Supplementary Material for KING: Generating Safety-Critical Driving Scenarios for Robust Imitation via Kinematics Gradients

Niklas Hanselmann[1,2] **Katrin Renz**[2,3] **Kashyap Chitta**[2,3]
Apratim Bhattacharyya[2,3] **Andreas Geiger**[2,3]

[1]Mercedes-Benz AG R&D, Stuttgart [2]University of Tübingen
[3]Max Planck Institute for Intelligent Systems, Tübingen

Abstract: In this **supplementary document**, we first provide additional information regarding AIM-BEV (Sec. 1.1) and the rule-based expert (Sec. 1.2). Next, we provide additional information on the kinematics model (Sec. 2) and a detailed description of our cost functions (Sec. 3). Furthermore, we discuss details regarding the maps, routes and traffic initialization of the scenarios in Sec. 4. Finally, we present details on the hyperparameter choices for all optimization methods, further results, as well as additional qualitative examples in Sec. 5. The **supplementary video** contains qualitative visualizations of the scenarios generated via KING and of the improved robustness of AIM-BEV in CARLA when fine-tuned on those scenarios.

1 Driving Agents

In this work, we consider three driving agents: (1) Our AIM-VA [1] variant termed AIM-BEV, (2) TransFuser [2] and (3) a privileged rule-based expert, which provides supervision for AIM-BEV and TransFuser and is additionally used to determine solvability for the scenarios generated by KING. In the following we provide additional details on AIM-BEV and the rule-based expert. For TransFuser, we refer readers to the original paper. In addition, we provide further information on the driving metrics used to benchmark their performance in the CARLA simulator.

1.1 AIM-BEV

Model: Our architecture is similar to the AIM models introduced in previous work [2, 1]. For efficiency during simulation, we use a lightweight MobileNetV3-Small [3] instead of the ResNet34 used in AIM as a convolutional backbone. As described in the main paper, it takes a bird's-eye view (BEV) semantic occupancy grid observation $\mathbf{o}_t$ as input. From this, a 512-dimensional feature vector is extracted (same as in AIM). This is then further processed by three fully-connected layers (256, 128 and 64 units with ReLU activations) to yield a 64-dimensional representation of $\mathbf{o}_t$. To be able to train and evaluate the model in the CARLA simulator [4], we concatenate a binary variable indicating if the vehicle should stop for a red light to this representation. As we do not model traffic lights in KING scenarios, this variable is always set to zero in our proxy simulation.[1] Finally, the resulting feature is input to a single layer gated recurrent unit (GRU) with a linear decoder, which autoregressively estimates a set of four future waypoints $\{\mathbf{w}_i\}_{i=1}^{4}$ representing the desired trajectory. In each timestep, the GRU is additionally conditioned on a sparse goal location $\mathbf{x}_{goal}$ indicating the intended high-level route. To convert the waypoints into low-level actions we use a set of two PID controllers, which we describe next.

[1]Similarly, we use an additional channel in the BEV representation to encode pedestrians for deployment in CARLA, which is inactive when optimizing for and evaluating on KING scenarios.

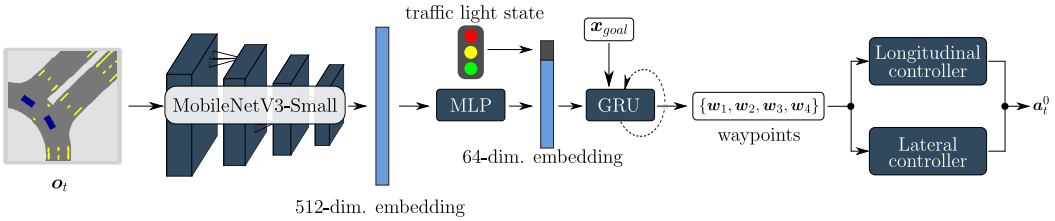


Figure 1: **Architecture Overview.** Illustration of the AIM-BEV architecture. First, a low-dimensional embedding from the input BEV semantic grid map observation $\mathbf{o}_t$ is extracted. This is then used as the basis for autoregressive waypoint prediction, from which the final actions are obtained via lateral and longitudinal PID controllers.

Controllers: We obtain actions for steering and acceleration from separate lateral and longitudinal controllers (LatPID and LonPID in Algorithm 1). These are similar to those used in [5, 6, 1], with slight modifications. In particular, we replace hard clipping and thresholding operations with soft versions to maintain differentiability. For these, we tune the shape parameters $\beta_{eb}, \beta_{decel}$ and clipping bounds $c_\delta, c_{throttle}$ to approximate the original non-differentiable implementation. For steering, we obtain a control target by computing the relative orientation Φ_d of the halfway point $\mathbf{w}^*$ between the first and second waypoint. This is then input to the lateral PID controller (with gains $K_p = 1.25$, $K_i = 0.75$, $K_d = 0.3$), which returns the corresponding steering action. Because this is unbounded, we constrain it to the $[-1, 1]$-interval by applying the *tanh* function. We have found a buffer of length 4 to be sufficient for computing the integral and derivative terms in both PID controllers. For throttle, we obtain the desired speed v_d by computing the norm of the vector connecting the first and second waypoint. From this we then compute a throttle value for the desired acceleration using the longitudinal PID controller ($K_p = 5$, $K_i = 0.5$, $K_d = 1$), as well as a weighting factor α indicating if the agent should decelerate (throttle = -1). The latter is high either when the current desired speed v_d is lower than an emergency brake threshold τ_{eb} or when the current speed is higher than v_d. In our kinematics model, which we detail in Section 2, the state update is then computed using the α-weighted sum of both the acceleration and deceleration throttle values. As this effectively allows to express the entire $[-1, 1]$ range of throttle values, we denote the action space of the driving policy as $\mathbf{a}_t^0 \in [-1, 1]^2$ in the main paper for notational convenience.

