EMERGENT ROAD RULES IN MULTI-AGENT DRIVING ENVIRONMENTS

Anonymous authors
Paper under double-blind review

ABSTRACT

In order for autonomous vehicles to share the road safely with human drivers, autonomous vehicles must abide by certain "road rules" that human drivers have agreed all road users must follow. "Road rules" include rules that drivers are required to follow by law – such as the requirement that vehicles stop at red lights – as well as more subtle social rules – such as the implicit designation of fast lanes on the highway. In this paper, we provide empirical evidence that suggests that – instead of hard-coding these road rules into self-driving algorithms – a scalable alternative may be to design multi-agent environments such that agents within the environments discover for themselves that these road rules are mutually beneficial to follow. We analyze what components of our chosen multi-agent environment cause the emergence of such behavior and find that two crucial factors are noisy perception and the spatial density of agents. We provide qualitative and quantitative evidence of the emergence of seven social driving behaviors, ranging from stopping at a traffic signal to following lanes. Our results add empirical support for the social road rules that countries around the world have agreed on for safe driving.

1 INTRODUCTION

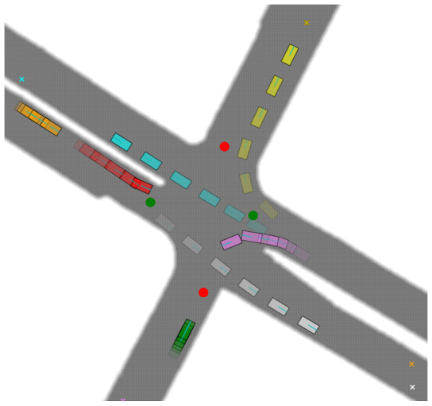
Figure 1. Multi-agent Driving Environment We train agents to travel from a→b as quickly as possible with limited perception while avoiding collisions and find that "road rules" such as lane following and traffic light usage emerge.

Public roads are significantly more safe and efficient when equipped with conventions that restrict how one may use the roads. These conventions are, to some extent, arbitrary. For instance, a "drive on the left side of the road" convention is, practically speaking, no better or worse than a "drive on the right side of the road" convention. However, the decision to reserve *some* orientation as the canonical orientation for driving is far from arbitrary in that establishing such a convention improves both safety (doing so decreases the probability of head-on collisions) and efficiency (cars can drive faster without worrying about dodging oncoming traffic).

In this paper, we investigate the extent to which these road rules – like the choice of a canonical heading orientation – can be learned in simple multi-agent driving environments. As visualized in Figure 1, our agents are initialized in random positions in different maps (either simulated or scraped from real intersections from the nuScenes dataset (Caesar et al., 2019)) and tasked with reaching a given target destination as quickly a possible without colliding with other cars or the road. Intuitively, when agents have full access to the map and exact states of other agents, the agents drive in ways that are qualitatively aggressive and un-humanlike. However, when perception is imperfect and noisy, we show in Section 5 that the agents begin to rely on constructs such as lanes, traffic lights, and safety distance in order to drive safely at high speeds.

Importantly, while prior work has primarily focused on building driving simulators with *realistic sensors* that mimic LiDARs and cameras (Dosovitskiy et al., 2017; Manivasagam et al., 2020; Yang et al., 2020; Bewley et al., 2018), we focus on the high-level design choices for the simulator – such as the definition of reward and perception noise – that determine if agents trained in the simulator exhibit *realistic behaviors*. Our hope is that the lessons in state space, action space, and reward design gleaned from this paper will transfer to simulators in which the prototypes for perception and interaction used in this paper are replaced with more sophisticated sensor simulation.

Our main contributions are as follows:

- We define a multi-agent driving environment in which agents equipped with noisy LiDAR sensors are rewarded for reaching a given destination as quickly as possible without colliding with other agents and show that agents trained in this environment learn road rules that mimic road rules common in human driving systems.
- We analyze what choices in the definition of the MDP lead to the emergence of these road rules and find that the most important factors are perception noise and the spatial density of agents in the driving environment.
- We release a suite of 2D driving environments with the intention of stimulating interest within the MARL community to solve fundamental self-driving problems.

2 Related Works

Reinforcement Learning Deep Reinforcement Learning (DeepRL) has become an immensely popular tool in control problems and has been successfully used in various complex problems like Atari (Mnih et al., 2013), Strategy Games (Peng et al., 2017; OpenAI, 2018), and Traffic Control (Wu et al., 2017a; Belletti et al., 2018). Vanilla Policy Gradient (Sutton et al., 2000) is a DeepRL algorithm that optimizes an agent's policy by using monte-carlo estimates of the expected return. Proximal Policy Optimization (Schulman et al., 2017) – which we use in this work – is an on-policy policy gradient algorithm that alternately samples from the environment and optimizes the policy using stochastic gradient descent. PPO stabilizes the Actor's training by limiting the step size of the policy update using a clipped surrogate objective function.

Multi-Agent Reinforcement Learning The central difficulties of Multi-Agent RL (MARL) include environment non-stationarity (Hernandez-Leal et al., 2019; 2017; Busoniu et al., 2008; Shoham et al., 2007), credit assignment (Agogino and Tumer, 2004; Wolpert and Tumer, 2002), and the curse of dimensionality (Busoniu et al., 2008; Shoham et al., 2007). Recent works (Son et al., 2019; Rashid et al., 2018) have attempted to solve these issues in a centralized training decentralized execution framework by factorizing the joint action-value Q function into individual Q functions for each agent. Alternatively, MADDPG (Lowe et al., 2017) and PPO with Centralized Critic (Baker et al., 2019) have also shown promising results in dealing with MARL Problems using policy gradients. In this paper, we use policy gradients for all experiments.

