Broader Impacts

Safety-critical multi-agent cooperative policy learning has the potential to significantly benefit society by enhancing safety, efficiency, and coordination in various domains. In autonomous driving, robust multi-agent policies can prevent collisions, optimize traffic flow, and ensure pedestrian safety, contributing to safer urban mobility. In multi-drone systems, such as UAV swarms for search-and-rescue operations or environmental monitoring, agents can collaboratively navigate and perform tasks without risking collisions or environmental damage. Similarly, in industrial automation, multi-robot systems can coordinate safely in shared spaces, increasing productivity while reducing the risk of accidents.

However, the deployment of safety-critical multi-agent systems also introduces potential negative societal impacts. Poorly designed safety mechanisms can result in unsafe behaviors, leading to accidents or property damage. Our work addresses this issue by proposing a safety-guaranteed approach that ensures safety during both training and deployment phases, leveraging Control Barrier Functions (CBFs) to enforce strict safety constraints on agent behavior.

A Appendix / supplemental material

A.1 Environment Details

The five environments from the METADRIVE simulator are illustrated in the figure 4. As mentioned these environments involve regions where agents can cross path with other agents, Hence, it is necessary for the agents to cooperate but at the safe time stay safe with each other to achieve the task i.e. reach destination successfully and timely.

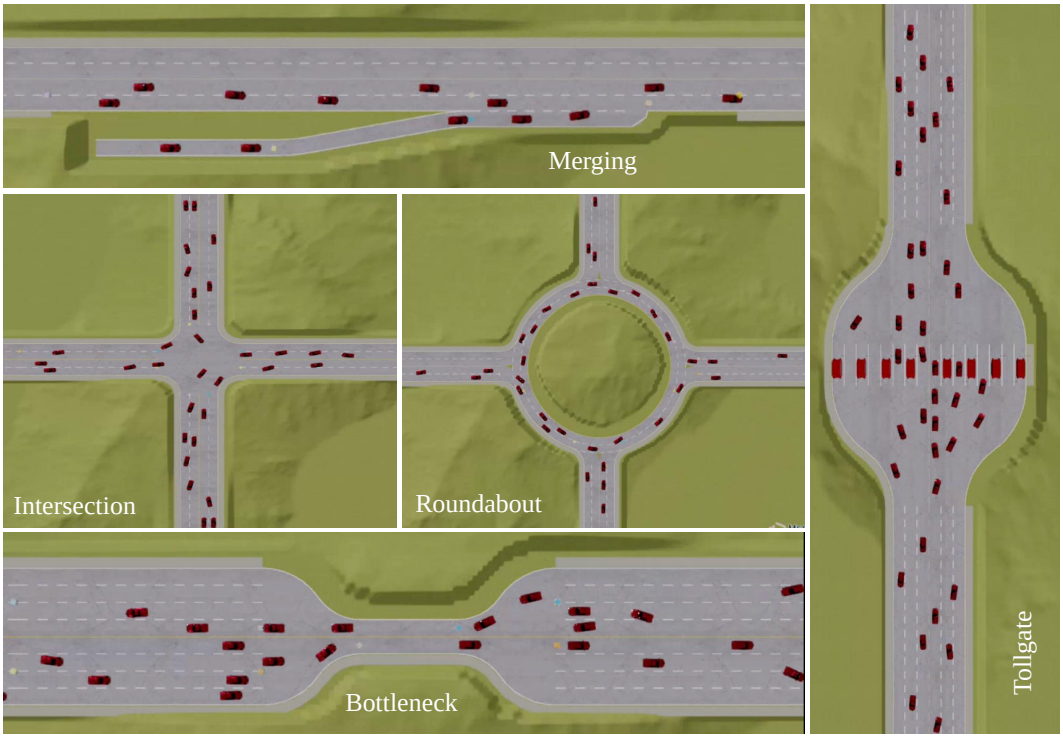
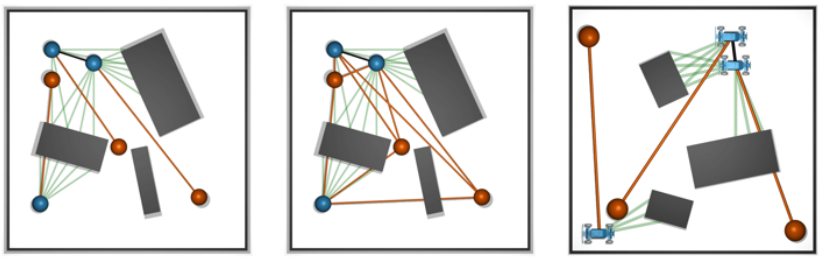


Figure 4: Illustration of the various METADRIVE environments.



(a) Target (b) Spread (c) Target bicycle

Figure 5: Illustration of the lidar based environments which include: (a) target environment, (b) spread environment and (c) target bicycle environment.

A.2 Experimental Details and Additional Results

We begin this section by presenting the details of the training, evaluation and ablation experiments. Besides that, we include results for the generalization experiments used to validate our proposed method.

In addition, we evaluate our method on a suite of lidar-based environments introduced in [51], as illustrated in Figure 5. All environments are partially observable, with agents equipped with lidar sensors. In the *Target* environments, each agent is assigned a fixed goal and must reach it while avoiding collisions with other agents and obstacles. The primary distinction between the *Target* and *Target-Bicycle* environments lies in the underlying agent dynamics; further details are provided in Figure 5. In the *Spread* environment, agents must cooperate to cover all goal locations while avoiding both obstacles and one another.

A.2.1 Training Setup and Results.

In all five environments used in our experiments, the vehicles are initialized at random spawn points, and assigned destinations to each of them randomly. The agents are terminated in three situations: reaching the destinations (deemed as successful), crashing with others or driving out of the road (deemed as failure). Once an existing vehicle is terminated new vehicles are spawned immediately to ensure continuous traffic flow. The terminated vehicles remains static for 10 steps (2s) in the original positions, creating impermeable obstacles. If a vehicle crashes into other terminated vehicles before it is removed from the environment, it is also considered as failure. The *success rate* is the fraction of the successful agents to the total number of spawned agents. Thus, the environment allows for the total number of vehicles to vary in time and exceed the the initial number of agents which mimics real-world challenging traffic scenarios.

We trained with 15 vehicles for merging, 30 vehicles for intersection and roundabout, 25 vehicles for bottleneck and 40 vehicles for tollgate respectively. As mentioned previously, the extrinsic reward corresponding to (9) was set to the environment reward, besides the regularization terms. The intrinsic reward is described in subsection A.5. We present the training results for the remaining three environments from figure 3 i.e. merging, roundabout and intersection in figure 6.

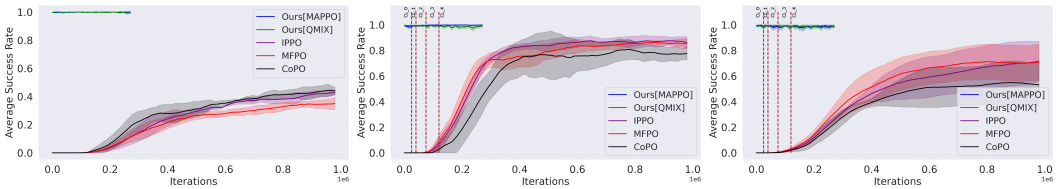


Figure 6: Plot of the success rate of the agents with our algorithm vs the baselines in merging, roundabout, and intersection environments respectively. CL refers to curriculum learning, four different curriculum were used with varying traffic densities. The benchmark methods use 2000+ hours of individual driving data prior to the training.

As can be noticed, our algorithm achieves a near perfect safety rate with both on-policy and off-policy algorithms during training, and, keeping consistent with the tollgate and bottleneck environments are trained for 300k iterations. However, in order to encourage exploration in roundabout and intersection environments for learning the cooperative behavior, we use curriculum learning by varying the arrival rates, while keeping the traffic density constant. The curriculum transitions happened at 30k, 50k, 80k and 125k iterations during training.