Algorithm 1: Computing low-level actions from waypoints.

parameters: $\tau_{eb} \leftarrow 0.6$, $\beta_{eb} \leftarrow 10$, $\beta_{decel} \leftarrow 1$, $c_\delta \leftarrow 0.25$, $c_{throttle} \leftarrow 0.75$
input : v, $\{\mathbf{w}_i\}_{i=1}^4$
output : steer $\in [-1, 1]$, throttle $\in (0, 1)$, $\alpha \in (0, 1)$

$\mathbf{w}^* \leftarrow \frac{\mathbf{w}_1 + \mathbf{w}_2}{2}$;
$\Phi_d \leftarrow \tan^{-1}\left(\frac{\mathbf{w}^*[1]}{\mathbf{w}^*[0]}\right)$;
steer $\leftarrow \tanh\left(\text{LatPID}\left(\Phi_d\right)\right)$;
$v_d \leftarrow \|\mathbf{w}_2 - \mathbf{w}_1\| \times 2$;
$\alpha_{eb} \leftarrow \text{sigmoid}\left(-\beta_{eb} \times (v_d - \tau_{eb})\right)$;
$\alpha_{decel} \leftarrow \tanh\left(\beta_{decel} \times \text{ReLU}\left(v - v_d\right)\right)$;
if $v_d < \tau_{eb}$ **then**
  $\alpha \leftarrow \alpha_{eb}$
else
  $\alpha \leftarrow \alpha_{decel}$
end
$\delta \leftarrow c_\delta \times \tanh\left(\beta_\delta \times \text{ReLU}\left(v_d - v\right)\right)$;
throttle $\leftarrow c_{throttle} \times \tanh\left(\text{ReLU}\left(\text{LonPID}\left(\delta\right)\right)\right)$;

Training on regular data: For initial training of AIM-BEV, we collect a dataset $\mathcal{D}_{reg}$ of demonstrations by the rule-based expert described in Section 1.2 in regular traffic. We sample a dense set of routes for the expert to complete, covering all possible junctions in Town01-Town06 available in

the CARLA simulator as well as routes outside of the junctions (e.g., highways). We include the hand-crafted CARLA scenarios during data collection to cover basic critical situations. In total, the dataset contains 91000 demonstrations, the equivalent of $12.6\,\mathrm{h}$ of driving. We obtain ground truth supervision on the waypoints predicted by the driving policy from the future trajectory driven by the expert during data collection. The goal location $\mathbf{x}_{goal}$ is obtained by the route planner of [5], which is based on an A* planner. Using this, the sparse waypoint at least $7.5\,\mathrm{m}$ ahead of the current position provided by CARLA is queried to obtain $\mathbf{x}_{goal}$. To train the driving policy using this dataset, we further employ data augmentation. For this, we rotate the BEV, waypoints and goal location around the vehicle center to simulate situations where the policy has to recover to stable lane keeping. We augment 50% of the samples with rotation angles sampled uniformly between ± 20 degrees. The policy is then trained using an L1-Loss on the waypoints for 350 epochs. We use the Adam optimizer with a learning rate of 0.001 and a batch size of 128.

Fine-tuning with safety-critical scenarios: Similar to $\mathcal{D}_{reg}$, we build $\mathcal{D}_{crit}$ by collecting demonstrations from the rule-based expert. The overall approach remains similar. However now, instead of driving in CARLA under regular traffic, the expert solves scenarios generated by KING in our closed-loop proxy simulation. As described in the main paper, we first collect a larger set of KING scenarios that are solvable by the expert, containing around 8000 training samples. When training on a union of both datasets $\mathcal{D}_{crit} \cup \mathcal{D}_{reg}$, we ensure a certain proportion of critical to regular data in each minibatch. We find a mix of 60% regular and 40% critical data to work well in practice. Aside from an increased batch size of 256 the training hyperparameters and augmentations remain the same. We fine-tune all models for 15 epochs.

1.2 Rule-based Expert

We use a rule-based expert policy based on privileged information to generate training data and determine solvability for the scenarios generated by KING. For the initialization of non-critical scenarios (see Section 4) we further adapt this expert to also work in our proxy simulation. We build on the code provided by [5, 1], which is based on the structure of the CARLA traffic manager[2]. It consists of three main components, (1) a collision stage, (2) a traffic light stage and (3) a motion planner stage, which we detail next.

