reinforcement Learning (RL) framework. The addition of a social term in the reward function design allows us to tune the AV behaviour towards the pedestrian from a more reckless to an extremely prudent one. We train the vehicle agent with a state of the art RL algorithm and show that Social Value Orientation is an effective tool to obtain pro-social AV behaviour.**

I. Introduction

In the past few years, an increasing number of automotive industries and researchers are investigating how road safety and performances may be affected by autonomous driving systems. Even though we have witnessed a rapid spread of Advanced Driver-Assistance Systems (ADAS), the Global Status Report on Road Safety [1] reports an unacceptably high number of deaths on the world's roads, with an estimated 1.35 million people dying each year.

One of the major challenges in autonomous driving is *collision-free navigation* in cluttered and interactive environments. This requires solving a constrained optimisation problem, where the time to reach a goal location is minimised while being subject to collision avoidance constraints and the vehicle's dynamics. Although current motion control algorithms can provide a solution to the problem, AV behaviour usually suffers from two main issues. First, it can be overly conservative, causing traffic congestion and allowing other road users to exploit it for their advantage. Secondly, other road users can have a hard time predicting the AV intentions, as the behaviour can be very different from those of other human drivers.

In this paper, we propose to solve the above issues by integrating concepts from social psychology directly into the AV motion controller design. Social psychology defines the concept of Social Value Orientation (SVO) as a value that describes how much a person values other people's welfare compared to their own. In a general perspective, we can think of traffic participants as individuals that try to optimise their *own* objective or utility function in relation to those of the *others*.

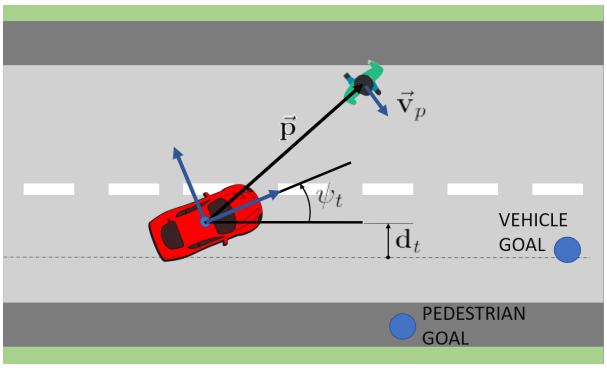

Fig. 1: Scenario illustration. The vehicle measures the pedestrian position and velocity and adjusts its longitudinal acceleration to balance time efficiency and pedestrian safety.

A controller design framework that intrinsically uses the concept of utility function is Reinforcement Learning (RL). Agents trained with RL have been capable of exceeding human-level performance in video games [2, 3], in the game of Go [4], in continuous control problems [5], and robotics [6, 7]. We let the vehicle agent learn the best control action based on the experience of the dynamic interactions with a pedestrian in a simulated environment. Rather than treating the pedestrian as a mere moving obstacle, we emulate a real-world pedestrian motion with a social force-based model and add a social term in the reward function that is based on Social Value Orientation.

We consider a typical road scenario with pedestrian crossing and train a set of RL agents with different SVO. We show how this reward choice produces a controller that naturally exhibits human-like behaviour and how we can achieve various behaviours for the Autonomous Vehicle by properly tuning SVO value at training time, ranging from more aggressive to more cautious ones.

This paper contribution is twofold:

1) We introduce the concept of SVO in the RL reward function design to obtain control policies that take the pedestrian goal into account, achieving egoistic or pro-social AV behaviour;
2) We show a successful application of a state of the art

RL algorithm, called Soft-Actor Critic, to the pedestrian collision avoidance problem.

II. Related Work

Most traditional methods to compute safe trajectories for autonomous vehicles can be classified into three main categories: input space discretisation with collision checking, randomized planning, such as rapidly exploring random trees, and constrained optimisation. A detailed survey about navigation for autonomous vehicles can be found in [8]. Here we will focus on recent learning-based approaches that are more relevant to this work.