A.2.2 Evaluation Setup and Results

METADRIVE Simulator. As mentioned previously we used *success rate weighted average time and energy* for evaluating the performance of our algorithms. In order to compare against the baseline methods, for each environment and algorithm (e.g. on-policy, on-policy w/Grouping, off-policy and so on) we train using five seeds. Then we run five episodes for each seed and generate the evaluation results by computing the mean and standard deviation over the 25 runs. In addition to the presented results in table 2, we run the baseline algorithms by changing the penalty factors for the safety related terms (both inter-agent and environment safety) in the environment reward for the bottleneck environment. The results are presented in table 3. The primary takeaway is that the added emphasis on safety does not render better results for the baseline algorithms compared to our method. Besides that, different algorithms perform differently with increase in penalty factor, thus not exhibiting any trend between penalty and success rate.

Table 3: Summary of baseline comparison using various penalty factors for Bottleneck environment. ↑: higher is better; ↓: lower is better. **Bold:** the best performance for each metric.

Penalty Factors	Metrics	IPPO	CL	MFPO	CoPo	**HMARL-CBF (on-policy)**
1	Success ↑	$0.09_{\pm 0.03}$	$0.34_{\pm 0.15}$	$0.48_{\pm 0.16}$	$0.75_{\pm 0.07}$	$\mathbf{0.99_{\pm 0.00}}$
	Time ↓	$2644.44_{\pm 353.44}$	$939.35_{\pm 147.21}$	$679.83_{\pm 158.33}$	$651.54_{\pm 63.52}$	$\mathbf{241.16_{\pm 8.33}}$
	Energy ↓	$69.66_{\pm 9.77}$	$11.61_{\pm 2.02}$	$9.95_{\pm 1.39}$	$6.93_{\pm 0.61}$	$\mathbf{7.37_{\pm 0.00}}$
5	Success ↑	$0.58_{\pm 0.11}$	$0.395_{\pm 0.01}$	$0.615_{\pm 0.08}$	$0.716_{\pm 0.02}$	$\mathbf{0.99_{\pm 0.00}}$
	Time ↓	$568_{\pm 56}$	$646.5_{\pm 43.16}$	$713_{\pm 273.7}$	$529.13_{\pm 79}$	$\mathbf{241.16_{\pm 8.33}}$
	Energy ↓	$30_{\pm 41.5}$	$10.8_{\pm 0.24}$	$8.64_{\pm 0.65}$	$7.69_{\pm 0.02}$	$\mathbf{7.37_{\pm 0.00}}$
10	Success ↑	$0.18_{\pm 0.08}$	$0.55_{\pm 0.11}$	$0.42_{\pm 0.23}$	$0.655_{\pm 0.04}$	$\mathbf{0.99_{\pm 0.00}}$
	Time ↓	$1292.17_{\pm 647}$	$606.49_{\pm 204.89}$	$1414_{\pm 973}$	$514_{\pm 126.92}$	$\mathbf{241.16_{\pm 8.33}}$
	Energy ↓	$40.97_{\pm 27}$	$9.14_{\pm 0.93}$	$22.12_{\pm 22.08}$	$8.37_{\pm 0.04}$	$\mathbf{7.37_{\pm 0.00}}$

Lidar Environment Suite.

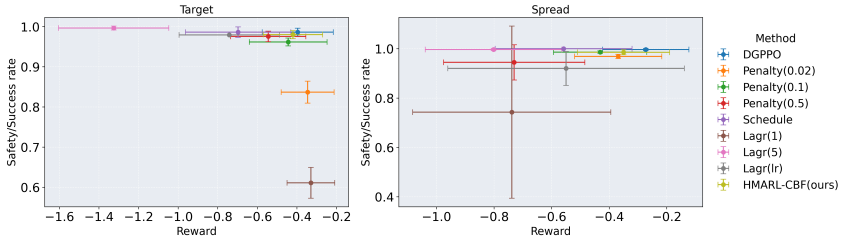


Figure 7: Success/safety rate vs reward for Target and Spread environment.

Baselines. We compare our method with DGPPO [51], InforMARL [71] and MAPPO-Lagrangian [72], three state of the art methods for multi-agent safe control. For InforMARL, we evaluate fixed and scheduled penalty variants. Further details about the baselines can be found in [51]. For MAPPO-Lagrangian, we test two settings for the Lagrange multiplier and one learning rate. Performance is better if the plot is located in top right corner. As can be seen, our method achieves a high success rate while also achieving high reward compared to the baselines in [51]. Finally, we present the results for the bicycle target environment in 8 which shows that our method scales with number of agents.

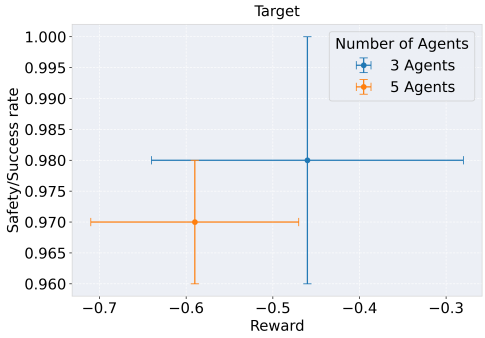


Figure 8: Success/safety rate vs reward for Target and Spread environment.

A.2.3 Generalization Results

Besides the evaluation experiments, we conducted a series of experiments to assess the generalizability of our proposed approach. Firstly we systematically vary the lane widths in the environments. The corresponding success rates are reported in Table 4. Our analysis indicates that reducing lane width has a minimal impact on success rates in the roundabout and bottleneck environments. However, a more comparatively greater decline is observed in the intersection environment. This performance degradation is primarily attributed to the default intersection configuration (Figure 1), where two lanes are available in each direction. Agents are trained to execute lane changes from both lanes depending on their assigned routes. When lane width is reduced, agents attempting to perform lane changing from the inside lane at times gets stuck (and, eventually get eliminated from the episode due to timeout), or, hit road boundary which causes the drop in success rate. Conversely, increasing lane width generally maintains or enhances success rates across all environments.

Secondly, we investigated skill transferability by training low-level skills in an intersection environment and subsequently using them to train a high-level policy across multiple environments. Namely, we investigate the transferability in the merging and roundabout environments. This is because the observation is different and distinct for the Bottleneck and Tollgate environments compared to the other environments. The results are presented in table 5. This analysis reveals that skills learned in one environment can generalize effectively to others producing comparatively similar performance. In the roundabout environment, we observe a minor decline in the success rate and increase in the weighted average energy and time; yet our approach outperforms the baselines presented in table 2.

Finally, we test the generalizability of our method for varying traffic density/ number of agents. The results for this test are presented in tables 6 and 7 respectively. For every environment we trained for the default density as shown in the table and tested for lower and higher values. From the results, we can conclude that the success rate is unaffected by traffic density. The average time of the agents increases with increasing traffic density as expected. Finally, the energy values show a decreasing trend with increase in traffic density across most environments. This was anticipated because energy has a negative correlation with time.

A.2.4 Computer Details.

We conducted our experiments on a desktop equipped with a single NVIDIA RTX A5000 GPU and 32 CPU cores. Given that our environments involve a high number of agents (up to 45 agents in the tollgate environment), the primary computational cost is attributed to data collection. Training and convergence are achieved efficiently, as demonstrated in the training and evaluation results. Furthermore, we utilize shared neural network architectures across agents, which significantly reduces the overall computational cost and training time.

A.3 Background

Definition A.1 (Relative degree)**.** The relative degree of a (sufficiently many times) differentiable function $b : \mathbb{R}^n \to \mathbb{R}$ with respect to system (3) is the number of times it needs to be differentiated along its dynamics until the control $\boldsymbol{a}$ explicitly appears in the corresponding derivative.

Table 4: Success rate across different environments for varying lane widths.

Lane Width (m)	Success Rate				
	Merging	**Intersection**	**Roundabout**	**Bottleneck**	**Tollgate**
3.25	$0.99_{\pm 0.01}$	$0.90_{\pm 0.03}$	$0.98_{\pm 0.02}$	$0.97_{\pm 0.02}$	$0.99_{\pm 0.001}$
3.5 (default)	$0.99_{\pm 0.01}$	$0.97_{\pm 0.04}$	$0.99_{\pm 0.01}$	$0.99_{\pm 0.003}$	$0.99_{\pm 0.02}$
4	$0.99_{\pm 0.01}$	$0.97_{\pm 0.015}$	$0.99_{\pm 0.01}$	$0.99_{\pm 0.001}$	$0.99_{\pm 0.02}$

Table 5: Summary of skill transfer results (* indicates runs done by transferring skills).