Collision stage: In order to predict possible future collisions we use a kinematic bicycle model to extrapolate the position and orientation of all other agents within a radius of $30\,\mathrm{m}$. For the extrapolation we use an action repeat assumption (i.e., the exact same throttle and steer value is repeated for all extrapolation timesteps) since we do not have any information about the future ground truth actions. For the ego agent, we extrapolate assuming actions that maintain its current target speed. This allows reasoning about the possibility of a collision and necessary braking maneuvers. Outside of junctions, we extrapolate for a horizon of $1\,\mathrm{s}$. In junctions we extrapolate for a longer horizon of $4\,\mathrm{s}$ due to a higher probability of collision. We express each future timestep through a bounding box with the size of the respective vehicle and the extrapolated future position and orientation. With these bounding boxes we then check for overlaps between the ego and the other vehicles in each timestep to detect possible future collisions. In case of a bounding box intersection we set a brake hazard flag. As an additional security measure we extend the ego bounding box of the current timesteps to the front and check for intersections in this timestep. The extent is computed depending on the braking distance for the current ego speed.

Traffic light stage: In the traffic light stage we detect traffic lights and stop signs. For this we use privileged information provided by CARLA through trigger boxes. If there is an intersection of the boxes and the internal CARLA state of the traffic light is red or yellow we set the brake hazard flag.

Motion planner stage: For the main logic we refer to Section 2.2 of the supplementary material of [1]. However, we observed a higher driving score when modifying the set of target speeds. We use a default target speed of $4\,\mathrm{m/s}$. When entering an intersection this is increased to $5\,\mathrm{m/s}$. We

[2] https://carla.readthedocs.io/en/latest/adv_traffic_manager/

Infraction Type	Penalty
Collisions with pedestrians	0.50
Collisions with other vehicles	0.60
Collisions with static elements	0.65
Running a red light	0.70
Running a stop sign	0.80

Table 1: **Infraction Score (IS) penalties for different infraction types.**

observed that moving swiftly in intersections lowers the probability of collision. When the brake hazard flag is set, the target speed is always $0\,\mathrm{m/s}$. To adapt to the lower frame rate in our simulator ($4\,\mathrm{fps}$ vs $20\,\mathrm{fps}$), we use adjusted gains (K_p = 1.75, K_i = 0.75, K_d = 3.5) for the lateral controller (only in our simulator). We leave the longitudinal controller unchanged.

1.3 Driving Metrics for CARLA Benchmarking

In the main paper we benchmark the driving agents wrt. to their driving performance in CARLA. This is measured by the *Driving Score (DS)*, which is composed of two terms, the Route Completion and the Infraction Score, which we detail in the following. The *Route Completion (RC)* is given by

$$\mathrm{RC} = \frac{1}{N_R} \sum_{i}^{N_R} R_i \tag{1}$$

where R_i is the percentage of the total distance to be driven on route i that the agent has completed, and N_r is the total number of routes. This is complemented by the *Infraction Score (IS)*, which measures the agent's compliance with traffic rules. It is computed as a product of infraction penalties p_i^j for each instance i of an infraction of type j. An overview of the type-specific penalties is shown in Table 1. In addition to these, there is one further penalty for deviations from the drivable area. This is computed as $(1 - R_{offroad})$, where $R_{offroad}$ denotes the percentage of the route completed outside of the drivable area. Overall, the agent starts with a base IS of one, which is reduced for each infraction. The DS is then computed as the product of the RC and IS, averaged over all routes.

2 Kinematics Model

For the kinematic bicycle model we follow previous work [7] and use the author provided code and parameters, which were fitted to the CARLA dynamics and default vehicle model. Their implementation constrains the speed to be positive via a hard threshold to prevent backwards motion. This zeroes out gradients through the indirect path in timesteps where the current throttle action would result in negative speeds. We hence replace the hard threshold with a softplus activation. We set its shape parameter β to 7, which we found to result in sufficiently low error.

3 Cost Functions

Here, we provide more details of the costs from Section 3.3. We begin with the ego-collision cost $\phi_{col}^{ego}(\mathcal{S})$ followed by the two regularization terms $\phi_{col}^{adv}(\mathcal{S})$ and $\phi_{dev}^{adv}(\mathcal{S})$.

Ego Collision Cost: As described in the main paper, the ego collision cost $\phi_{col}^{ego}(\mathcal{S})$ is an attractive potential encouraging close encounters between the driving policy and the adversarial agents. This attractive potential is proportional to the distance between the ego agent and the adversarial agents, for which the geometry of the ego agent and the adversarial agents is taken into account. From the main paper, let B_t^i be the bounding rectangle of an agent i. The distance $d(\mathrm{B}_t^0, \mathrm{B}_t^i)$ between the bounding rectangle of the ego-agent (0) to that of the adversarial agent (i) can be efficiently computed as the minimum of the distances between every vertex-edge pair of B_t^0 and B_t^i.

Adversarial Collision Regularization: We discourage collisions between adversarial agents through a repulsive potential $\phi_{col}^{adv}(\mathcal{S})$ with an activation threshold. This potential is inversely proportional to the distance between the closest pair of adversarial agents. Let B_t^j and B_t^k be the bounding rectangles of the adversarial agents j and k. Then, the distance $d(\mathrm{B}_t^j, \mathrm{B}_t^k)$ is again computed as the minimum of the distances between every vertex-edge pair of B_t^j and B_t^k (similar to the ego collision as described above). Finally, we apply a threshold τ to the distance. This ensures that the repulsive potential is active only in a local neighbourhood around each adversarial agent j, preventing penalization of agents which are further apart than this threshold. We find that $\tau = 1.25\,\mathrm{m}$ is sufficient in practice.