Reinforcement Learning applications in the field of autonomous driving mainly focus on navigation amongst other vehicles, and the problem of pedestrian collision avoidance in structured environments is a less studied one. One of the first RL applications to the field of autonomous driving can be found in [9]. In this work, RL was used to train an AV in the TORCS simulator to drive across a diverse range of track geometries. In [10], [11], [12] Reinforcement Learning is used to train a vehicle agent to successfully learn an automated lane change policy. Deshpande et al. [13] trained a Deep Q-Network vehicle agent at a typical intersection crossing scenario. The pedestrian information is represented in a grid-based approach as a state space input to the learning agent. Sallab et al. [14], have used Recurrent Neural Networks in the Reinforcement Learning framework to account for Partially Observable scenarios and integrated attention models in the framework, making use of attention networks to reduce computational complexity. In [15] the authors used a two-level hierarchical approach that integrates model-free deep learning with model-based path planning. A local neural network motion controller is trained with RL and, at the high level, a path planner uses a 2D map to compute a path from the robot's current location to the goal. More recently, Chen et al. [16] trained several state-of-the-art model-free deep RL algorithms to tackle the problem of urban navigation, with focus on a roundabout scenario with multiple vehicles. Their method is trained and evaluated in the CARLA simulator [17]. Sun et al. developed a [18] courteous car model based on Inverse Reinforcement Learning for lane merging scenarios. A direct application of SVO to the RL has been explored in the framework of Inverse Reinforcement Learning [19] with the purpose of predicting other drivers' behaviour. The optimal car control law is then computed by approximating the Nash equilibrium with a nonlinear optimiser, in a game-theoretic setting.

To summarise, studies in the field of pedestrian collision avoidance mainly focus on unstructured scenarios, where the vehicle and the pedestrians share a common space. In our study, we focus on a typical lane-crossing scenario, where the pedestrian and the vehicle usually occupy separate regions of space, i.e. the road and the pavement, and interact only at some predefined regions, for example, at crossings. To the knowledge of the authors, this is the first time that SVO is used to shape the reward function in RL to the problem of pedestrian collision avoidance in structured scenarios.

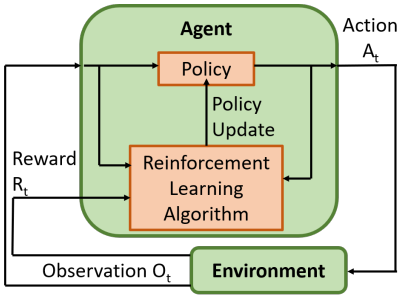

Fig. 2: The RL framework. As the agent interacts with the environment, the RL algorithm updates the policy function based on the experience gathered in order to improve the policy and achieve better cumulative reward in the future.

III. Technical Background

A. Reinforcement Learning

In a typical reinforcement learning framework, see Figure 2, an agent, which our case represents the ego-vehicle, interacts with its environment at discrete time steps. The agent performs an action a_t which causes a change in the environment state s_t. The action taken is chosen according to a policy $\pi(a|s)$, which is a probabilistic function that maps the current state to an action. In turn, the environment gives the agent a numerical reward r_t and an observation of the current environment state o_t. The goal of an RL algorithm is to learn an optimal policy π^* that maximises the expected future total rewards, which is defined as:

$$J(\pi) = \mathbb{E}_\pi[\sum_{k=0}^{\infty} \gamma^k r_{t+k}] \tag{1}$$

A reinforcement learning problem is formulated as a Markov Decision Process (MDP) (S, A, P, R), where S is the set of states, A is the set of actions, P is the state transition probability, and R is the reward function.

B. Soft-Actor Critic Algorithm

The Soft-Actor Critic algorithm (SAC) is an RL algorithm that combines the RL framework with the principle of maximum entropy. The policy seeks to maximise a modified version of the expected future reward which is defined as:

$$\max_\pi J(\pi) = \sum_{t=0} \mathbb{E}_\pi[r(s_t, a_t) + \alpha \mathcal{H}(\pi(\cdot|s_t))] \tag{2}$$

$J(\pi)$ maximises both the expected cumulative reward and an entropy term $\mathcal{H}(\pi(\cdot|s_t))$, to encourage exploration at the time of training and improve training speed. The parameter α is known as the temperature and it affects the weight of the entropy term. Precisely, SAC aims to learn three functions: the policy network with parameter θ, π_θ, a soft Q-value function parametised by w, Q_w, and a soft state value function parametrised by ψ, V_ψ. The experience gathered by the agent is stored in a replay buffer and, similar to DQN and DDPG, the Q network and the value network are trained using supervised

learning with the data contained in a replay buffer. The targets for the network update are defined as:

$$\hat{Q}(s_t, a_t) = r(s_t, a_t) + \gamma \mathbb{E}_{s_{t+1} \sim \rho_{pi}(s)} \left[V_\psi(s_{t+1}) \right] \quad (3)$$

$$\hat{V}(s_t) = \mathbb{E}_{a_t \sim \pi_\theta} \left[Q_w(s_t, a_t) - \alpha \log \pi_\theta(a_t | s_t) \right] \quad (4)$$