Experiments	**Success**	**Energy**	**Time**
Merging	$0.99_{\pm 0.01}$	$14.24_{\pm 0.23}$	$166.18_{\pm 2.64}$
Merging*	$0.96_{\pm 0.016}$	$14.49_{\pm 0.58}$	$178.34_{\pm 5.90}$
Roundabout	$0.99_{\pm 0.01}$	$9.89_{\pm 0.23}$	$403.18_{\pm 21.66}$
Roundabout*	$0.96_{\pm 0.016}$	$11.39_{\pm 0.08}$	$404.16_{\pm 18.6}$

Table 6: Summary of results for varying number of agents in Merging, Intersection, and Roundabout environments.

Metric	**Merging**			**Intersection**			**Roundabout**		
	10	15 (default)	20	20	30 (default)	40	20	30 (default)	40
Success	$0.997_{\pm 0.0}$	$0.99_{\pm 0.01}$	$0.98_{\pm 0.02}$	$0.955_{\pm 0.02}$	$0.97_{\pm 0.04}$	$0.94_{\pm 0.04}$	$0.99_{\pm 0.01}$	$0.99_{\pm 0.0}$	$0.98_{\pm 0.0}$
Energy	$14.03_{\pm 0.36}$	$14.24_{\pm 0.23}$	$14.47_{\pm 0.18}$	$5.85_{\pm 0.07}$	$5.82_{\pm 0.04}$	$5.94_{\pm 0.05}$	10.48	10.32	9.95
Time	$165.18_{\pm 2.15}$	$166.18_{\pm 2.64}$	$169.55_{\pm 2.15}$	$318.54_{\pm 19}$	$332.3_{\pm 36}$	$361.8_{\pm 20.46}$	389.34	404.38	415.03

Table 7: Summary of results for varying number of agents in the Bottleneck and Tollgate environments.

Metric	**Bottleneck**			**Tollgate**		
	20	25 (default)	30	30	40 (default)	45
Success	$0.99_{\pm 0.01}$	$0.99_{\pm 0.003}$	$0.978_{\pm 0.02}$	$0.99_{\pm 0.02}$	$0.99_{\pm 0.02}$	$0.99_{\pm 0.01}$
Energy	$7.47_{\pm 0.03}$	$7.44_{\pm 0.001}$	$7.43_{\pm 0.02}$	$8.51_{\pm 0.02}$	$8.51_{\pm 0.048}$	$8.50_{\pm 0.02}$
Time	$227.22_{\pm 7.77}$	$243.59_{\pm 8.33}$	$244.62_{\pm 6.62}$	$371.53_{\pm 6.78}$	$371.05_{\pm 5.82}$	$374.02_{\pm 8.48}$

For a constraint $b(\boldsymbol{s}) \geq 0$ with relative degree m, $b : \mathbb{R}^n \to \mathbb{R}$, and $\zeta_0(\boldsymbol{s}) := b(\boldsymbol{s})$, we define a sequence of functions $\zeta_i : \mathbb{R}^n \to \mathbb{R}, i \in \{1, \ldots, m\}$:

$$\zeta_i(\boldsymbol{s}) := \dot{\zeta}_{i-1}(\boldsymbol{s}) + \alpha_i\left(\zeta_{i-1}(\boldsymbol{s})\right), \; i \in \{1, \ldots, m\}, \tag{16}$$

where $\alpha_i(), \; i \in \{1, \ldots, m\}$ denotes a $(m-i)^{\text{th}}$ order differentiable class $\mathcal{K}$ function. We further define a sequence of sets $C_i, \; i \in \{1, ..., m\}$ associated with (16) which take the following form,

$$C_i := \{\boldsymbol{s} \in \mathbb{R}^n : \zeta_{i-1}(\boldsymbol{s}) \geq 0\}, \; i \in \{1, ..., m\}. \tag{17}$$

Definition A.2 (High Order CBF (HOCBF) [73, 74])**.** Let $C_1, ..., C_m$ be defined by (17) and $\zeta_1(\boldsymbol{s}), ..., \zeta_m(\boldsymbol{s})$ be defined by (16). A function $b : \mathbb{R}^n \to \mathbb{R}$ is a High Order Control Barrier Function (HOCBF) of relative degree m for system (3) if there exists $(m-i)^{th}$ order differentiable class $\mathcal{K}$ functions $\alpha_i, \; i \in \{1, ..., m-1\}$ and a class $\mathcal{K}$ function α_m such that

$$\sup_{\boldsymbol{a} \in \mathcal{U}} [L_f^m b(\boldsymbol{s}) + L_g L_f^{m-1} b(\boldsymbol{s})\boldsymbol{a} + \Omega(b(\boldsymbol{s})) + \alpha_m(\zeta_{m-1}(\boldsymbol{s}))] \geq 0 \tag{18}$$

for all $\boldsymbol{s} \in \bigcap_{i=1}^m C_i$. In (18), L_f^m and L_g denotes derivative along f and g m times and one time respectively, and $S(.)$ denotes the remaining Lie derivative along f with degree less than or equal to $m-1$ (omitted for simplicity, see [73]).

Note that the HOCBF in (18) is a general form of the degree one CBF [54] ($m = 1$) and exponential CBF in [75]. The following theorem on HOCBFs implies the forward invariance property of the CBFs and the original safety set. The proof is omitted (see [73] for the proof).

Definition A.3 (Control Lyapunov function (CLF)[54])**.** A continuously differentiable function $V : \mathbb{R}^n \to \mathbb{R}$ is a globally and exponentially stabilizing CLF for (3) if there exists constants $c_i \in \mathbb{R}_{>0}$, $i = 1, 2$, such that $c_1\|\boldsymbol{s}\|^2 \leq V(\boldsymbol{s}) \leq c_2\|\boldsymbol{s}\|^2$, and the following inequality holds

$$\inf_{\boldsymbol{a}\in\mathcal{U}}\left[L_f V(\boldsymbol{s}) + L_g V(\boldsymbol{s})a + \eta(\boldsymbol{s})\right] \leq e, \tag{19}$$

where e makes this a soft constraint. If the dynamics are of the form: $\dot{\boldsymbol{s}} = f(\boldsymbol{s}, \boldsymbol{a})$, we can express it in the form:

$$\dot{\boldsymbol{s}}' = f'(\boldsymbol{s}', \boldsymbol{a}) + g'(\boldsymbol{s}')\boldsymbol{a}' \tag{20}$$

where $\boldsymbol{s}' = \begin{bmatrix}\boldsymbol{s}\\ \boldsymbol{a}\end{bmatrix}$ and $\boldsymbol{a}'$ are the augmented system states and new primitive actions of the system respectively, $f' = \begin{bmatrix} f(\boldsymbol{s},\boldsymbol{a})\\ \mathbf{0}_{\dim(\boldsymbol{a})}\end{bmatrix}$, and $g' = \begin{bmatrix}\mathbf{0}_{\dim(\boldsymbol{s})\times\dim(\boldsymbol{a}')}\\ \boldsymbol{B}\end{bmatrix}$, and $B \in \mathbb{R}^{\dim(\boldsymbol{a})\times\dim(\boldsymbol{a}')}$. Generally, B contains ones in its diagonal entries. The redefined dynamics can be used subsequently for defining CBFs.