Out-of-bounds Regularization: Finally, as described in the main paper, we regularize adversarial agents to stay within drivable parts of the map. For this, we use a repulsive potential $\phi_{dev}^{adv}(\mathcal{S})$ between the adversarial agents and off-road areas. To ensure that this is compliant with the geometry of the vehicle, the overall potential is composed of multiple terms computed at each of the corners of the bounding rectangle of every adversarial agent. Concretely, it is defined through Gaussian potentials $g(\mathrm{B}_t^i(c_k))$ at each corner c_k of the polygon B_t^i corresponding to the adversarial agent i. The potential at each corner is obtained through a convolution operation between the Gaussian potential $g(\mathrm{B}_t^i(c_k))$ and the map $\mathcal{M}_{oob}$, which is $g(\mathrm{B}_t^i(c_k)) \circledast \mathcal{M}_{oob}$. We consider a binary map $\mathcal{M}_{oob}$ which marks the non-drivable areas. The potential $g(\mathrm{B}_t^i(c_k)) \circledast \mathcal{M}_{oob}$ is thus large when the corner c_k is near (or on) a non-drivable area. The full "out-of-bounds" repulsive potential is defined as sum over all corners of the polygon B_t^i (as defined in the main paper) across agents and over all time-steps,

$$\phi_{rddev}^{adv} = \frac{1}{T} \sum_{i=0}^{N} \sum_{t=0}^{T} \sum_{k=0}^{C} g(\mathrm{B}_t^i(c_k)) \circledast \mathcal{M} \tag{2}$$

In practice, a variance of $1\,\mathrm{m}$ works well for the Gaussians. This ensures that the potential is large when any corner is $\sim 0.5\,\mathrm{m}$ from any non-drivable area.

Cost Weighting: As the number of actors in the scene is always known ahead of optimization, we use separate cost weightings for each traffic density. This accommodates for the fact that for different densities, a slightly different emphasis of the regularization terms can be optimal. The used weightings can be found in Tab. 2.

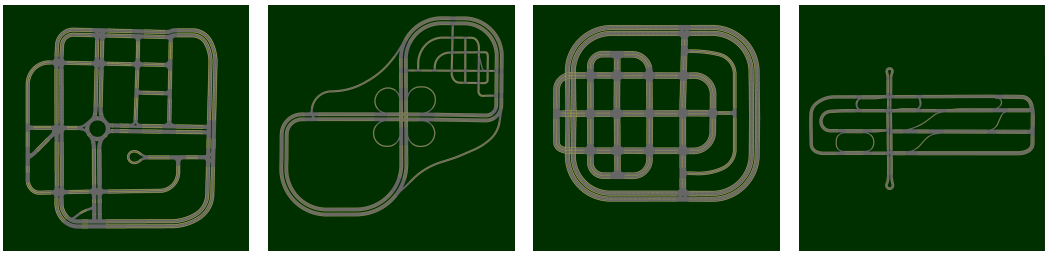
# agents	γ (for ϕ_{dev}^{adv})	λ (for ϕ_{col}^{adv})
1	20	0
2	23	5
4	20	3

Table 2: **Cost weightings.** Weighting of the costs for different traffic densities.

4 Scenarios

In this section we provide additional information on the maps and routes as well as the traffic initialization scheme, which provide the basis for the dataset of initial non-critical scenarios used throughout the main paper.

Maps and Routes: We use routes from towns 3-6 provided by the CARLA simulator for optimization. The layouts of these towns are illustrated in Fig. 2. Each town has unique features in terms of road topology, such as multi-lane highways, highway exits or roundabouts. From each of these towns we densely sample routes through junctions and intersections to obtain a set of 714 routes in total, each of which is a potential mission for the ego agent. This forms a diverse basis for possible scenarios. For evaluation on the Town10 intersections, we sample additional routes using the same sampling mechanism from CARLA's Town10, which we found to be the most challenging for the driving agents. This set consists of 82 different routes. We use the maximum traffic density for



(a) Town03 (b) Town04 (c) Town05 (d) Town06

Figure 2: **Town layouts.** Each town has unique features, such as roundabouts (a) or highway junctions (b, d).

Town10 (156 background vehicles) and spawn one hand-crafted CARLA scenario[3] along each route. Specifically, we consider Scenarios 7, 8, 9, and 10, which involve multiple vehicles simultaneously entering the intersection.

Initializating Traffic: As a starting point for safety-critical scenario generation, we require initial non-critical scenarios. Furthermore, to be able to study the effects of different traffic densities, the number of adversarial agents present in these initializations should be controllable. To this end, we mimic CARLA traffic by controlling the adversarial agents with the rule-based expert described in Section 1.2 to obtain an initialization. Specifically, we first sample a route that passes by the ego agent's route from the dense set of 714 routes for every adversarial agent. For this we consider routes which start and end at least $4\,\mathrm{m}$ and at most $100\,\mathrm{m}$ away from the ego agent's route. If possible, each agent is spawned on an individual route at the corresponding start location. If the set of candidate routes is smaller than the number of adversarial agents, additional spawn locations further along already occupied routes are used for the remaining agents while making sure a distance of $6\,\mathrm{m}$ between spawn locations is kept. Finally, using the expert, we roll out initial trajectories for the adversarial agents that follow these routes while avoiding collisions with each other and the ego agent.