Emergent Behavior Emergence of behavior that appears human-like in MARL has been studied extensively (Leibo et al., 2019) for problems like effective tool usage (Baker et al., 2019), ball passing and interception in 3D soccer environments (Liu et al., 2019), capture the flag (Jaderberg et al., 2019), hide and seek (Chen et al., 2019; Baker et al., 2019), communication (Foerster et al., 2016; Sukhbaatar et al., 2016), and role assignment (Wang et al., 2020). For autonomous driving and traffic control, emergent behavior has primarily been studied in the context of imitation learning (Bojarski et al., 2016; Zeng et al., 2019; Bansal et al., 2018; Philion and Fidler, 2020). In contrast to work that studies emergent behavior in mixed-traffic autonomy (Wu et al., 2017b) and traffic signal control (Stevens and Yeh, 2016), we consider a fully autonomous driving problem in a decentralized execution framework and show the emergence of standard traffic rules that are present in transportation infrastructure.

3 Problem Setting

We frame the task of driving as a discrete time Multi-Agent Dec-POMDP (Oliehoek et al., 2016). This problem can be formally defined as a tuple $G = \langle \mathcal{S}, \mathcal{A}, \mathcal{P}, r, \rho_0, \mathcal{O}, n, \gamma, T \rangle$. $\mathcal{S}$ denotes the state space of the environment, $\mathcal{A}$ denotes the joint action space of the n agents s.t. $\bigcup_{i=1}^{n} a_i \in \mathcal{A}$, $\mathcal{P}$ is the state transition probability $\mathcal{P} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \mapsto \mathbb{R}_+$, r is a bounded reward function $r : \mathcal{S} \times a \mapsto \mathbb{R}$, ρ_0 is the initial state distribution, O is the joint observation space of the n agents s.t. $\bigcup_{i=1}^{n} o_i \in \mathcal{O}$, $\gamma \in (0, 1]$ is the discount factor, and T is the time horizon.

We parameterize the policy $\pi_\theta : o \times a \mapsto \mathbb{R}_+$ of the agents using a neural network with parameters θ. In all our experiments, the agents share a common policy network. Let the expected return for the i^{th} agent be $\eta_i(\pi_\theta) = \mathbb{E}_\tau \left[\sum_{t=0}^{H-1} \gamma^t r(s_t, (a_i)_t) \right]$, where $\tau = (s_0, (a_i)_0, \ldots, s_{T-1}, (a_i)_{H-1})$ is the trajectory of the agent, $s_0 \sim \rho_0$, $(a_i)_t \sim \pi_\theta((a_i)_t|(o_i)_t)$, and $s_{t+1} \sim \mathcal{P}(s_{t+1}|s_t, \bigcup_{i=1}^{n}(a_i)_t)$. Our objective is to find the optimal policy which maximizes the utilitarian objective $\sum \eta_i$.

Reward We use high-level generic rewards and avoid any extensive reward engineering. The agents receive a reward of +1 if they successfully reach their given destination. They are rewarded by -1 if they collide with any other agent or go off the road. We additionally regularize the actions of the agents to encourage smooth actions. In the event of a collision, the simulation for that agent stops there. In an inter-agent collision, we penalize both the agents equally with a -1 reward without attempting to determine which agent was responsible for the collision. Finally, we add a normalized penalty proportional to the longitudinal distance of the agent from the destination. We ensure that the un-discounted sum of each component of the reward for an agent over the entire trajectory is bounded in $[0, 1]$ or $[-1, 0]$.

Map and Goal Representation We use multiple environments: four-way intersection, highway tracks, and real-world road patches from nuScenes (Caesar et al., 2019), to train the agents. The initial state distribution ρ_0 is defined by the drivable area in the base environment. The agents "sense" a traffic signal if they are near a traffic signal and facing the traffic signal. In all but our communication experiments, agents have the ability to communicate exclusively through the motion that other agents observe them execute. In our communication experiments, we open a discrete communication channel designed to mimic turn signals and discuss the direct impact on agent behavior. Additionally, to mimic a satnav dashboard, the agents observe the distance from their goal, the angular deviation of the goal from the current heading, and the current speed.

LiDAR observations We simulate a LiDAR sensor for each agent by calculating the distance to the closest object – dynamic or static – along each of n equi-angular rays. We restrict the range of the LiDAR scan to be 50m. Human eyes themselves are imperfect sensors and are easily thwarted by weather, glare, or visual distractions; in our experiments, we study the importance of this "visual" sensor by introducing noise in the sensor. To give agents the capacity to infer the velocity and acceleration of nearby vehicles, we concatenate the LiDAR observations from multiple past timesteps.