A.4 Safe Skill learning

We provide the details about the computation of policy gradients necessary to learn the skill policy parameters in this subsection. We follow [76] by applying the Karush-Kuhn-Tucker condition on the Lagrangian to derive the gradient of the state feedback policy with respect to its parameters. Specifically, let λ denote the dual variables on the HOCBF and CLF constraints, let $D(\cdot)$ create a diagonal matrix from a vector, and let $\boldsymbol{\mu}^*, \lambda^*$ denote the optimal solutions of $\boldsymbol{\mu}$ and λ, respectively. We can then write $d_{\boldsymbol{\mu}}$ and d_λ in the form:

$$\begin{bmatrix} d_\mu \\ d_\lambda \end{bmatrix} = \begin{bmatrix} H & G^T D(\lambda^*) \\ G & D(G\boldsymbol{a}^* - h) \end{bmatrix}^{-1} \begin{bmatrix} \mathbf{1}_{\dim(\mathcal{A}_i)} \\ 0 \end{bmatrix} \tag{21}$$

where G, h are concatenated by $G_b^j, G_v^j, h_b^j, h_v^j$,

$$\begin{aligned}
G_b^j &= -L_g L_f^{m-1} b_i^j(\boldsymbol{s}^i, \boldsymbol{s}^j);\ \ j \in \mathcal{N}^i(\boldsymbol{s}) \\
G_v^j &= -L_g V_i^j(\boldsymbol{s}^i, \boldsymbol{s}^j; \boldsymbol{\phi}_v^{i,j});\ \ j = 1, 2, \ldots \\
h_b^j &= L_f^m b_i^j(\boldsymbol{s}^i, \boldsymbol{s}^j) + \Omega\left(b_j^i(\boldsymbol{s}^i, \boldsymbol{s}^j); \boldsymbol{\phi}_b^{i,j}\right) + \alpha_m\left(\zeta_{m-1}\left(\boldsymbol{s}^i, \boldsymbol{s}^j; \boldsymbol{\phi}_b^{i,j}\right)\right); j \in \mathcal{N}^i(\boldsymbol{s}) \\
h_b^j &= L_f V_i^j(\boldsymbol{s}^i, \boldsymbol{s}^j) + \eta(\boldsymbol{s}^i, \boldsymbol{s}^j; \boldsymbol{\phi}_v^{i,j});\ \ j = 1, 2, \ldots
\end{aligned} \tag{22}$$

As the control bounds in (14) are not trainable, they are not included in G and h. Then, the relevant gradients with respect to all the parameters are given by

$$\nabla_H \boldsymbol{\mu} = \frac{1}{2}\left(d_{\boldsymbol{u}}\boldsymbol{u}^T + \boldsymbol{u}d_{\boldsymbol{u}}^T\right), \nabla_F \boldsymbol{\mu} = d_{\boldsymbol{u}} \tag{23}$$

$$\nabla_G \boldsymbol{\mu} = D(\lambda^*)\left(d_\lambda \boldsymbol{u}^T + \lambda d_{\boldsymbol{u}}^T\right), \nabla_h \boldsymbol{\mu} = -D(\lambda^*)\, d_\lambda \tag{24}$$

With the hierarchical composition of the policies $(\pi_H \circ \pi_{\boldsymbol{z}})$, we define the augmented state $\hat{\boldsymbol{s}} = [\boldsymbol{s}^i, \boldsymbol{s}^{-i}, \boldsymbol{z}, \boldsymbol{s}_{\boldsymbol{z},init}]$ where $\boldsymbol{s}^{-i}$ is the stage of the agents other than agent i and $\boldsymbol{s}_{\boldsymbol{z},init}$ is the joint state of the agents at the time of the initiation of the joint skill $\boldsymbol{z}$. The initiation state is included as the joint termination of the skill depends on the initiation state. The state and action value functions for agent i corresponding to reward r_L^i, skill policy π_{z^i} (given the high level policy π_H and the skill policy other agents $\pi_{\boldsymbol{z}^{-i}}$) are defined as follows:

$$\begin{aligned}
V^{\pi_{z^i}}(\hat{\boldsymbol{s}}) &= \mathbb{E}[r_L^i(\hat{\boldsymbol{s}}_0, \boldsymbol{a}_0^i | \hat{\boldsymbol{s}}_0 = \hat{\boldsymbol{s}}) + \mathbb{E}_{\hat{\boldsymbol{s}}_1}[Q(\hat{\boldsymbol{s}}_1, \boldsymbol{a}_1^i)]] \\
&= \mathbb{E}[r_L^i(\boldsymbol{s}_0^i, \boldsymbol{a}_0^i | \boldsymbol{s}_0^i = \boldsymbol{s}^i, z^i) + \mathbb{E}_{\hat{\boldsymbol{s}}_1}[Q(\hat{\boldsymbol{s}}_1, \boldsymbol{a}_1^i)]]
\end{aligned} \tag{25}$$

The state-action value function for agent i, given the hierarchical policy composition $\pi_H \circ \pi_{\boldsymbol{z}}$, is defined as:

$$Q^{\pi_{z^i}}(\hat{\boldsymbol{s}}, \boldsymbol{a}^i) = \mathbb{E}\left[r_L^i(\hat{\boldsymbol{s}}, \boldsymbol{a}^i) + \gamma\, \mathbb{E}_{\hat{\boldsymbol{s}}'}\left[V^{\pi_{z^i}}(\hat{\boldsymbol{s}}')\right]\right] \tag{26}$$

The corresponding advantage function is given by:

$$A^{\pi_{z^i}}(\hat{\boldsymbol{s}}, \boldsymbol{a}^i) = Q^{\pi_{z^i}}(\hat{\boldsymbol{s}}, \boldsymbol{a}^i) - V^{\pi_{z^i}}(\hat{\boldsymbol{s}}) \tag{27}$$

Based on the above definition of value functions and advantage function, as well the gradient of QP controller output with respect to its parameters, standard on-policy and off-policy algorithms can be used to derive the policy gradient to learn the controller parameters.

Proof of Theorem 5.1

Proof. Given, the constraints b_i^j's are satisfied at the initial time, the satisfaction of the CBF constraints makes the corresponding safety sets forward invariant per Theorem (3.4), thus guaranteeing their satisfaction at all future times. This makes the policies corresponding to the skills of the agents safe, thus, guaranteeing the safety of our proposed HMARL approach. □

High-level task specific reward to low level learning. The low-level policies should be such that they optimize the entire trajectory given by the problem in (5). Hence, in addition to introducing intrinsic motivation through the low-level reward, inspired by [23], we introduce an advantage term to the low-level reward resulting in the following revised low-level reward:

$$\tilde{r}_L^i(\boldsymbol{s}_t^i, \boldsymbol{a}_t^i | z_{t'}^i) = \lambda \frac{A^{\pi_H}(\boldsymbol{s}_{t'}, \boldsymbol{z}_{t'})}{\mathcal{N}} + (1-\lambda) r_L^i(\boldsymbol{s}_t^i, \boldsymbol{a}_t^i | z_{t'}^i) \tag{28}$$

where $\lambda \in [0,1]$ and $t' \leq t$ is the time when the joint skill at time t was initiated. The high-level advantage $A^{\pi_H}(\boldsymbol{s}_{t'}, \boldsymbol{a}_{t'})$ is defined as below:

$$A^{\pi_H}(\boldsymbol{s}_{t'}, \boldsymbol{z}_{t'}) = Q^{\pi_H}(\boldsymbol{s}_{t'}, \boldsymbol{z}_{t'}) - V^{\pi_H}(\boldsymbol{s}_{t'}) \tag{29}$$

where $Q^{\pi_H}(\boldsymbol{s}_{t'} \boldsymbol{z}_{t'})$ and $V^{\pi_H}(\boldsymbol{s}_{t'})$ are as defined in (2) and (1) respectively. The parameter λ can be used to balance between skill learning and optimization of the joint behavior of the agents. From [23] we have thatfor high level policies the following holds:

$$J(\tilde{\pi}_H) = J(\pi_H) + \mathbb{E}_{\boldsymbol{s}_{t'}, \boldsymbol{z}_{t'} \sim \pi_H} \left[\sum_{t'=0} \gamma^{t'} A^{\pi_H}(\boldsymbol{s}_{t'}, \boldsymbol{z}_{t'}) \right] \tag{30}$$

Proposition A.4. *Given the low level reward in* (28)*, the revised low level learning objective $\tilde{J}^i$ for agent i can be expressed as:*

$$\tilde{J}^i(\pi_H, \pi_{\boldsymbol{z}^{-i}}, \pi_{z^i}) \approx \frac{\lambda}{\mathcal{N}(1-\gamma)} \mathbb{E}\left[\sum_{t'=0} \gamma^{t'} A^{\pi_H}(\boldsymbol{s}_{t'}, \boldsymbol{z}_{t'}) \right] + (1-\lambda) J^i(\pi_H, \pi_{\boldsymbol{z}^{-i}}, \pi_{z^i}) \tag{31}$$