The policy is parametrised as stochastic neural network

$$a_t = f_\theta(\epsilon_t, s_t) \quad (5)$$

where ϵ_t is an input noise vector, sampled from a Gaussian distribution. Then objective function for policy optimization can be rewritten as:

$$J_\pi(\theta) = \mathbb{E}_{s_t, \epsilon_t} \left[\alpha \log(\pi_\theta(f_\theta(\epsilon_t, s_t) | s_t) - Q_\theta(s_t, f_\theta(\epsilon_t, s_t)) \right] \quad (6)$$

IV. Methodology

In social psychology, Social Value Orientation (SVO) is a value that describes how much a person values other people's welfare in relation to their own. With SVO theory in mind, we can model each individual as a decision-making agent that maximises their own utility function. Such a utility function can be expressed as a combination of the ego agent's utility U_{self} and other agents' utility U_{other}:

$$U_{total} = \cos(\varphi) U_{self} + \sin(\varphi) U_{other} \quad (7)$$

where φ is the SVO. It is an angle, whose value affects the weights of the two utility terms, and therefore the balance between the selfish reward and the altruistic reward. As we can see in Figure 3, we can characterise the personality of each individual with the SVO value. For example, an SVO value of 90° corresponds to fully altruistic behaviour, whereas an SVO value of 0° corresponds to an individualistic agent. In our work, we focused on SVO values between 0° and 90°, as we want the AV to exhibit pro-social behaviour and yield to the pedestrian if necessary to avoid dangerous situations.

SVO has been previously used to design controllers in a game-theoretic setting [19], but this demands long complex computations to solve for a Nash equilibrium. In our work, we try to mitigate the computational cost of the optimisation problem by using SVO in the RL framework, thereby moving the computational cost from execution time to training time, in a learning-based fashion. We integrate the SVO concept directly in the MDP model formulation, by constructing a reward function that is composed of two terms, one that models the car's own objective U_{self} and one that models the pedestrian's objective U_{other}. As model-free RL algorithms are computationally less complex and do not require an accurate representation of the environment to be effective, we choose the SAC algorithm, which has proven to be very effective for autonomous car traffic navigation [16]. The SAC has also the advantage of using a continuous action space, which is more suitable for our problem. To generate realistic and interactive experience at training and test time, we use social forces to model the pedestrian's behaviour. We adopted a model similar to the one used in [20], where we considered a single vehicle and neglected the interaction with other pedestrians. We include an awareness probability for the pedestrian to improve robustness to collisions and avoid overfitting of the pedestrian behaviour. This way, the trained policy will be able to deal with dangerous situations where the pedestrian starts crossing without seeing the vehicle. In the next section, we introduce the state and action spaces of the MDP and in section IV-B we describe the social reward function design with SVO.

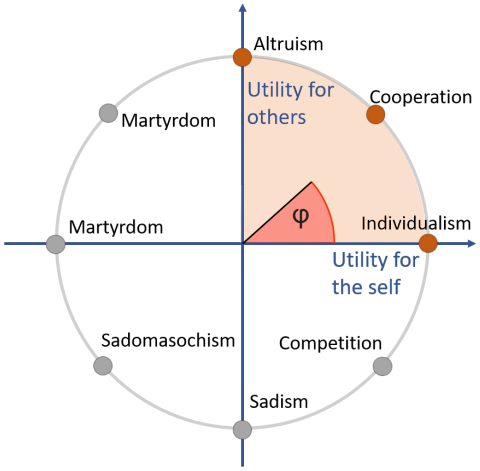

Fig. 3: Social Value Orientation ring. The SVO value φ affects the behaviour of the ego-vehicle.