4 Policy Optimization

4.1 Policy Parameterization

In our experiments we consider the following two parameterizations for our policy network(s):

1. **Fixed Track Model**: We optimize policies that output a Multinomial Distribution over a fixed set of discretized acceleration values. This distribution is defined by $\pi_\phi(\mathfrak{a}|\mathfrak{o})$, where π_ϕ is our policy network, $\mathfrak{a}$ is the acceleration, and $\mathfrak{o}$ is the observation. This acceleration is used to drive the vehicle along a fixed trajectory from the source to target destination. This model trains efficiently but precludes the emergence of lanes.
2. **Spline Model**: To train agents that are capable of discovering lanes, we use a two-stage formulation inspired by Zeng et al. (2019) in which trajetories shapes are represented by clothoids and time-dependence is represented by a velocity profile. Our overall policy is factored into two "subpolicies" – a spline subpolicy and an acceleration subpolicy. The spline subpolicy is tasked with predicting the spline along which the vehicle is supposed to be driven. This subpolicy conditions on an initial local observation of the environment and predicts the spline. More details can be found in Section 5. We use a Centripetal Catmull Rom Spline (Catmull and Rom, 1974) to parameterize this spatial path. The acceleration subpolicy follows the same parameterization from Fixed Track Model, and controls the agent's motion along this spline. We formalize the training algorithm for this bilevel problem in Section 4.4.

Note that the fixed track model is a special case of the spline model parameterization, where the spline is hard-coded into the environment. These splines can be extracted from lane graphs such as those found in the HD maps provided by the nuScenes dataset.

4.2 Proximal Policy Optimization using Centralized Critic

We consider a Centralized Training with Decentralized Execution approach in our experiments. During training, the critic has access to all the agents' observations while the actors only see the local observations. We use Proximal Policy Optimization (PPO) (Schulman et al., 2017) and Generalized

Advantage Estimation (GAE) (Schulman et al., 2015) to train our agents. Let vf_ϕ denote the value function network. To train our agent, we optimize the following objective:

$$\begin{aligned} L_1(\phi) = \hat{\mathbb{E}}_t \big[\min(\tilde{r}(s_t, a_t)\hat{A}_t,\ \text{clip}(\tilde{r}(s_t, a_t), 1-\epsilon, 1+\epsilon)\hat{A}_t) \\ -c_1(vf_\phi(s_t) - vf_t^{target}) - c_2 H[\pi_\phi(o_t)]\big] \\ \text{where } \tilde{r}(s_t, a_t) = \frac{\pi_{\phi_i}[a_t|o_t]}{\pi_{\phi_{i-1}}[a_t|o_t]}, \quad \hat{A}_t = \text{ Estimated Advantage at step } t \end{aligned}$$

Training is performed using a custom adaptation of SpinningUp (Achiam, 2018) for MARL and Horovod (Sergeev and Balso, 2018). The agents share a common policy network in all the reported experiments. In our experiments, the number of agents present in the environment can vary over time, as vehicles reach their destinations and new agents spawn. To account for the dynamic number of agents and a permutation invariance to their ordering, the centralized critic takes as input the mean of the latent vector obtained from all the observations.

4.3 Single-Step Proximal Policy Optimization

In a single-step MDP, the expected return modelled by the critic is equal to the reward from the environment as there are no future timesteps. Hence, optimizing the critic is unnecessary in this context. Let $\overline{R}_t$ denote the normalized reward. The objective function defined in Sec 4.2 reduces to:

$$L_2(\theta) = \hat{\mathbb{E}}_t \left[\min(\tilde{r}(s_t, a_t)\overline{R}_t, \text{clip}(\tilde{r}(s_t, a_t), 1-\epsilon, 1+\epsilon)\overline{R}_t - c_2 H[\pi_\theta(o_t)]\right]$$

4.4 Bilevel Optimization for Joint Training of Spline/Acceleration Subpolicies

In this section, we present the algorithm we use to jointly train two RL subpolicies where one subpolicy operates in a single step and the other operates over a time horizon $T \geq 1$. The subpolicies operate oblivious of each other, and cannot interact directly. The reward for the spline subpolicy is the undiscounted sum of the rewards received by the acceleration subpolicy over the time horizon. Pseudocode is provided in Algorithm 1.

Algorithm 1: Alternating Optimization for Spline and Acceleration Control

```
Result: Trained Actors $\pi_\theta$ and $\pi_\phi$
$\pi_\theta \leftarrow$ Actor for predicting the Spline Path;
$\pi_\phi, vf_\phi \leftarrow$ Actor Critic for Acceleration Control;
for i = 1 ... N do
    /* Given $\pi_\phi$ optimize $\pi_\theta$ */
    for k = 1 ... $K_1$ do
        Collect set of Partial Trajectories $\mathcal{D}_k$ using $a_s \sim \pi_\theta(a_s|o_k^s)$ and $a_a \leftarrow \arg\max \pi_\phi(a_a|o_k^a)$;
        Compute the normalized rewards $\overline{R}_t$;
        Optimize the parameters $\theta$ using the objective $L_2(\theta)$
    end
    /* Given $\pi_\theta$ optimize $\pi_\phi$ and $vf_\phi$ */
    for k = 1 ... $K_2$ do
        Collect set of Partial Trajectories $\mathcal{D}_k$ using $a_s \leftarrow \arg\max \pi_\theta(a_s|o_k^s)$ and $a_a \sim \pi_\phi(a_a|o_k^a)$;
        Compute the advantage estimates $\hat{A}_t$ using GAE;
        Optimize the parameters $\phi$ using the objective $L_1(\phi)$
    end
end
```

5 Emergent Social Driving Behavior

We describe the various social driving behaviors that emerge from our experiments and analyze them quantitatively. Qualitative rollouts are provided in our supplementary material.