Proof. Based on 28, we can write our low level policy's learning objective for agent i as the following:

$$\begin{aligned}
\tilde{J}^i(\pi_H, \pi_{\boldsymbol{z}^{-i}}, \pi_{z^i}) &= \mathbb{E}\left[\sum_{t'=0} \mathbb{E}_{\boldsymbol{s}_t^i, \boldsymbol{a}_t^i \sim \pi_{z_{t'}^i}} \left[\sum_{t=0}^{k_{t'}-1} \gamma^{t+k_{t'}} \tilde{r}_L^i(\boldsymbol{s}_t^i, \boldsymbol{a}_t^i | z_{t'}^i) \right] \right] \\
&= \mathbb{E}\left[\sum_{t'=0} \mathbb{E}_{\boldsymbol{s}_t^i, \boldsymbol{a}_t^i \sim \pi_{z_{t'}^i}} \left[\sum_{t=0}^{k_{t'}-1} \gamma^{t+k_{t'}} \left(\lambda \frac{A^{\pi_H}(\boldsymbol{s}_{t'}, \boldsymbol{z}_{t'})}{\mathcal{N}} + (1-\lambda) \tilde{r}_L^i(\boldsymbol{s}_t^i, \boldsymbol{a}_t^i | z_{t'}^i) \right) \right] \right] \\
&= \frac{\lambda}{\mathcal{N}} \mathbb{E}\left[\sum_{t'=0} \frac{\gamma^{t'}(1-\gamma^{k_{t'}})}{1-\gamma} A^{\pi_H}(\boldsymbol{s}_{t'}, \boldsymbol{z}_{t'}) \right] + (1-\lambda) \mathbb{E}\left[\sum_{t'=0} \mathbb{E}_{\boldsymbol{s}_t^i, \boldsymbol{a}_t^i \sim \pi_{z_{t'}^i}} \left[\sum_{t=0}^{k_{t'}-1} \gamma^{t+k_{t'}} r_{L_A}^i(\boldsymbol{s}_t^i, \boldsymbol{a}_t^i | z_{t'}^i) \right] \right] \\
&\approx \frac{\lambda}{\mathcal{N}(1-\gamma)} \mathbb{E}\left[\sum_{t'=0} \gamma^{t'} A^{\pi_H}(\boldsymbol{s}_{t'}, \boldsymbol{z}_{t'}) \right] + (1-\lambda) J^i(\pi_H, \pi_{\boldsymbol{z}^{-i}}, \pi_{z^i})
\end{aligned}$$

□

Similarly, the low level reward for every agent can be revised as above to align skill learning process with the task learning process, as the first term is related to the high level policy return.

A.5 Algorithm Implementation

A.5.1 Algorithm details

As mentioned previously, our method can be implemented using both existing on-policy and off-policy RL algorithms. The bilevel RL problems presented in (10) and (15), respectively, should be solved sequentially. However, as previous works illustrated [77], jointly optimizing both policies by updating their corresponding parameters alternatively renders similar performances, which is consistent with our observation from the experiments. Based on this observation, the algorithm is presented in (1). The high-level policy $\pi_{H_{\boldsymbol{\theta}}}$ is decomposed into agent-level parameterized policies $\pi^i_{H_{\boldsymbol{\theta}^i}}$ to learn decentralized policies.

Algorithm 1 Hierarchical Multi-Agent Reinforcement Learning with CBFs (On-Policy)

Initialize randomly the high-level policy parameters $\boldsymbol{\theta}$ and parameters $\boldsymbol{\nu}$ of the low-level policy parameters $\boldsymbol{\phi}$ //network parameter sharing done between agents.
Set $Grouping$ flag.
for each episode **do**
  $z^i_0 \sim \pi_{H_{\boldsymbol{\theta}}}(z^i|\boldsymbol{o}^i_0),\ \forall i \in \mathcal{N}$ // sample initial skill
  Set data buffer $\mathcal{D} = \{\}$
  for $t \leftarrow 0, \ldots, T$ **do**
    Get action for agent i based on the selected skill $a^i_t = \mu^i(\boldsymbol{o}^i_t|z^i_{t'}; \boldsymbol{\phi_\nu})\ \forall i \in \mathcal{N}$ // sample primitive action
    Get s_{t+1} and r_H from environment and $r^i_L \forall i \in \mathcal{N}$
    Sample skill for agent i, $z^i_{t+1} \sim \pi^i_{H_{\boldsymbol{\theta}}}(z^i|\boldsymbol{o}^i_t),\ \forall i \in \mathcal{N}_z(t)$ // $\mathcal{N}_z(t)$ is the list of agents who have to update their skills based on termination scheme used.
    if $\sim Grouping$ **then**
      $\mathcal{D} = \mathcal{D} \cup \{\boldsymbol{s}_t, \{\boldsymbol{o}^i_t, z^i_t, \boldsymbol{a}^i_t, \boldsymbol{\phi}^i_t, r^i_L, \boldsymbol{o}^i_{t+1}\}_{i\in\mathcal{N}}, r_H, \boldsymbol{s}_{t+1}\}$
    else
      $\mathcal{D} = \mathcal{D} \cup \{\boldsymbol{s}^{G_j}_t, \{\boldsymbol{o}^i_t, z^i_t, \boldsymbol{a}^i_t, \boldsymbol{\phi}^i_t, r^i_L, \boldsymbol{o}^i_{t+1}\}_{i\in G^j}, r^{G^j}_H, \boldsymbol{s}^{G_j}_{t+1}\}_{\forall j \in \mathcal{N}_G}$
    end if
  end for
  Update high level policy network $\boldsymbol{\theta}$ according to (11), or, (12), using sampled data.
  Update low level policy network parameters $\boldsymbol{\nu}$ based on (23) and (24), using sampled data
end for

We implement the algorithm using two different approaches inspired by existing works in the area of MARL. Notably, we implemented the algorithm in the following ways. It is important to note that under all these approaches, the hierarchical policy is learned agent-wise to allow for decentralized execution.

1. Joint policy learning: This approach involves solving (10) by optimizing the high-level policies of the agents jointly by learning the policy parameters $\boldsymbol{\theta}^i$s.
2. Group policy learning: This approach is inspired by the idea of learning the value function by decomposing the joint value function into the value function of groups of agents. In our fully cooperative setting, the total step reward can be composed as a sum of the rewards of the individual agents, i.e. $r_H(\boldsymbol{s}, \boldsymbol{a}) = \sum_{i=1}^{N} r^i_H(\boldsymbol{s}^i, \boldsymbol{a}^i)$. Let $\{G^j\}_{j=1}^{N_G}$ represent the groups with each $G^j \subset \mathcal{N}$ such that, $G^j \cap G^{j'} = \varnothing$ and $\mathcal{N} = \cup_{i=1}^{N_G} G^i$. We can define the group reward as the sum over the individual reward of the agents in the group Then, under this approach we can express the objective in (10) as below (We only include the necessary symbols in the expectation to avoid

symbol overload):

$$
\begin{aligned}
J(\pi_H, \pi_{\boldsymbol{z}}) &= \mathbb{E}\left[\sum_{t'=0}^{\infty} \mathbb{E}_{\boldsymbol{a}_t \sim \pi_{\boldsymbol{z}_{t'}}}\left[\sum_{t=0}^{k_{t'+1}-1} \gamma^{t+k_{t'}} r_H(\boldsymbol{s}_t, \boldsymbol{a}_t)\right]\right] \\
&= \mathbb{E}\Bigg[\sum_{t'=0}^{\infty} \mathbb{E}_{\boldsymbol{a}_t \sim \pi_{\boldsymbol{z}_{t'}}, }\Bigg[\sum_{t=0}^{k_{t'+1}-1} \gamma^{t+k_{t'}} \sum_{j=1}^{N_G} \underbrace{\sum_{i=1}^{N} r_H^i(\boldsymbol{s}_t^i, \boldsymbol{a}_t^i)\mathbb{I}[i \in G^j]}_{=r_H^{G^j}(\boldsymbol{s}_t^{G^j}, \boldsymbol{a}_t^{G^j})}\Bigg]\Bigg] \\
&= \sum_{j=1}^{N_G} \mathbb{E}\Bigg[\sum_{t'=0}^{\infty} \mathbb{E}_{\boldsymbol{a}_t^{G^j} \sim \pi_{\boldsymbol{z}_{t'}}}\Bigg[\sum_{t=0}^{k_{t'+1}-1} r_H^{G^j}(\boldsymbol{s}_t^{G^j}, \boldsymbol{a}_t^{G^j})\Bigg]\Bigg] \qquad (32)
\end{aligned}
$$