Pre-filtering: For our experiments we sample a subset of the available routes. Instead of directly sampling from the total set of 714 routes, we first apply a pre-filtering step. For this, we initialize a scenario on every route as described above. Then, we remove all routes for which the scenario terminates due to an adversarial agent collision within the first 10 timesteps. This is done to handle edge cases not covered by the spawning logic. After pre-filtering, we sample 20 routes and corresponding initial scenarios per town from the remaining set to obtain the benchmark routes used to compare KING against black-box optimization (BBO) baselines in Tab. 2 of the main paper. To build the larger set used in the fine-tuning experiments for improving robustness, we use the entire set of routes remaining after pre-filtering.

5 Additional Results and Details

In this section we provide additional results supplementing the findings in the main paper. First, we detail the optimization hyperparameter choices for all methods used to generate safety-critical scenarios. Then, we perform an ablation study on the regularization terms in the generation objective. Next, we assess the impact of their higher computational burden for KING with both the direct and indirect path and for Bandit-TD. Finally, we provide additional qualitative examples of safety-critical scenarios.

5.1 Optimization Hyperparameters

For gradient-based generation via KING, we use the Adam optimizer with a learning rate of $5e-3$. We use a decay rate of $\beta_1 = 0.8$ for all traffic densities, and $\beta_2 = 0.999$ for a traffic density of

[3] https://leaderboard.carla.org/scenarios/

Method	Hyperparameter	Sweep Range	$N=1$	$N=2$	$N=4$
Random Search	Perturbation Bounds	$[0.01, 0.3]$	0.2	0.2	0.2
Bayesian Optimization	Exploration Kappa κ	$[0.1, 10]$	5	10	0.1
	Initial Random Points	$[1, 50]$	20	20	20
SimBA [8]	Perturbation Bounds	$[0.001, 0.2]$	0.05	0.05	0.05
CMA-ES [9]	Initial Standard Deviation σ	$[0.1, 0.4]$	0.2	0.1	0.4

Table 3: **Hyperparameter choices for BBO baselines.**

ϕ_{col}^{ego}	ϕ_{dev}^{adv}	ϕ_{col}^{adv}	CR ↑	$t_{50\%}$ ↓	s/it ↓
✓	-	✓	66.25	6.94	1.59
✓	✓	-	57.50	10.21	2.03
✓	-	-	60.00	7.08	1.59
✓	✓	✓	**78.75**	**6.40**	2.03

Table 4: **Cost ablation.** We show the performance for different regularization combinations for the 4 agent setting. Both regularization terms in combination with our main objective results in the best performance.

$N = 1$ and $\beta_2 = 0.99$ for $N \in \{2, 4\}$. We also adopt this to optimize the scenario parameters via the numerical gradients obtained from Bandit-TD. An overview of the remaining hyperparameters can be found in Tab. 3. For Random Search and SimBA [8], we tune the bounds of the interval from which perturbations can be sampled. For Bayesian Optimization, we search for both the optimal exploration value κ and the number of initial random points. Finally, for CMA-ES [9], we select an optimal initial standard deviation σ.

5.2 Cost ablation study

We study the influence of the regularization terms in our objective in Tab 4. While the out-of-bounds term $\phi_{dev}^{adv}(\mathcal{S})$ leads to a drop in performance when considering the regularization terms in isolation due to its computational expense, it is evident that combining both terms is crucial to achieve good results. Note that the losses for ego-adversarial and adversarial-adversarial collisions are computed in parallel as a batched operation, resulting in the same runtime when deactivating one of them.

5.3 Convergence

Bandit-TD and KING with both the direct and indirect gradient path are disadvantaged by their higher computational expense when imposing a fixed wall clock budget. We are interested in as-

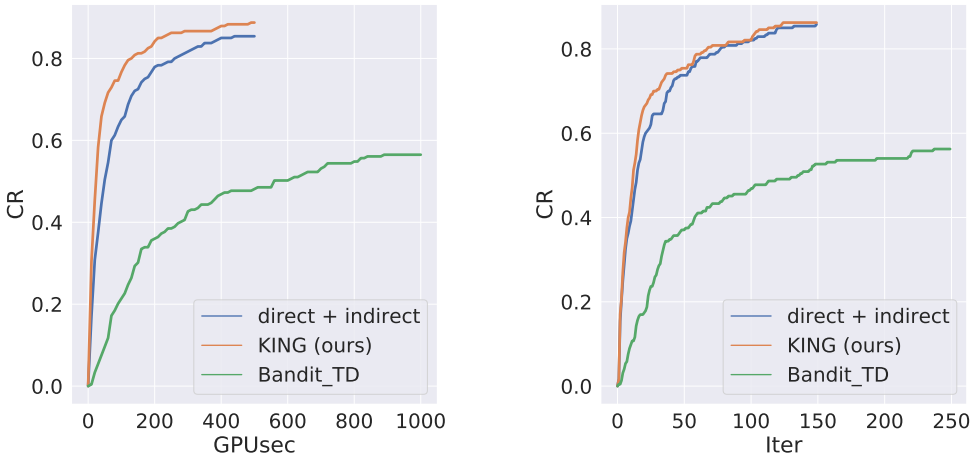


Figure 3: **Results at higher compute budget.** We compare KING to KING with both the direct and indirect path, and to Bandit-TD for a larger compute budget (left) and in terms of optimization iterations (right).

sessing their performance in the absence of this. For this, we show both the overall CR over GPU seconds for a less strict budget and the overall CR over optimization iterations (Fig. 3). As can be seen, given a high enough budget, Bandit-TD can eventually partially close the gap to other BBO methods. For KING with both paths, we observe that given the same number of optimization iterations, it achieves similar results compared to the indirect path only.