5.1 Stopping at a Traffic Signal

In 4 way intersections, traffic signals play an important part by imposing an ordering on the direction of traffic flow. In this experiment, we simulate a simple four-way intersection, where agents need to drive from one road pocket to the other in minimum time. We give them the ground truth trajectory shape and the agents need to only optimize their accelerations.

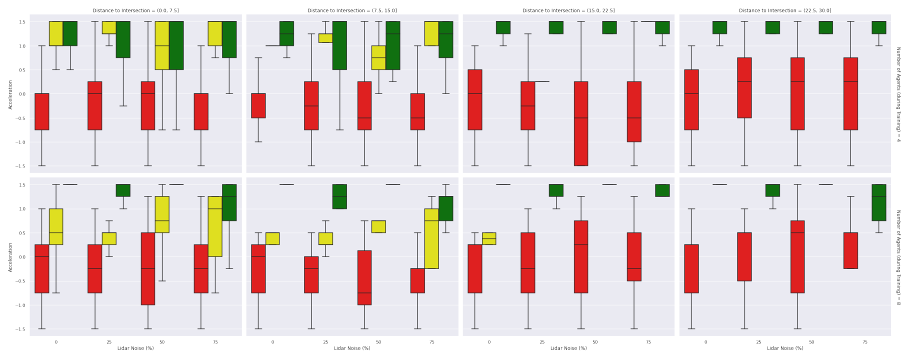

Figure 2. Traffic light usage Actions taken by the agents with varying amount of perception noise.

In Figure 2, we visualize if the variation of the action taken by the agent at varying distance from the intersection and with different signals. We can see that when the agents from far from the intersection, the effect of signal is less prominent that when they are closer to the intersection. Additionally we see a consensus across the agents where they decide to have lower acceleration when they see a red light and higher value on green light.

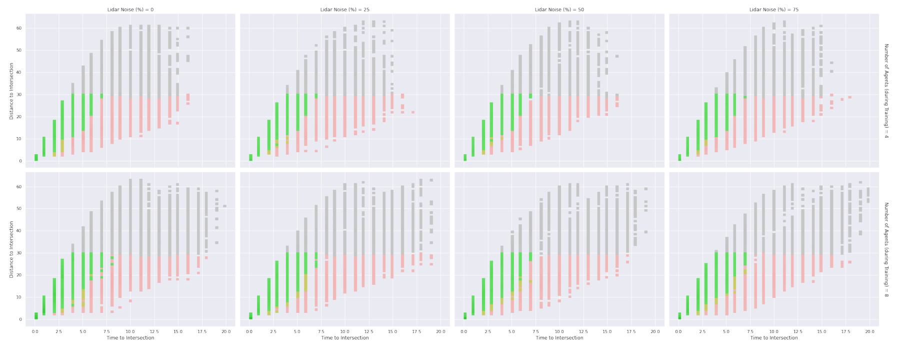

Figure 3. Traffic light usage Actions taken by the agents with varying amount of perception noise.

In Figure 3, we show that the agents do respond to red light and stop. The colors represent the three traffic lights (red, green and yellow) and gray represents a state where the agent can not see the signal. When the signal is green agents are able to reach the intersection much faster compared to when the signal is red. This behavior is consistent across increasing perception noise. We observe similar behavior with increasing number of agents, the only difference being that the waiting time increases due to increase in the number of agents.

Note that the agents merely observe a ternary value representing the state of the traffic light, not color. To make the plots in this section, for each converged policy, we visually inspect rollouts to find a permutation of the ternary states that aligns with human red/yellow/green traffic light conventions.

5.2 Emergence of Lanes

In this task, we analyze how perception noise and the number of training agents affects the emergence of lanes in an intersection. We relax the constraint on the agent paths in the environment setup of Section 5.1 and attempt to learn lane following and traffic signals jointly.

We define the spline subpolicy's observation space as the nodes of a path along the central axis of the roads connecting the start position to the destination. This observation is analogous to the GPS navigation maps used by human drivers. Given this path, the agent needs to predict a deviation from these nodes. We construct a Catmull-Rom Spline from these deviated points, and then the acceleration subpolicy controls the agent's position along this spline.

To empirically analyze the emergence of lanes, we plot the "Normalized Lane Position" of the agents over time. The magnitude of the "Normalized Lane Position" refers to the agent's deviation from the axis of the road, and the sign denotes if it is on the right side or left side in its local frame. Figure 4 shows the rollouts for agents trained on a 4-agent environment with varying LiDAR noise. We observe that increasing the LiDAR noise promotes distributions with lower variance. On increasing the number of agents from 4 (Figure 5), we observe the agents follow lanes more consistently and travel in multiple lanes, which further speeds up the traffic flow.

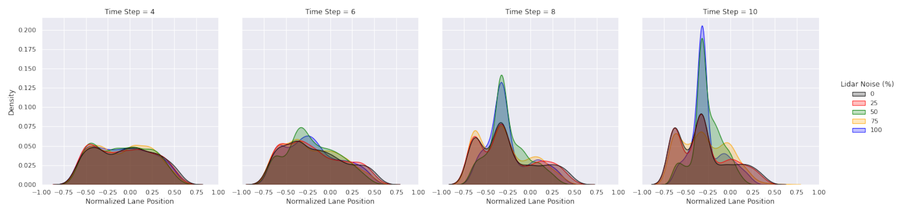

Figure 4. Lanes emerge with more perception noise Norm. Lane Pos. of Agents till they reach intersection.