A. MDP Formulation

1) State Space:
In our model, we focused on a scenario consisting of a straight lane and a single pedestrian, see Figure 1. We assume the ego-vehicle can access a reference path computed by a Path-Planning module using the GPS and global map information and that it is able to locate itself with respect to this reference path. The purpose of the control algorithm is to adapt the vehicle current trajectory to the reference track based on the pedestrian's behaviour. As for pedestrian detection, we assume that the vehicle is equipped with a LiDAR, which measures the pedestrian position relative to the vehicle. Therefore, we make the assumption that the state space available to the ego-vehicle consists of:

- the offset d_t of the vehicle centre of gravity from the vehicle reference path;
- the vehicle orientation relative to the reference path direction ψ_t;
- the vehicle longitudinal velocity v_t^{ego};
- the pedestrian position $\mathbf{r}_t^{ped}$ and velocity $\mathbf{v}_t^{ped}$ relative to the vehicle.

$$s_t = [d_t, \psi_t, v_t^{ego}, \mathbf{p}_t, \mathbf{v}_t^{ped}]^T \in \mathbb{R}^7 \quad (8)$$

2) Action Space: We adopted a continuous state-space representation for the action space. In particular, the action space consists of the longitudinal acceleration a_t. The action a_t, computed by the policy network is normalised in the interval $[-1, 1]$ and than rescaled in the interval $[-g/2, g/2]$.

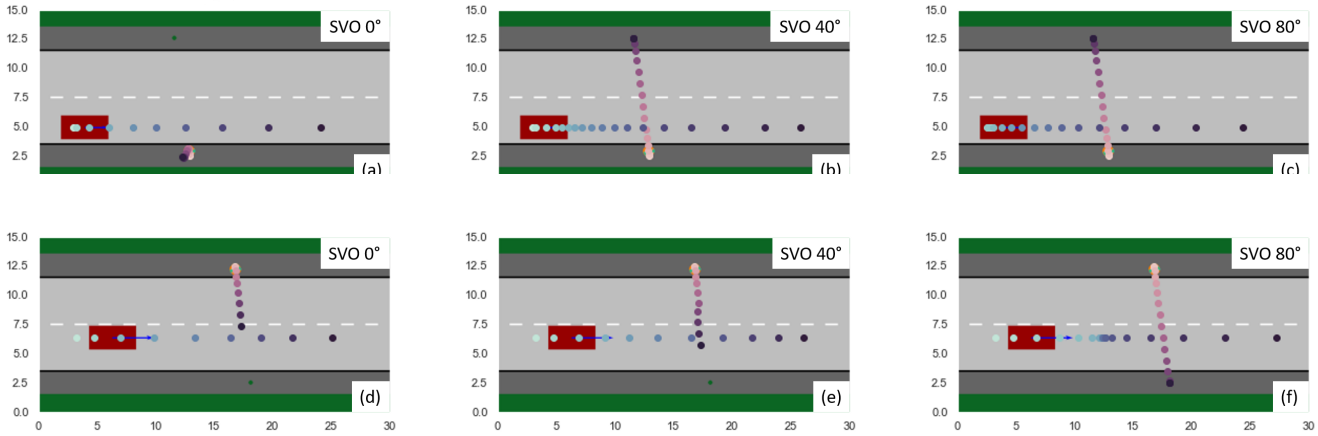

Fig. 4: Pedestrian and vehicle agent trajectories for two episodes and three SVO values. Figures in the same row refer to the same episode and share the same initial conditions but have different SVO values. The temporal progression is indicated by coloring the trajectories from lighter to darker colors.

B. Social Reward Function

In order to design an AV that behaves similarly to a human driver, we introduce the SVO concept directly inside the reward function. In particular, the designed reward function consists of two terms:

$$r(s_t, a_t) = \cos\varphi \cdot r_{car}(s_t, a_t) + \sin\varphi \cdot r_p(s_t, a_t) \tag{9}$$

where r_{car} indicates a reward function that takes the vehicle own performance parameters into account and the r_p term is a term that favours the pedestrian's goal.

In particular, the car's reward function is also a combination of multiple terms:

$$r_{car}(s_t, a_t) = r_c + r_g + r_v \tag{10}$$

For avoiding collisions with pedestrians, a penalty r_c of -30 is given to the car in case of collision and the episode is terminated. A positive reward r_g of $+30$ is given to the agent when it reaches the goal. Finally the term r_v is a speed reward that encourages the car to reach the goal in the minimum amount of time. It is expressed as:

$$r_v = c_1 \mathbf{v_t} \cdot \hat{\mathbf{n}} = c_1 \cos\psi_t v_t \tag{11}$$

The term $\mathbf{v_t} \cdot \hat{\mathbf{n}}$ is the dot product between the vehicle velocity and a unit vector representing the reference path forward direction. This encourages the car to go as fast as possible and at the same time discourages the car from moving backwards.