Here, $\boldsymbol{s}^{G^j}$ and $\boldsymbol{a}^{G^j}$ are the states and actions of the agents in group G^j, The agent skill policies are learnt independently of each other. Hence. the joint skills policies can be written as $\pi_{\boldsymbol{z}} = \prod_{i=1}^{N} \pi_{z^i} = \prod_{j=1}^{N_G} \prod_{i \in G^j} \pi_{z^i} = \prod_{j=1}^{N_G} \pi_{\boldsymbol{z}^{G^j}}$ where $\pi_{\boldsymbol{z}^{G^j}}$ is the joint skills policy of the agents in the group G^j. Then, (32) can be written as:

$$
\begin{aligned}
J(\pi_H, \pi_{\boldsymbol{z}}) &= \sum_{j=1}^{N_G} \mathbb{E}\Bigg[\sum_{t'=0}^{\infty} \mathbb{E}_{\boldsymbol{a}_t^{G^j} \sim \pi_{\boldsymbol{z}_{t'}^{G^j}}}\Bigg[\sum_{t=0}^{k_{t'+1}-1} r_H^{G^j}(\boldsymbol{s}_t^{G^j}, \boldsymbol{a}_t^{G^j})\Bigg]\Bigg] \\
&= J^1(\pi_H^{G^1}) + \cdots + J^{\mathcal{N}_G}(\pi_H^{G^{N_G}})
\end{aligned}
$$

where, $k_{t'}$ now becomes the time at which any agent in the group changes selects/switches to a new skill. We drop the low-level policy corresponding to skills for brevity. This approach enables better scalability as the overall problem is decomposed into one that learns policies for the groups; however, this comes at the expense of performance as the original optimization problem involves all the agents.

Note that all these implementations retain the safety guarantee as it is offered by our safe skill-learning approach.

A.5.2 Implementation Details: METADRIVE Simulator

Metadrive environments illustrated in figure 4 by default, include 3 basic sensors: lidar, sideDetector and laneLineDetector which are used for detecting moving objects, sidewalks/solid lines, and broken/solid lines respectively. The lidar state observation returns a state vector containing necessary information for navigation tasks. The details of the observation can be found here [2]. The environmental safety constraints include i. the road boundaries and ii. the booth for tollgate environment only. And, each agent has to stay safe to all neighboring agents which are within the lidar sensing radius. It is important to mention that we only use a dynamic bicycle model for the CBF constraints, the environment uses pybullet physics engine for simulation. The kinematic bicycle model of the vehicle is provided in (33).

$$
\begin{aligned}
\dot{p} &= \frac{v\cos(\psi)}{1 - d\kappa(p)}, \\
\dot{d} &= v\sin(\psi), \\
\dot{\psi} &= \frac{v}{L}\tan(\delta) - \kappa(p)\dot{p}, \\
\dot{v} &= a.
\end{aligned} \qquad (33)
$$

where p represents the position of the vehicle along the reference path, d is the lateral deviation of the vehicle from the reference path, ψ denotes the yaw angle of the vehicle (heading angle relative to the path), and v is the longitudinal velocity of the vehicle. The control inputs are δ, which is the steering angle, and a, which is the acceleration (or deceleration). The parameter L denotes the wheelbase of the vehicle, and $\kappa(p)$ is the curvature of the reference path at position s. It is important to add that, these states are provided by the environment for each ego vehicle/agent in its observation.

[2] https://metadrive-simulator.readthedocs.io/en/latest/obs.html

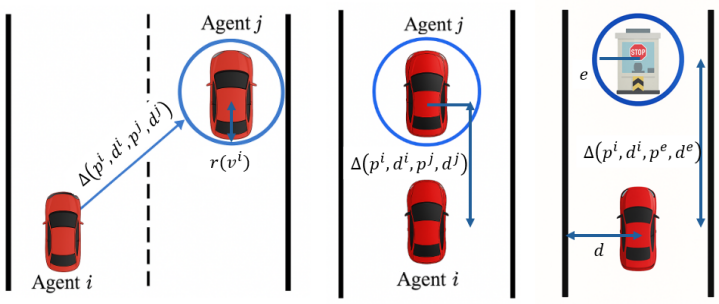


Figure 9: From left to right: illustration of i. inter agent safety constraints on another lane, ii. inter agent safety constraints on same lane and ii. environmental constraints.

In order to incorporate safety with respect to other agents and environmental obstacles, we define simple circles-based constraints as illustrated in figure 9. And, for the road boundaries we use the lateral deviation from the center lane to define safety functions. Let p^j and d^j denote the distance along the path and lateral deviation of another agent j with respect to agent i, located within within the sensing radius of agent i. This information is provided in the observation of agent i i.e. $(d^i, p^j, d^j) \in \mathcal{O}^i$. Note that, for an ego vehicle p^i is always 0 in its body frame. let $\Delta(p^i, d^i, p^j, d^j)$ denote the euclidean distance between agent i and j respectively. Similarly, let p^e and d^e denote the longitudinal and lateral distance of an environmental obstacle, like a tollgate booth as illustrated in figure 9 from agent i, and $\Delta(p^i, d^i, p^e, d^e)$ be the euclidean distance. Then, the safety constraints corresponding to other agent j, environmental constraints for an agent i are given below:

$$\Delta(p^i, d^i, p^j, d^j)^2 \geq r(v^i)\ ,\ \Delta(p^i, d^i, p^e, d^e)^2 \geq e;\ \text{, and } |d^i| \leq c \tag{34}$$

where $r(v^i)$ is a monotonically increasing function of v^i, $e \in \mathbb{R}_{>0}$ and $c \in \mathbb{R}_{>0}$.

Consider the CBF for the constraint for inter-agent safety defined as $h(\mathcal{O}^i) = \Delta(p^i, d^i, p^j, d^j) - r(v^i)$ where $\Delta(p^i, d^i, p^j, d^j) = \sqrt{(p^i - p^j)^2 + (d^i - d^j)^2}$, then the corresponding CBF constraint is given by:

$$\underbrace{2(p^i - p^j)\left(\frac{v^i \cos(\psi^i)}{1 - d^i \kappa(p^i)} - \dot{p}^j\right) + 2(d^i - d^j)\left(v^i \sin(\psi^i) - \dot{d}^j\right)}_{L_f h(\mathcal{O}^i)} + \underbrace{-2r(v^i)\frac{dr(v^i)}{dv^i}a^i}_{L_g h(\mathcal{O}^i) a^i}$$
$$+ \underbrace{\alpha\left((p^i - p^j)^2 + (d^i - d^j)^2 - r(v^i)^2\right)}_{h(\mathcal{O}^i)} \geq 0 \tag{35}$$

Additionally, we define CLFs for state reference tracking namely velocity and heading reference tracking. As we define termination state set corresponding to each skill, we define CLFs to drive the system to the termination state to be able to successfully execute the skill. However, it is important to note that the CLFs are not defined specific to skills and, hence, are generalized to the task. The CLFs corresponding to velocity and heading reference are given by the equation below. where v_t^i and $v_{t_f}^i$ are the velocities at We parameterize the corresponding CLF constraint which in case of the velocity and heading constraints becomes the following:

$$V(v^i, v_{des}^i) = (v^i - v_{des}^i)^2; \tag{36}$$

$$V(\psi^i, \psi_{des}^i) = (\psi^i - \psi_{des}^i)^2 \tag{37}$$

where v_{des} is the desired velocity and ψ_{des} is the desired heading of the vehicle.