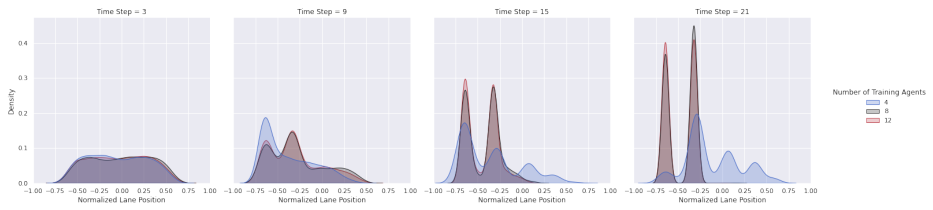

Figure 5. Lanes emerge with more agents Normalized Lane Position of Agents over time with varying number of agents during training

5.3 Right of Way

For tasks that can be performed simultaneously and take approximately equal time for completion, First In First Out (FIFO) scheduling strategy minimizes the average waiting time. In the context of driving through an intersection where each new agent symbolizes a new task, the agent that arrives first at the intersection should also be able to leave the intersection first. In other words, given any two vehicles, the vehicle arriving at the intersection first has the "right of way" over the other vehicle.

Let the time at which agent $i \in [n]$ arrives at the intersection be $(t_a)_i$ and leaves the intersection be $(t_d)_i$. If $\exists j \in [n] \setminus \{i\}$, such that $(t_a)_i < (t_a)_j$ and $(t_d)_i > (t_d)_j$, we say that agent i doesn't respect j's right of way. We evaluate this metric on a model trained on a nuScenes intersection (Figure 7). We observe that, at convergence, the agents follow this right of way 91.8% of time (Figure 6).

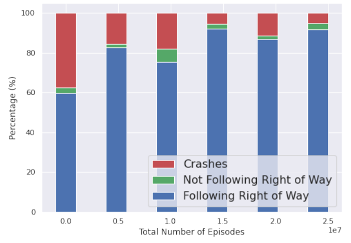

Figure 6. Right of way Agents that successfully reach their destination respect the right of way of other agents right from beginning. With time more agents are able to reach their destination while following this rule.

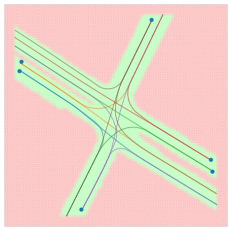

Figure 7. nuScenes Intersection used for Right of Way Evaluation (see Sec 5.3). Agents' starting positions are uniformly sampled from the trajectories shown.

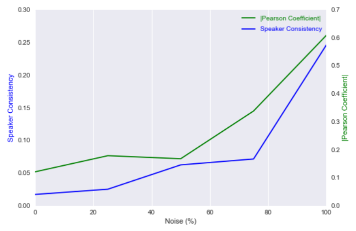

Figure 8. Communication increases with more perception noise Speaker Consistency & Pearson coefficient between the agent's heading and its sent message.

5.4 Communication

One way to safely traverse an intersection is to signal one's intention to nearby vehicles. In this task, we analyze the impact of perception noise on emergent communication at an intersection. In particular, we measure the Speaker Consistency (SC), proposed in Jaques et al. (2019). SC can be considered as the mutual information between an agent's message and its future action. We report the mutual information and the Pearson coefficient (Freedman et al., 2007) between the agent's heading and its sent message. For simplicity, we limit the communication channel to one bit and each car only receives signal from the car in the front within $-30°$ and $30°$. Fig 8 shows that agents rely more heavily on communication at intersections when perception becomes less reliable.

5.5 Fast Lanes on a Highway

There are often special dedicated lanes for faster cars to allow a smooth flow of traffic on highways/expressways. In this task, we want to see empirically if autonomous vehicles exhibit similar behavior of forming "fast" lanes while moving on a highway. We consider a highway with a unidirectional flow of traffic. Agents are spawned at random positions along the road's axis and are given a destination that they can reach by moving straight ahead.

Every agent is associated with a scalar value called "Acceleration Rating," which scales the agent's acceleration and velocity limits. Thus, a higher acceleration rating implies a faster car. Even though the agents can decide to move straight by design, it is not an optimal choice as slower cars in front will hinder smooth traffic flow.

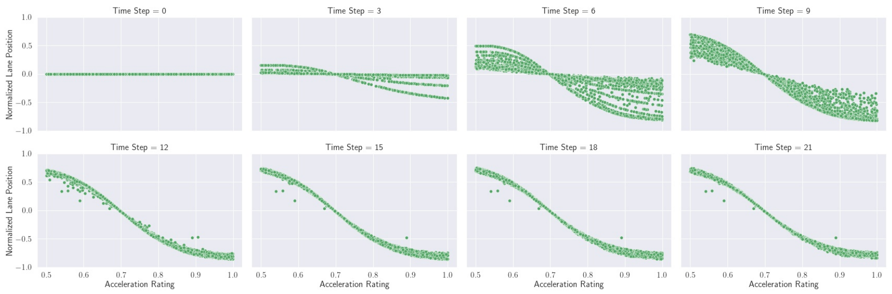

Figure 9. Fast Lane Emergence Visualization of rollouts from a 10-agent highway environment. In y-axis, we show the agent's position relative to the axis of the road normalized by road width. x-axis shows the acceleration rating, which scales the maximum acceleration and velocity of the agent.