The second term of equation 9 is used to model the pedestrian's objective. We assume that if a pedestrian is crossing, they are trying to reach their goal in the minimum amount of time possible. Also, since we want our RL agent to behave pro-socially, it is reasonable to give a positive reward only if the pedestrian is crossing in front of the vehicle. Let $\mathbf{v}_p$ be the pedestrian velocity, then the pedestrian reward function can be expressed as:

$$r_p = \begin{cases} c_2 ||\mathbf{v}_p^{ped}||, & \text{if } \textit{pedestrian is crossing} \quad \textbf{and} \quad x_p > x_v \\ 0, & \text{otherwise} \end{cases} \tag{12}$$

C. Network Training

We trained a total of 9 different policies with different SVO values from $0°$ to $80°$. For each SVO value, an agent is trained for 20,000 steps, at each a tuple consisting of the current observed state, action taken, reward received, and the subsequent next state is stored in a replay buffer. A normally distributed action noise is also added to the actions taken by the agent at training time to favour exploration. The replay buffer size is equal to the number of steps of the entire simulation so that the entire experience gathered by the agent is used during training. The learning rate is initially set to 0.001, then it decreases linearly to 10% of the initial value. The batch size, τ and γ parameters are set to 256, 0.005, and 0.99 respectively.

The neural network architecture consists of two fully connected layers with 256 hidden neurons each, shared by both the actor and critic networks. A simple fully-connected multilayer perceptron network was used, as the input consists of features of the ego-vehicle and the pedestrian rather than raw images.

V. Experiments

A simulator was developed using the Python programming language to validate the proposed method. The overall system architecture is represented in Figure 6. The simulator models the physics of the problem, i.e. it performs time integration and simulates the interaction between the ego-vehicle and the pedestrian. The simulator is wrapped in an OpenAI Gym [21] environment, which communicates with a SAC agent of the Stable-Baselines3 package [22]. The OpenAI Gym package is an open-source package for developing reinforcement learning environments, whereas Stable Baselines3 [23]

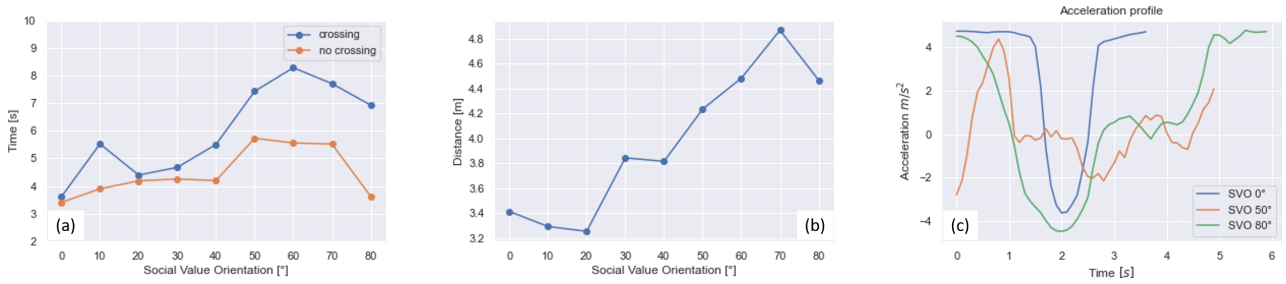

Fig. 5: (a) Average time to complete the task if the pedestrian is crossing (blue) or if the pedestrian is not crossing (orange). (b) Average minimum distance between the pedestrian and the vehicle. (c) Vehicle acceleration profile with the same episode initial conditions for three SVO values.

is an open-source python package that implements state-of-the-art Reinforcement Learning algorithms. The SAC agent performs an action based on the observations provided by the environment and improves the policy. After executing an action, the environment state is updated and a new set of observations is given to the agent. This cycle repeats until the policy converges.

A. Scenario

The road scenario is represented in Figure 1. It consists of a single vehicle and a single pedestrian on a straight road. At the start of each episode, the pedestrian randomly spawns on either the bottom or the top pavement with equal probability and also has a fixed probability of crossing the road. The pedestrian spawn position on the pavement is chosen according to a normal distribution. If the pedestrian crosses, a random goal position on the opposite side of the road is generated according to a normal distribution, otherwise the pedestrian simply walks along the pavement. This way we are able to generate episodes with both crossing behaviours and the RL agent learns to distinguish between them and exploit that to its advantage. We chose a value of 0.9 for the pedestrian crossing probability in order to generate more episodes with actual car-pedestrian interactions, as this kind of interactions are more complex and the RL agent needs more episodes to learn the correct policy in these situations. The car spawn position and velocity are also chosen randomly according to a normal distribution. The ego vehicle goal position is located along the centre of the lane, 30 m away in front of the ego-vehicle starting point. An episode terminates if the car reaches its goal or if a collision occurs.