Skills description. We define five skills for each agent for the METADRIVE environments which are: i. cruising, ii. accelerating (speed up), iii. yielding (slowing down), iv. left lane changing, and v. right lane changing.

Cruising. As the name suggests, this skill causes the vehicle to move at a constant speed for a fixed distance. The velocity CLF constraint in (36) is incorporated in the skill QP based policy by setting $v_{des} = 0$.

Accelerating/speeding up This skill corresponds to an agent increasing its longitudinal velocity. Let the initial velocity at the time of the initiation of the skill be v_0. The skill terminates when the velocity reaches $v_T = v_0 + dv$, where $dv \in \mathbb{R}_{>0}$ denotes the desired velocity increment and v_T is the velocity of the agent at the time of termination. Hence, the v_{des} is set to $v_0 + dv$ for this skill in the low level QP based skill policy. This skill facilitates timely task completion and helps avoid congestion by preventing agents from unnecessarily slowing down the traffic flow.

Yielding/slowing down. Yielding. This skill enables an agent to reduce its longitudinal velocity. Let v_0 be the velocity at skill initiation. The skill terminates when the velocity decreases to $v_T = v_0 - dv$, where $dv \in \mathbb{R}_{>0}$ is the desired reduction and v_T is the velocity at the time of termination of the skill. Thus, the velocity CLF in (36) is incorporated to the QP based skill policy by setting $v_{des} = v_0 - dv$ Yielding allows agents to safely navigate dense traffic and cooperate by yielding to agents approaching from other directions, thereby facilitating efficient joint task completion.

Left Lane Changing. This skill enables an agent to switch to the adjacent left lane. Let $d_{left,0}$ denote the distance to the left road boundary and d_0 be the lateral deviation from the center lane at the initiation of the skill. Let d_{lane} denote the lane width, provided by the environment. The target lateral position is computed as $d_{left,T} = d_{left,0} - d_{lane} + d_0$. We define the desired heading angle as a function of e i.e. $\phi_{\text{des}} = f(e)$, where $e = d_{left,T} - d_{left,0}$, and incorporate it into the CLF constraint on heading in (37) via a QP-based low-level skill control policy. This skill is essential for scenarios such as tollgates, bottlenecks, and merging zones, where adaptive lateral movement is required to maintain flow and avoid congestion.

Right Lane Changing. This skill enables an agent to switch to the adjacent right lane. Let $d_{right,0}$ denote the distance to the right road boundary and d_0 be the lateral deviation from the center lane at the initiation of the skill. Let d_{lane} denote the lane width, provided by the environment. The target lateral position is computed as $d_{right,T} = d_{right,0} - d_{lane} + d_0$. We define the desired heading angle as a function of e, i.e., $\phi_{\text{des}} = f(e)$, where $e = d_{right,T} - d_{right,0}$, and incorporate it into the CLF constraint on heading in (37) via a QP-based low-level skill control policy. This skill is useful in scenarios such as avoiding obstacles, preparing for highway exits, or facilitating merging from the right, enabling smoother and safer lane coordination.

As mentioned previously we include a timeout to ensure finite time termination of the skills for algorithmic sanity.

Intrinsic reward The intrinsic reward serves as an internal shaping signal that guides each agent toward desirable driving behaviors—smoothness, lane-centering, and reference-tracking—independently of any external task reward. By combining several negative penalties, the agent is discouraged from abrupt maneuvers, drifting off its lane, or deviating from its target speed and heading.

$$r_1^i = -c_1\|\boldsymbol{a}^i\|^2 - c_2\left(\frac{v^i - v_{\text{des}}}{v_{\text{des}}}\right)^2 - c_3\left(\frac{\psi^i - \psi_{\text{des}}}{\pi}\right)^2 - c_4\|d^i\|^2 \tag{38}$$

Where c_1, c_2, c_3, c_4, c_5 are the coefficients of each terms. The intrinsic reward combines penalties on aggressive acceleration and steering, deviations from reference speed and heading, and lateral offset from the lane center, with a small forward-progress bonus. Together, these terms encourage smooth, centered, and efficient driving without abrupt maneuvers.

CBF-Based Safe Skill Implementation. As described earlier, we parameterize the QP-based deterministic policy $\mu(\boldsymbol{o} \mid z, \boldsymbol{\phi})$ for each skill z, as detailed in Section 5.2. The QP formulation is parameterized through the objective, as well as the CBF and CLF constraints. In particular, we introduce parameters for the class-$\mathcal{K}$ function in the CBF constraint, the shape and size of the circular safety regions, and the convergence rate of the CLF constraint. To ensure validity, all parameters are constrained to be nonnegative using box constraints of the form $(0, \phi_H]$, where ϕ_H is a fixed upper bound applied uniformly across all parameters.

A.5.3 Implementation Details: Lidar Suite Simulator

Target. Agents dynamics in this environment are governed by double integrator dynamics with continuous-time states $\boldsymbol{x}_i = [p_x^i, p_y^i, v_x^i, v_y^i]^\top \in \mathbb{R}^4$, where $\boldsymbol{p}_i = [p_x^i, p_y^i]^\top \in \mathbb{R}^2$ denotes position and $\boldsymbol{v}_i = [v_x^i, v_y^i]^\top \in \mathbb{R}^2$ denotes velocity; control inputs $\boldsymbol{u}_i = [a_x^i, a_y^i]^\top \in \mathbb{R}^2$ represent accelerations

along the Cartesian axes, and the agent dynamics follow $\dot{\boldsymbol{x}}_i = [v_x^i, v_y^i, a_x^i, a_y^i]^\top$, with velocity and control constraints bounded in $[-10, 10]$ and $[-1, 1]$ respectively.

Spread. The underlying dynamics, state representation, control parametrization, and bounds are identical to those in the Target environment,i.e., agents evolve under double integrator dynamics with four-dimensional state vectors comprising positions and velocities, and two-dimensional control inputs that modulate accelerations; the bounded constraints on velocities and controls likewise remain $[-10, 10]$ and $[-1, 1]$, respectively.

Target Bicycle In the Target Bicycle environment, agents dynamics are modeled using a bicycle model where the state vector $\boldsymbol{x}_i = [p_x^i, p_y^i, \cos\theta^i, \sin\theta^i, v^i]^\top \in \mathbb{R}^5$ encapsulates the agent's position, heading orientation via its sine and cosine, and scalar speed; the control input $\boldsymbol{u}_i = [\delta^i, a^i]^\top$ includes the steering angle δ^i and linear acceleration a^i, with dynamics specified as $\dot{\boldsymbol{x}}_i = [v\cos\theta, v\sin\theta, -v\sin\theta\tan\delta, v\cos\theta\tan\delta, a]^\top$, where $v \in [-10, 10]$ and $\delta \in [-1.47, 1.47]$ are the admissible bounds on speed and steering angle.

Similar to the METADRIVE environments, we define CBFs based on distance/proximity as in (34). In our formulation, we do not employ a separate heading CLF; instead, we regulate agent behavior using only the velocity CLF by aligning the desired velocity vector with the desired heading direction. Specifically, given a desired heading direction θ_{des}, we compute a target velocity vector as $\boldsymbol{v}_{\text{des}} = v_{\text{mag}}[\cos(\theta_{\text{des}}), \sin(\theta_{\text{des}})]^\top$, where v_{mag} is a task-specific magnitude (e.g., maximum safe speed or adaptive goal-directed speed). This $\boldsymbol{v}_{\text{des}}$ is then used in the velocity CLF described below to guide both the speed and heading behavior of the agent.