In Figure 9, we visualize the rollouts generated from a 10-agent highway environment. We see that there is a clear segregation of lanes among the agents according to their acceleration ratings. In this particular case, the faster agents end up on the right-hand side lane, and as we go from right to left, the maximum acceleration/velocity of the agents goes down. This pattern ensures that slower vehicles do not obstruct faster vehicles once the traffic flow has reached a steady state.

5.6 Minimum Distance between Vehicle

In this task, we evaluate the presence of a minimum distance between agents while driving. In our simulator, we assume ideal conditions and agents can change their velocity according to $v^2 = u^2 + 2ad$, where v and u are the final and initial velocities respectively, a is acceleration, and d is the distance for which the acceleration remains constant. Hence, for an agent to stop entirely from a state with velocity v_0, it needs at least a distance of $\frac{v_0^2}{2a_{max}}$ in front of it, where a_{max} is the maximum possible deceleration of the agent.

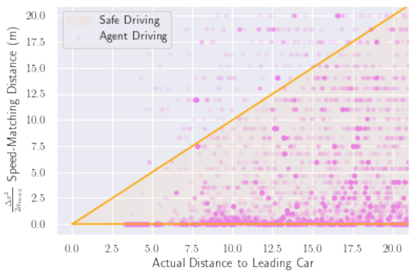

Figure 10. Safety Distance Maintained in Nuscenes Environment

Our agents perceive the environment through LiDAR; thus, agents can estimate the velocity and acceleration of nearby agents. We define the safe distance as the distance needed for a trailing agent to have a zero velocity in the leading agent's frame. We assume that the leading agent travels with a constant velocity, and as such, the safe distance is defined by $\frac{\Delta s^2}{2a_{max}}$, where Δs is the relative velocity. Any car having a distance greater than this can safely slow down. In Fig 10, we show the scatter plot for the safe distance the agents should maintain on the y-axis vs. the distance they maintain on x-axis. We observe that in most of the cases, agents do observe this safety distance.

5.7 Slowing down near a Crosswalk

In this task, we evaluate if agents can detect pedestrians and slow down in their presence. We augment the environment setup of Sec. 5.5, to include a crosswalk where at most 10 pedestrians are spawned at the start of every rollout. The pedestrians cross the road with a constant velocity. If any agent collides with a pedestrian they get a collision reward of -1 and the simulation for that agent stops.

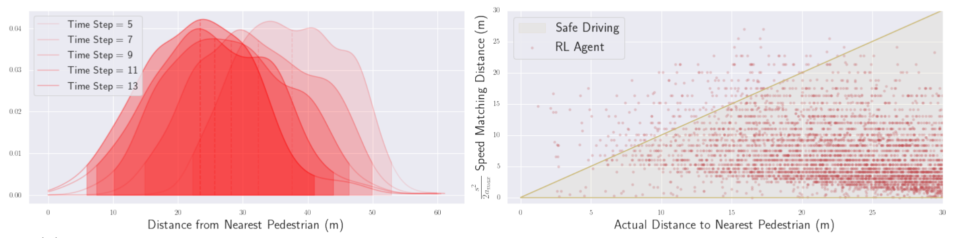

Figure 11. Distribution of the Distance of an agent from the nearest Pedestrian. The shaded region in the KDE plots indicate the 95% confidence interval and the dotted line is the sample mean

The KDE plots in Figure 11 show that the agents indeed detect the pedestrians and most of them maintain a distance greater than 6m. To determine if the agents can safely stop and prevent collision with the pedestrians, we calculate a safe stopping distance of $\frac{s^2}{2a_{max}}$, where s is the velocity of the agent. In the scatter plot, we observe that most agents adhere to this minimum distance and drive at a distance which lies in the safe driving region.

6 Discussion

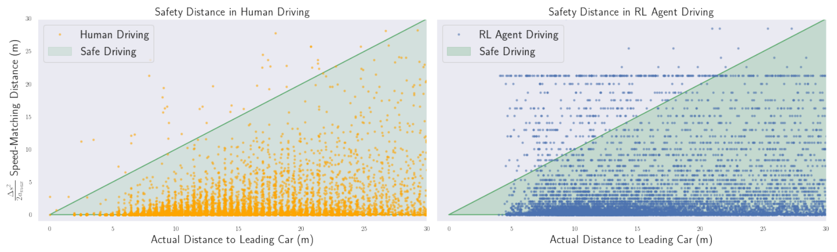

Figure 12. Safety Distance for Humans vs. Safety Distance for RL Agents The agents trained in our MDP (right) tend to violate the safety distance more than human drivers (left), but in both cases a safety distance is observed the vast majority of the time (green triangle region)

6.1 Statistics of Human Driving

In some cases, the same statistics accumulated over agent trajectories that we use in Section 5 to quantitatively demonstrate emergence can also be accumulated over the human driving trajectories labeled in the nuScenes dataset. In Figure 12, we visualize safety distance statistics across nuScenes trajectories and safety distance statistics across RL agents trained on nuScenes intersections side-by-side. The nuScenes trainval split contains 64386 car instance labels, each with an associated trajectory. For each location along the trajectory, we calculate the safety distance as described in Section 5.6. The same computation is performed over RL agent trajectories. The agents trained in our MDP tend to violate the safety distance more than human drivers, but in both cases a safety distance is observed the vast majority of the time (green triangle region).