5.4 Additional Qualitative Examples

Finally, we show additional qualitative examples of safety-critical scenarios for both AIM-BEV (Fig. 4-9) and TransFuser (Fig. 10). As can be seen, they cover diverse situations and highlight the brittleness of current imitation learning (IL)-based driving agents.

Figure 4: **AIM-BEV - Cluster (a).** The driving agent collides with a slow moving vehicle while taking a turn (left). Additionally, it often struggles to handle lane-change scenarios safely (right). This is also evident in cluster (c).

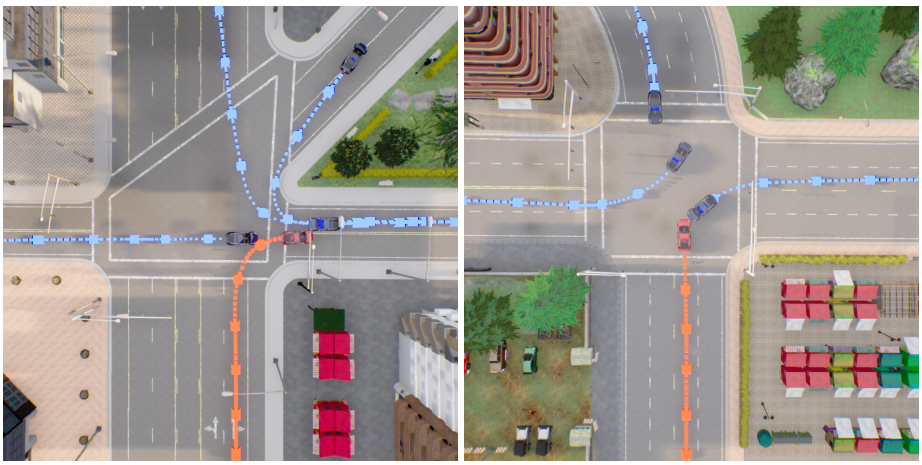


Figure 5: **AIM-BEV - Cluster (b).** The driving agent understeers when taking a turn, colliding with another vehicle that has been slightly displaced from its lane center by the safety-critical perturbation (left). Background traffic in CARLA usually closely adheres to the lane centers, making penalization of bad lane following less likely. On the right hand side, overconfident behavior in an unprotected intersection leads to collision.

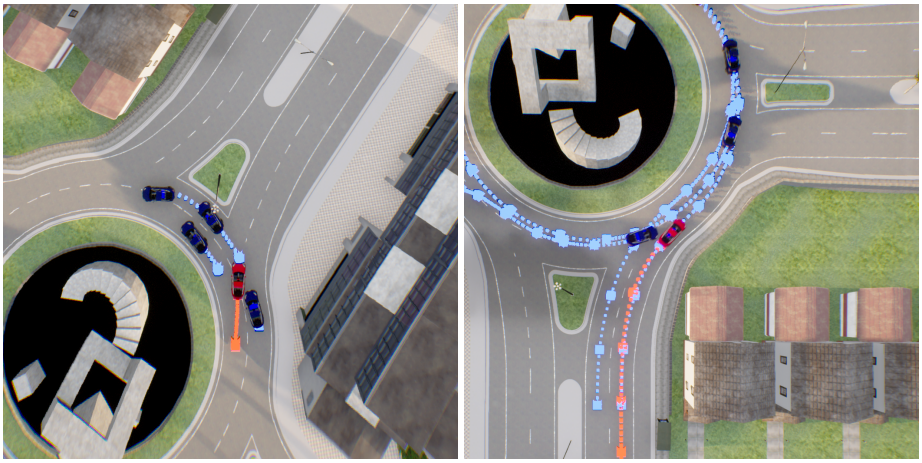


Figure 6: **AIM-BEV - Cluster (c).** The driving agent fails to handle merges and lane-changes in a roundabout safely. These maneuvers are especially problematic due to AIM-BEV's forward facing field of view. This shortcoming is not as apparent in regular CARLA traffic, highlighting KING's ability to draw attention to possible failure modes.