For the Target and Spread environments, where the agent's longitudinal and lateral velocities v_x^i, v_y^i are explicit components of the state, we define the Control Lyapunov Function (CLF) to track a desired velocity vector $\boldsymbol{v}_{\text{des}} = [v_{x,\text{des}}, v_{y,\text{des}}]^\top \in \mathbb{R}^2$ as:

$$V(v_x^i, v_y^i) = \frac{1}{2}\left((v_x^i - v_{x,\text{des}})^2 + (v_y^i - v_{y,\text{des}})^2\right).$$

Given the double integrator dynamics, where the control inputs a_x^i and a_y^i directly affect the velocity via $\dot{v}_x^i = a_x^i$ and $\dot{v}_y^i = a_y^i$, the time derivative of the CLF becomes:

$$\dot{V} = (v_x^i - v_{x,\text{des}})a_x^i + (v_y^i - v_{y,\text{des}})a_y^i.$$

To enforce convergence, we include the following CLF constraint in the QP:

$$(v_x^i - v_{x,\text{des}})a_x^i + (v_y^i - v_{y,\text{des}})a_y^i \le -\alpha\left((v_x^i - v_{x,\text{des}})^2 + (v_y^i - v_{y,\text{des}})^2\right),$$

where $\alpha > 0$ is a tunable rate parameter.

In the Target Bicycle environment, each agent's velocity is represented as a scalar speed $v^i \in \mathbb{R}$, and the longitudinal acceleration $a^i \in \mathbb{R}$ serves as the control input directly affecting the speed via $\dot{v}^i = a^i$. To drive v^i to a desired speed $v_{\text{des}} \in \mathbb{R}$, we define the CLF as:

$$V(v^i) = \frac{1}{2}(v^i - v_{\text{des}})^2.$$

The time derivative of this CLF, using $\dot{v}^i = a^i$, is:

$$\dot{V} = (v^i - v_{\text{des}})a^i.$$

To enforce convergence of the velocity to the target, the following CLF constraint is imposed in the QP:

$$(v^i - v_{\text{des}})a^i \le -\alpha(v^i - v_{\text{des}})^2,$$

where $\alpha > 0$ is a tunable parameter determining the exponential rate of convergence.

Skills Description. We define four interpretable low-level skills for each agent in the LiDAR suite environments, which are: i. speeding up, ii. slowing down, iii. turning left, and iv. turning right.

Speeding Up. This skill allows an agent to increase its translational velocity to navigate more efficiently through open space or move rapidly towards its assigned goal. Let the velocity at the start of the skill be $\boldsymbol{v}_0 \in \mathbb{R}^2$; the skill is considered complete once the velocity reaches $\boldsymbol{v}_T = \boldsymbol{v}_0 + \Delta\boldsymbol{v}$, where $\Delta\boldsymbol{v} \in \mathbb{R}^2$, with $\|\Delta\boldsymbol{v}\| > 0$. The CLF constraint associated with velocity tracking is enforced

by setting the desired velocity $\boldsymbol{v}_{\text{des}} = \boldsymbol{v}_0 + \Delta\boldsymbol{v}$ in the QP-based low-level control policy. This skill facilitates faster progression in low-density scenarios while maintaining safety guarantees via CBFs.

Slowing Down. This skill enables the agent to reduce its speed when approaching dynamic obstacles, other agents, or goal regions where precise navigation is required. Let the initial velocity be $\boldsymbol{v}_0 \in \mathbb{R}^2$, and define the skill termination condition as reaching $\boldsymbol{v}_T = \boldsymbol{v}_0 - \Delta\boldsymbol{v}$, where $\Delta\boldsymbol{v} \in \mathbb{R}^2$ and $\|\Delta\boldsymbol{v}\| > 0$. To enforce this behavior, the velocity CLF is specified using $\boldsymbol{v}_{\text{des}} = \boldsymbol{v}_0 - \Delta\boldsymbol{v}$. This skill is critical for smooth and safe deceleration, reducing the risk of collisions and improving maneuverability near goals.

Turning Left. This skill enables the agent to redirect its heading counterclockwise to avoid obstacles or navigate toward an alternate subgoal. Let the initial heading be θ_0, and the desired heading be $\theta_{\text{des}} = \theta_0 + \Delta\theta$, where $\Delta\theta > 0$. Instead of using a heading CLF, we convert the desired heading into a velocity vector using a fixed target magnitude v_{mag}, and set $\boldsymbol{v}_{\text{des}} = v_{\text{mag}}[\cos(\theta_{\text{des}}), \sin(\theta_{\text{des}})]^\top$. This desired velocity is then tracked via the standard velocity CLF in the QP. The turning-left skill is useful for curvature adjustment in cluttered or multi-agent scenes.

Turning Right. This skill enables clockwise heading correction using the same velocity CLF formulation. Given initial heading θ_0 and desired adjustment $\Delta\theta > 0$, the target heading is $\theta_{\text{des}} = \theta_0 - \Delta\theta$, which is converted to a velocity vector as $\boldsymbol{v}_{\text{des}} = v_{\text{mag}}[\cos(\theta_{\text{des}}), \sin(\theta_{\text{des}})]^\top$. The agent's translational motion is then regulated by the velocity CLF, enforcing convergence to the desired heading direction through velocity tracking. This skill is particularly effective in obstacle-rich or competitive zones where sharp directional changes are required.

A.6 Additional Results

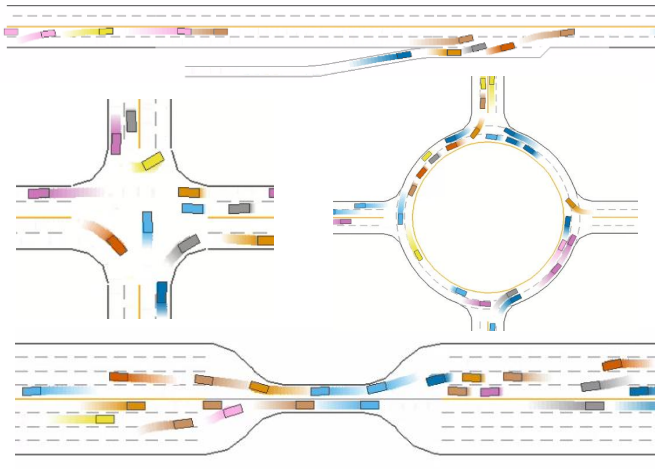


Figure 10: Visualization of agents' trajectories under the trained model: for each vehicle, its ten most recent positions are shown in a unique hue, with brightness fading from light (earlier points) to dark (later points) to indicate temporal progression.

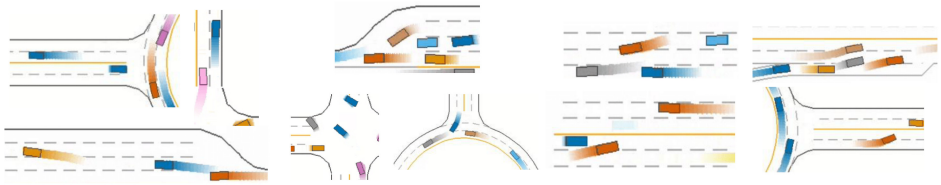


Figure 11: Visualization of skills under the trained model. From left to right: (a) acceleration (speed-up), (b) yielding (slow-down), and (c) lane changing

Here, we present results to further demonstrate the merit of our proposed algorithm. Specifically, we examine the trajectories in detail to analyze the distinct skills exhibited by the agents, offering insight

into how our method sustains an extremely high success rate while also outperforming the baselines in terms of energy efficiency and episode duration. To this end, we employ a visualization scheme in which each agent's ten most recent positions are depicted in a distinct color, with brightness gradually diminishing from light (earlier points) to dark (later points) to convey temporal progression. An example of this visualization is depicted in Figure 10. By looking more closely, we can identify some of the skills that the agents exhibit during the episode. For instance, as agents approach their destinations (i.e., exit points of the environment), they execute speed up skill only when safety constraints are satisfied, thereby enhancing road throughput and reducing average travel time. Several such examples are shown in Figure 11a. Yielding (i.e., controlled slow down/deceleration) is another common skill agents employ to maintain a safe distance from other vehicles. By anticipating the need to yield, agents avoid abrupt braking, which both enhances safety and reduces overall episode energy consumption. Several such examples are shown in Figure 11b. Another skill exhibited by agents is lane changing: when executed safely at appropriate locations, it enhances road throughput and consequently reduces the average episode duration. An example of such lane changing is depicted in 11c where the agent with brown color is changing its lane— creating a safe gap into which the grey agent then merges onto the main road.