6.2 Future Work

By parameterizing policies such that agents must follow the curve generated by the spline subpolicy at initialization (see Section 4.4), we prevent lane change behavior from emerging. The use of a more expressive action space should address this limitation at the cost of training time. Additionally, the fact that our reward is primarily based on agents reaching destinations means that convergence is slow on maps that are orders of magnitude larger than the dimensions of the vehicles. One possible solution to training agents to navigate in large maps would be to generate a curriculum of target destinations, as in Mirowski et al. (2018).

7 Conclusion

In this paper, we identify a lightweight multi-agent MDP that empirically captures the essential features of the driving problem. We equip our agents with a sparse LiDAR sensor and reward agents when they reach their assigned target destination as quickly as possible without colliding with other agents in the scene. We observe that agents in this setting rely on a shared notion of lanes and traffic lights in order to compensate for their noisy perception. We believe that dense multi-agent interaction and perception noise are critical ingredients in the design of any simulator that seeks to instill human-like road rules in self-driving agents.

REFERENCES

Joshua Achiam. Spinning Up in Deep Reinforcement Learning. 2018.

Adrian K Agogino and Kagan Tumer. Unifying temporal and structural credit assignment problems. 2004.

Bowen Baker, Ingmar Kanitscheider, Todor Markov, Yi Wu, Glenn Powell, Bob McGrew, and Igor Mordatch. Emergent tool use from multi-agent autocurricula. *arXiv preprint arXiv:1909.07528*, 2019.

Mayank Bansal, Alex Krizhevsky, and Abhijit S. Ogale. Chauffeurnet: Learning to drive by imitating the best and synthesizing the worst. *CoRR*, abs/1812.03079, 2018. URL `http://arxiv.org/abs/1812.03079`.

F. Belletti, Daniel Haziza, Gabriel Gomes, and A. Bayen. Expert level control of ramp metering based on multi-task deep reinforcement learning. *IEEE Transactions on Intelligent Transportation Systems*, 19:1198–1207, 2018.

Alex Bewley, Jessica Rigley, Yuxuan Liu, Jeffrey Hawke, Richard Shen, Vinh-Dieu Lam, and Alex Kendall. Learning to drive from simulation without real world labels. *CoRR*, abs/1812.03823, 2018. URL `http://arxiv.org/abs/1812.03823`.

Mariusz Bojarski, Davide Del Testa, Daniel Dworakowski, Bernhard Firner, Beat Flepp, Prasoon Goyal, Lawrence D. Jackel, Mathew Monfort, Urs Muller, Jiakai Zhang, Xin Zhang, Jake Zhao, and Karol Zieba. End to end learning for self-driving cars. *CoRR*, abs/1604.07316, 2016. URL `http://arxiv.org/abs/1604.07316`.

Lucian Busoniu, Robert Babuska, and Bart De Schutter. A comprehensive survey of multiagent reinforcement learning. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 38(2):156–172, 2008.

Holger Caesar, Varun Bankiti, Alex H. Lang, Sourabh Vora, Venice Erin Liong, Qiang Xu, Anush Krishnan, Yu Pan, Giancarlo Baldan, and Oscar Beijbom. nuScenes: A multimodal dataset for autonomous driving. *arXiv preprint arXiv:1903.11027*, 2019.

Edwin Catmull and Raphael Rom. A class of local interpolating splines. In *Computer aided geometric design*, pages 317–326. Elsevier, 1974.

Boyuan Chen, Shuran Song, Hod Lipson, and Carl Vondrick. Visual hide and seek, 2019.

Alexey Dosovitskiy, Germán Ros, Felipe Codevilla, Antonio M. López, and Vladlen Koltun. CARLA: an open urban driving simulator. *CoRR*, abs/1711.03938, 2017. URL `http://arxiv.org/abs/1711.03938`.

Jakob Foerster, Ioannis Alexandros Assael, Nando De Freitas, and Shimon Whiteson. Learning to communicate with deep multi-agent reinforcement learning. In *Advances in neural information processing systems*, pages 2137–2145, 2016.

David Freedman, Robert Pisani, and Roger Purves. Statistics (international student edition). *Pisani, R. Purves, 4th edn. WW Norton & Company, New York*, 2007.

Pablo Hernandez-Leal, Michael Kaisers, Tim Baarslag, and Enrique Munoz de Cote. A survey of learning in multiagent environments: Dealing with non-stationarity. *arXiv preprint arXiv:1707.09183*, 2017.

Pablo Hernandez-Leal, Bilal Kartal, and Matthew E. Taylor. A survey and critique of multiagent deep reinforcement learning. *Autonomous Agents and Multi-Agent Systems*, 33:750 – 797, 2019.

Max Jaderberg, Wojciech M Czarnecki, Iain Dunning, Luke Marris, Guy Lever, Antonio Garcia Castaneda, Charles Beattie, Neil C Rabinowitz, Ari S Morcos, Avraham Ruderman, et al. Human-level performance in 3d multiplayer games with population-based reinforcement learning. *Science*, 364(6443):859–865, 2019.

Natasha Jaques, Angeliki Lazaridou, Edward Hughes, Caglar Gulcehre, Pedro Ortega, DJ Strouse, Joel Z Leibo, and Nando De Freitas. Social influence as intrinsic motivation for multi-agent deep reinforcement learning. In *International Conference on Machine Learning*, pages 3040–3049. PMLR, 2019.