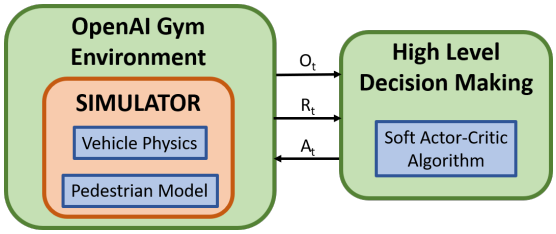

Fig. 6: System architecture.

B. Results

We carried out computer simulation experiments to evaluate the trained agents. Each agent is tested with 100 training episodes. In order to have comparable results for each agent, the testing episodes share the same initial conditions. Across all the episodes, no collisions with the pedestrians were detected. We compare the trained agents in terms of pedestrian safety and time efficiency in achieving the goal.

1) Quantitative results: Simply by considering the pedestrian crossing's velocity into the pedestrian reward term, we can see that the car automatically learns a more pro-social behaviour. This more pro-social behaviour corresponds to both increased safety for the pedestrian and an increased time to complete the task, indicating that the car is more likely to slow down and yield to the pedestrian. Figure 5a shows the average time taken by the autonomous vehicle to reach their goal. As the SVO increases, the car is more likely to yield to the pedestrian, therefore the average time to reach the goal has an increasing trend. We also computed the minimum distance across all testing episodes for each agent and plotted it in Figure 5b. As expected, this parameter also has an increasing trend.

We plot two curves depending on whether the pedestrian is crossing or not in the episodes. For low SVO values, the time to reach the goal is similar for the two curves, because the car arrogantly occupies the lane before the pedestrian is allowed to start crossing. From our simulations, we have seen that an SVO of at least 30° is required to see a significant change in the AV social behaviour, which explains why the two curves are quite similar at the beginning. For higher SVO values, the time to reach the goal when the pedestrian is not crossing also increases. This is due to the fact that the car exhibits a more cautious behaviour and it slows down when close to the pedestrian, even if the pedestrian is not crossing.

Figure 5c shows the acceleration profile for an episode with crossing pedestrian and three different SVO values. We can see that overall the acceleration value is lower for the SVO of 80° compared to the one at 0°, indicating that the car moves at a slower speed. Also, for an SVO of 0°, the car actually starts to slow down much later than the SVO of 0°. The acceleration profile for an SVO of 50° oscillates more than the other two. Indeed, the trajectories generated from that agent exhibit a much more hesitant behaviour compared to those at 0° and

80°. In such episodes, the pedestrian also starts hesitating and doesn't commit to any behaviour, which is why the average time to reach goal also decreases for higher SVO values. This reflects a typical scenario in which pedestrian and driver don't come to an agreement and reach an impasse, as they both try to claim the right of passing. This explains why the average time to reach goal decreases again after an SVO value of 60°.

2) Qualitative results: In Figure 4 we visualize the trajectories generated by our simulator in two episodes for three different SVO values. In the first episode the pedestrian is standing on the bottom pavement and is trying to cross the road, whereas in the second episode the situation is reversed. The reader can find a video demo of the trajectories in the supplementary material. We can see that in Figure 4a the pedestrian hasn't even started to cross the road when the episode is over. This is due to the fact that the low SVO value makes the car behave very aggressively. We can see that in the same Episode but for higher SVO values the pedestrian manages to reach the goal before the end of the episode, meaning that the car yields. The difference between SVO values 40° and 80° is where the car stops yielding to the pedestrian: the car stops almost immediately for an SVO value of 80°, but for an SVO value of 40° the car stops closer to the pedestrian. Similar behaviours can be found in the second episode. We can see that the distance covered by the pedestrian within the end of the episode increases as the SVO value increases and that the pedestrian manages to reach their goal completely in the third figure.