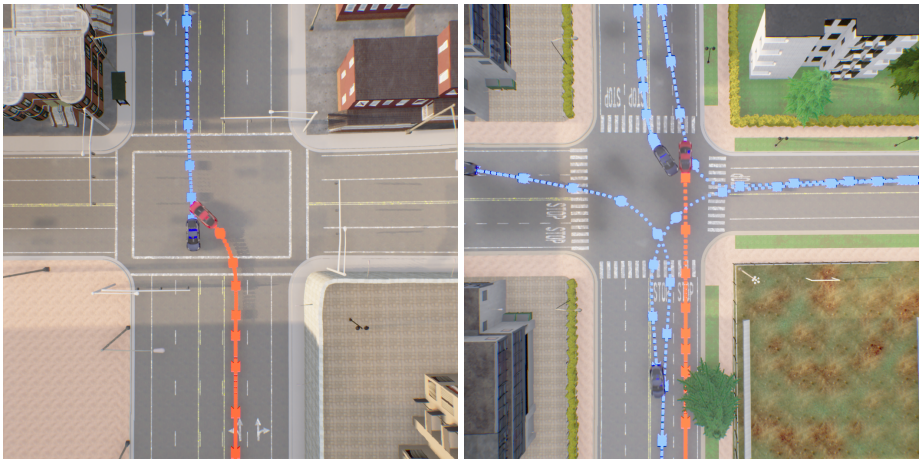


Figure 7: **AIM-BEV - Cluster (d).** This cluster primarily features collisions at the side of the driving agent. As depicted, this type of collision can arise from insufficient safety distance while turning (left) or failure to yield for another vehicle taking an aggressive turn (right).

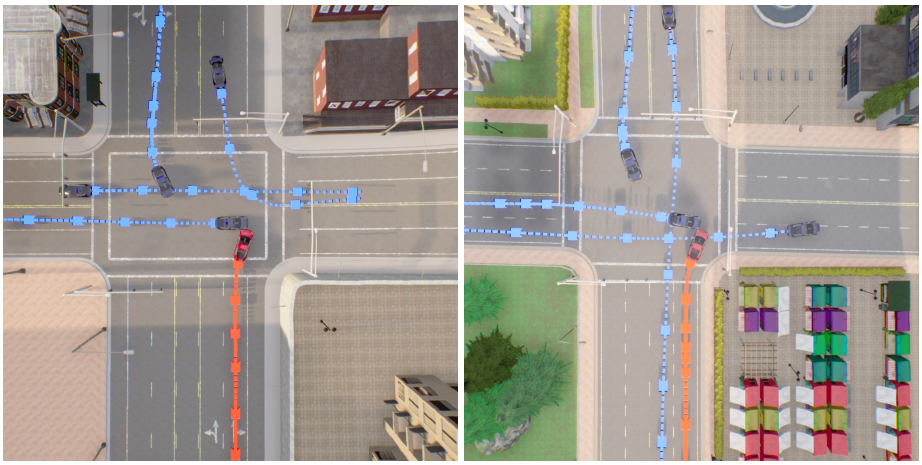


Figure 8: **AIM-BEV - Cluster (e).** The driving agent fails to brake when traffic in the intersection does not yield. As a result the agent collides into the side of another vehicle.

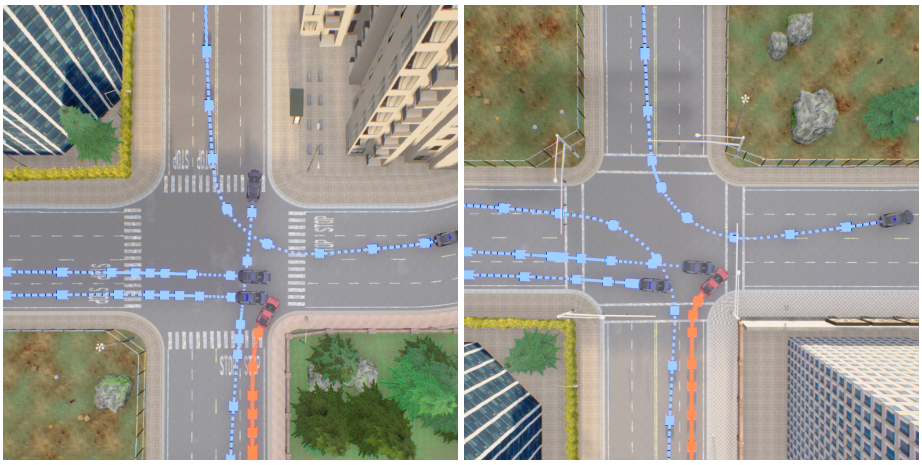


Figure 9: **AIM-BEV - Cluster (f).** The examples shown here depict similar situations as in cluster (e). The ego agent fails to correctly assess if it is safe to continue on, resulting in other traffic colliding into its side.