Joel Z. Leibo, Edward Hughes, Marc Lanctot, and Thore Graepel. Autocurricula and the emergence of innovation from social interaction: A manifesto for multi-agent intelligence research. *CoRR*, abs/1903.00742, 2019. URL `http://arxiv.org/abs/1903.00742`.

Siqi Liu, Guy Lever, Josh Merel, Saran Tunyasuvunakool, Nicolas Heess, and Thore Graepel. Emergent coordination through competition. *arXiv preprint arXiv:1902.07151*, 2019.

Ryan Lowe, Yi I Wu, Aviv Tamar, Jean Harb, OpenAI Pieter Abbeel, and Igor Mordatch. Multi-agent actor-critic for mixed cooperative-competitive environments. In *Advances in neural information processing systems*, pages 6379–6390, 2017.

Sivabalan Manivasagam, Shenlong Wang, Kelvin Wong, Wenyuan Zeng, Mikita Sazanovich, Shuhan Tan, Bin Yang, Wei-Chiu Ma, and Raquel Urtasun. Lidarsim: Realistic lidar simulation by leveraging the real world, 2020.

Piotr Mirowski, Matthew Koichi Grimes, Mateusz Malinowski, Karl Moritz Hermann, Keith Anderson, Denis Teplyashin, Karen Simonyan, Koray Kavukcuoglu, Andrew Zisserman, and Raia Hadsell. Learning to navigate in cities without a map. *CoRR*, abs/1804.00168, 2018. URL `http://arxiv.org/abs/1804.00168`.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin A. Riedmiller. Playing atari with deep reinforcement learning. *CoRR*, abs/1312.5602, 2013. URL `http://arxiv.org/abs/1312.5602`.

Frans A Oliehoek, Christopher Amato, et al. *A concise introduction to decentralized POMDPs*, volume 1. Springer, 2016.

OpenAI. Openai five. `https://blog.openai.com/openai-five/`, 2018.

P. Peng, Quan Yuan, Ying Wen, Y. Yang, Zhenkun Tang, Haitao Long, and Jun Wang. Multiagent bidirectionally-coordinated nets for learning to play starcraft combat games. *ArXiv*, abs/1703.10069, 2017.

Jonah Philion and Sanja Fidler. Lift, splat, shoot: Encoding images from arbitrary camera rigs by implicitly unprojecting to 3d. In *Proceedings of the European Conference on Computer Vision*, 2020.

Tabish Rashid, Mikayel Samvelyan, Christian Schroeder De Witt, Gregory Farquhar, Jakob Foerster, and Shimon Whiteson. Qmix: Monotonic value function factorisation for deep multi-agent reinforcement learning. *arXiv preprint arXiv:1803.11485*, 2018.

John Schulman, Philipp Moritz, Sergey Levine, Michael Jordan, and Pieter Abbeel. High-dimensional continuous control using generalized advantage estimation. *arXiv preprint arXiv:1506.02438*, 2015.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Alexander Sergeev and Mike Del Balso. Horovod: fast and easy distributed deep learning in TensorFlow. *arXiv preprint arXiv:1802.05799*, 2018.

Yoav Shoham, Rob Powers, and Trond Grenager. If multi-agent learning is the answer, what is the question? *Artificial intelligence*, 171(7):365–377, 2007.

Kyunghwan Son, Daewoo Kim, Wan Ju Kang, David Earl Hostallero, and Yung Yi. Qtran: Learning to factorize with transformation for cooperative multi-agent reinforcement learning. *arXiv preprint arXiv:1905.05408*, 2019.

Matt Stevens and Christopher Yeh. Reinforcement learning for traffic optimization. *Stanford. edu*, 2016.

Sainbayar Sukhbaatar, Rob Fergus, et al. Learning multiagent communication with backpropagation. In *Advances in neural information processing systems*, pages 2244–2252, 2016.

Richard S Sutton, David A McAllester, Satinder P Singh, and Yishay Mansour. Policy gradient methods for reinforcement learning with function approximation. In *Advances in neural information processing systems*, pages 1057–1063, 2000.

Tonghan Wang, Heng Dong, Victor Lesser, and Chongjie Zhang. Roma: Multi-agent reinforcement learning with emergent roles, 2020.

David H Wolpert and Kagan Tumer. Optimal payoff functions for members of collectives. In *Modeling complexity in economic and social systems*, pages 355–369. World Scientific, 2002.

Cathy Wu, Aboudy Kreidieh, Eugene Vinitsky, and Alexandre M. Bayen. Emergent behaviors in mixed-autonomy traffic. volume 78 of *Proceedings of Machine Learning Research*, pages 398–407. PMLR, 13–15 Nov 2017a. URL `http://proceedings.mlr.press/v78/wu17a.html`.

Cathy Wu, Aboudy Kreidieh, Eugene Vinitsky, and Alexandre M Bayen. Emergent behaviors in mixed-autonomy traffic. In *Conference on Robot Learning*, pages 398–407, 2017b.

Zhenpei Yang, Yuning Chai, Dragomir Anguelov, Yin Zhou, Pei Sun, Dumitru Erhan, Sean Rafferty, and Henrik Kretzschmar. Surfelgan: Synthesizing realistic sensor data for autonomous driving, 2020.

W. Zeng, W. Luo, S. Suo, A. Sadat, B. Yang, S. Casas, and R. Urtasun. End-to-end interpretable neural motion planner. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 8652–8661, 2019.