VI. Conclusion and Future Work

We presented an approach to solving the pedestrian collision avoidance problem in a scenario consisting of a vehicle and a single pedestrian using a state-of-the-art Reinforcement Learning algorithm called Soft-Actor Critic. We demonstrated that by including Social Value Orientation in the RL reward function design, the trained vehicle agent behaves in a human-like fashion. The SVO value affects how much the trained agent is likely to yield to the pedestrian. To the knowledge of the authors, the Soft Actor-Critic algorithm has never been used before in the pedestrian collision avoidance problem in structured scenarios. The main assumption in this work is the presence of a single pedestrian. An immediate extension of this work will be to tackle the presence of multiple pedestrians and vehicles. We would also analyse the addition of the vehicle steering angle to the control action space. In the future, we would also like to compare the controller with more traditional ones, such as an MPC controller, for both validation and verification.

References

[1] World Health Organization et al. *Global status report on road safety 2018: Summary*. Tech. rep. World Health Organization, 2018.

[2] Volodymyr Mnih et al. "Playing atari with deep reinforcement learning". In: *arXiv preprint arXiv:1312.5602* (2013).

[3] Volodymyr Mnih et al. "Human-level control through deep reinforcement learning". In: *nature* 518.7540 (2015), pp. 529–533.

[4] David Silver et al. "Mastering the game of go without human knowledge". In: *nature* 550.7676 (2017), pp. 354–359.

[5] Timothy P Lillicrap et al. "Continuous control with deep reinforcement learning". In: *arXiv preprint arXiv:1509.02971* (2015).

[6] Xiaoyun Lei, Zhian Zhang, and Peifang Dong. "Dynamic path planning of unknown environment based on deep reinforcement learning". In: *Journal of Robotics* 2018 (2018).

[7] Jing Xin et al. "Application of deep reinforcement learning in mobile robot path planning". In: *2017 Chinese Automation Congress (CAC)*. IEEE. 2017, pp. 7112–7116.

[8] Wilko Schwarting, Javier Alonso-Mora, and Daniela Rus. "Planning and decision-making for autonomous vehicles". In: *Annual Review of Control, Robotics, and Autonomous Systems* (2018).

[9] Adithya Ganesh et al. *Deep reinforcement learning for simulated autonomous driving*. 2016.

[10] Pin Wang, Ching-Yao Chan, and Arnaud de La Fortelle. "A reinforcement learning based approach for automated lane change maneuvers". In: *2018 IEEE Intelligent Vehicles Symposium (IV)*. IEEE. 2018, pp. 1379–1384.

[11] Carl-Johan Hoel, Krister Wolff, and Leo Laine. "Automated speed and lane change decision making using deep reinforcement learning". In: *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*. IEEE. 2018, pp. 2148–2155.

[12] Junjie Wang et al. "Lane change decision-making through deep reinforcement learning with rule-based constraints". In: *2019 International Joint Conference on Neural Networks (IJCNN)*. IEEE. 2019, pp. 1–6.

[13] Niranjan Deshpande and Anne Spalanzani. "Deep Reinforcement Learning based Vehicle Navigation amongst pedestrians using a Grid-based state representation". In: *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*. IEEE. 2019, pp. 2081–2086.

[14] Ahmad EL Sallab et al. "Deep reinforcement learning framework for autonomous driving". In: *Electronic Imaging* 2017.19 (2017), pp. 70–76.

[15] Wei Gao et al. "Intention-net: Integrating planning and deep learning for goal-directed autonomous navigation". In: *Conference on Robot Learning*. PMLR. 2017, pp. 185–194.

[16] Jianyu Chen, Bodi Yuan, and Masayoshi Tomizuka. "Model-free deep reinforcement learning for urban autonomous driving". In: *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*. IEEE. 2019, pp. 2765–2771.

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

[18] Liting Sun et al. "Courteous autonomous cars". In: *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2018, pp. 663–670.

[19] Wilko Schwarting et al. "Social behavior for autonomous vehicles". In: *Proceedings of the National Academy of Sciences* 116.50 (2019), pp. 24972–24978.

[20] Dongfang Yang et al. "Sub-Goal Social Force Model for Collective Pedestrian Motion Under Vehicle Influence". In: *arXiv preprint arXiv:2101.03554* (2021).

[21] Greg Brockman et al. "Openai gym". In: *arXiv preprint arXiv:1606.01540* (2016).

[22] Tuomas Haarnoja et al. "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor". In: *International Conference on Machine Learning*. PMLR. 2018, pp. 1861–1870.

[23] Antonin Raffin et al. *Stable Baselines3*. https://github.com/DLR-RM/stable-baselines3. 2019.