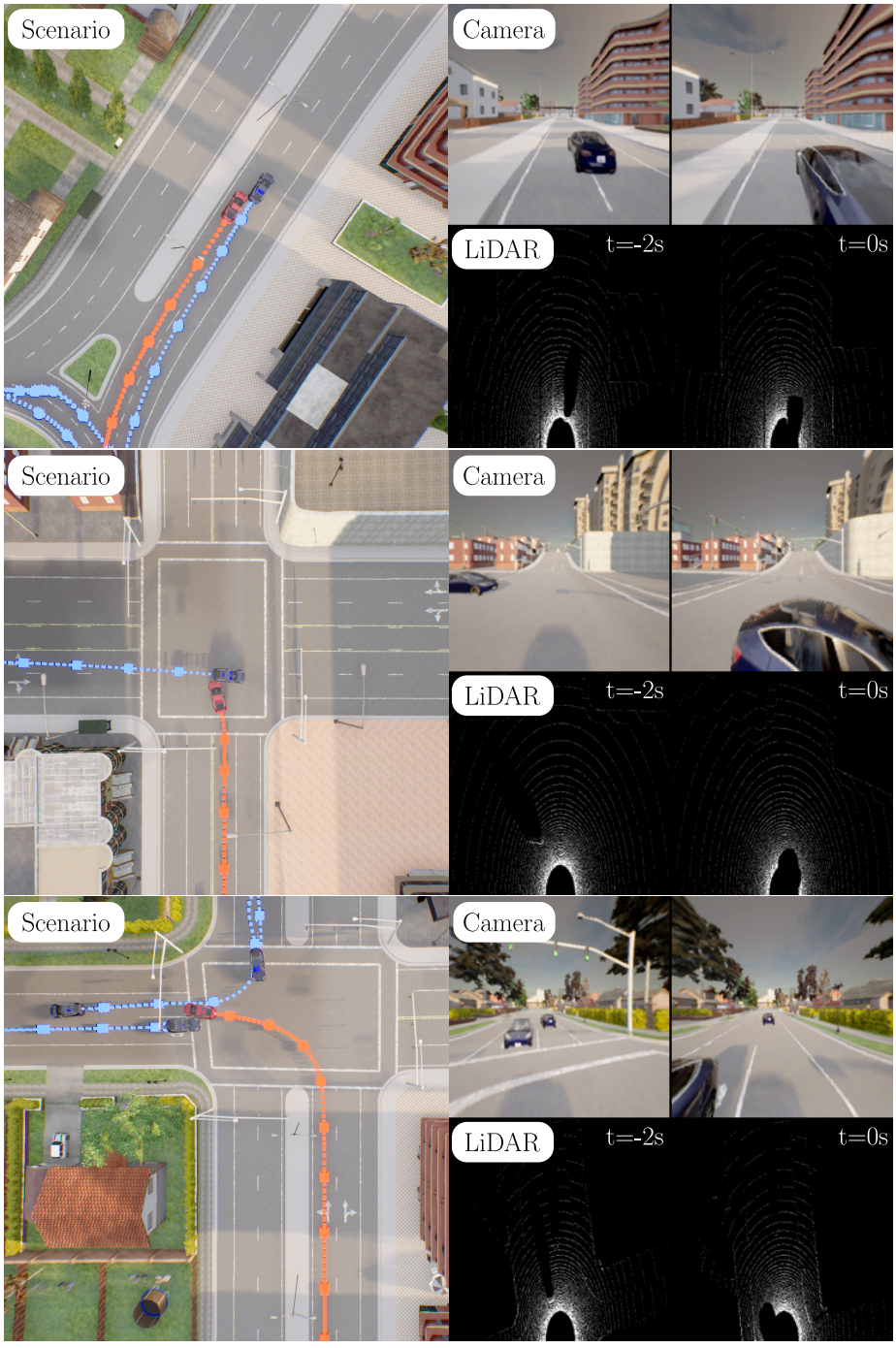


Figure 10: **Additional Qualitative Examples for TransFuser.** We show additional examples of safety-critical perturbations for TransFuser. It shares many of the failure modes of AIM-BEV, such as difficulty handling merges (top), unsafe behaviour when moving through intersections (middle) or the reliance on other vehicles strictly adhering to their lane centers and stopping points at intersections (bottom). The common failure modes of AIM-BEV and TransFuser suggest that these problems indeed stem from insufficient representation in the regular traffic data obtained from CARLA.

References

[1] K. Chitta, A. Prakash, and A. Geiger. Neat: Neural attention fields for end-to-end autonomous driving. In *Proc. of the IEEE International Conf. on Computer Vision (ICCV)*, 2021. 1, 2, 3

[2] A. Prakash, K. Chitta, and A. Geiger. Multi-modal fusion transformer for end-to-end autonomous driving. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2021. 1

[3] A. Howard, M. Sandler, G. Chu, L.-C. Chen, B. Chen, M. Tan, W. Wang, Y. Zhu, R. Pang, V. Vasudevan, Q. V. Le, and H. Adam. Searching for mobilenetv3. In *Proc. of the IEEE International Conf. on Computer Vision (ICCV)*, 2019. 1

[4] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun. CARLA: An open urban driving simulator. In *Proc. Conf. on Robot Learning (CoRL)*, 2017. 1

[5] D. Chen, B. Zhou, V. Koltun, and P. Krähenbühl. Learning by cheating. In *Proc. Conf. on Robot Learning (CoRL)*, 2019. 2, 3

[6] A. Prakash, A. Behl, E. Ohn-Bar, K. Chitta, and A. Geiger. Exploring data aggregation in policy learning for vision-based urban autonomous driving. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2020. 2

[7] D. Chen, V. Koltun, and P. Krähenbühl. Learning to drive from a world on rails. In *Proc. of the IEEE International Conf. on Computer Vision (ICCV)*, 2021. 4

[8] C. Guo, J. R. Gardner, Y. You, A. G. Wilson, and K. Q. Weinberger. Simple black-box adversarial attacks. In *Proc. of the International Conf. on Machine learning (ICML)*, 2019. 7

[9] N. Hansen and A. Ostermeier. Completely derandomized self-adaptation in evolution strategies. *Evolutionary Computation*, 2001. 7