

Approach	Density Estimation	Action Sampling Density
Decision Transformers (DTs)	$\log p_\theta(a_t o_t, G_t)$	$p_\theta(a_t o_t, G_t)$
Reward Weighted Regression (RWR)	$\exp(\eta^{-1}G_t) \log p_\theta(a_t o_t)$	$p_\theta(a_t o_t)$
Reward Conditioned Policies (RCPs)	$\log p_\theta(a_t o_t, G_t)p_\theta(G_t o_t)$	$p_\theta(a_t o_t, G_t)p_\theta(G_t o_t) \exp(\kappa G_t - \eta(\kappa))$
Reweighted Behavior Cloning (RBC)	$\log p_\theta(G_t o_t, a_t)p_\theta(a_t o_t)$	$p_\theta(G_t o_t, a_t)p_\theta(a_t o_t) \exp(\kappa G_t - \eta(\kappa))$
Implicit RL via SL (IRvS)	$\log p_\theta(a_t, G_t o_t)$	$p_\theta(a_t, G_t o_t) \exp(\kappa G_t - \eta(\kappa))$
Model-Based RCPs (MB-RCP)	$\log p_\theta(o_{t+1} a_t, o_t, G_t)p_\theta(a_t o_t, G_t)p_\theta(G_t o_t)$	$p_\theta(a_t o_t, G_t)p_\theta(G_t o_t) \exp(\kappa G_t - \eta(\kappa))$
RCP with Future Rollout (CtRL-Sim)	$\log p_\theta(s_{t+1:T} a_t, s_t, G_t)p_\theta(a_t s_t, G_t)p_\theta(G_t s_t)$	$p_\theta(a_t s_t, G_t)p_\theta(G_t s_t) \exp(\kappa G_t - \eta(\kappa))$

Table 3: Offline policy modelling approaches in prior work. We can see that methods differ in the decomposition of the joint distribution over actions and returns, with some approaches utilizing state prediction as a regularizer. We note that this table is adopted from prior work [18, 17].

A Offline RL Approaches

Table 3 presents the different ways explored in the literature for learning policies in offline RL, and how these methods can sample return-maximizing actions at test time.

B Action Sampling Algorithm

Algorithm 2 describes the proposed action sampling procedure for controllable behaviour generation with factorized exponential tilting.

Algorithm 2 The action sampling algorithm used by CtRL-Sim to allow for factorized tilting of the exhibited behaviour.

```

1: Input:  $\{\kappa^1, \dots, \kappa^C\}$   $\triangleright$  The specified inverse temperature for each return-to-go component.
2: for  $c = 1$  to  $C$  do
3:    $G_t'^c \sim p_\theta(G_t^c|s_t, s_G) \exp(\kappa^c G_t^c)$ 
4: end for
5:  $a_t \sim \pi_\theta(a_t|s_t, s_G, G_t'^1, \dots, G_t'^C)$ 
6: return  $a_t$ 

```

C Nocturne Physics Simulator and Offline RL Dataset

C.1 Physics-based Nocturne Simulator

CtRL-Sim extends the Nocturne simulation environment [13]. Nocturne is a lightweight 2D driving simulator that is built on real-world driving trajectory data from the Waymo Open Motion Dataset [14]. A scene in Nocturne is represented by a set of dynamic objects – such as vehicles, pedestrians, and cyclists – and the map context, which includes lane boundaries, lane markings, traffic signs, and crosswalks. Each dynamic object is prescribed a goal state, which is defined as the final waypoint in the ground-truth trajectory from the Waymo Open Motion Dataset. If there exist missing timesteps in the ground-truth trajectory, we re-define the goal as the waypoint immediately preceding the first missing timestep. By default, the dynamic objects track its 9 second trajectory from the Waymo Open Motion Dataset at 10 Hz. The Nocturne Simulator is originally designed for the development of RL driving policies, where the first 10 simulation steps (1s) of context is provided and the RL agent must reach the prescribed goal within the next 80 simulation steps (8s).

We extend Nocturne by integrating a physics engine based on the Box2D library for enabling realistic vehicle dynamics and vehicle collisions. We model the vehicle’s dynamics using basic physics principles, where forces applied to the vehicle are translated into acceleration, influencing its speed and direction, and with frictional forces applied to simulate realistic sliding and adherence behaviors. This extension additionally ensures that an agent’s acceleration, braking, and turn radius are bound by plausible limits and that vehicles can physically collide with each other. Such improvements open the possibility of more accurately simulating complex conditions, such as emergency braking maneuvers, slippery roads, and multi-vehicle collisions.

C.2 Offline RL Dataset Collection

The actions are defined by the acceleration and steering angle and the reward function is decomposed into three components: a goal position reward, a vehicle to vehicle collision reward, and a vehicle to road edge collision reward. We confirm that the trajectory rollouts obtained by feeding Waymo scenes through the simulator attain a reasonable reconstruction of the ground-truth Waymo trajectories (see Table 1). Following Nocturne [13], we omit bicyclist and pedestrian trajectories from the Waymo Open Motion Dataset and we omit scenes containing traffic lights. This yields a training, validation, and test set containing 134150, 9678, and 2492 scenes.

For each agent, to obtain the action a_t at timestep t , we compute the acceleration and steering value using an inverse bicycle model computed from the agent’s current state in the simulator \hat{s}_t and the ground-truth next state from the trajectory driving log s_{t+1} . We clip acceleration values between -10 and 10 and steering values between -0.7 and 0.7 radians. We then execute a_t with our proposed forward physics dynamics model to obtain the agent’s updated state \hat{s}_{t+1} , and we repeat until the agent has completed the full rollout. Table 1 confirms that this approach to offline RL trajectory data collection yields a reasonable reconstruction of the ground-truth driving trajectories.

We compute rewards at each timestep, where our reward function is factored into three rewards components: a *goal position* reward, *vehicle-vehicle collision* reward, and *vehicle-road-edge collision* reward. The goal position reward is defined by:

$$R_g(s_t, s_G) = \mathbb{1}_{\text{goal achieved}}(s_t, s_G),$$

where $\text{goal achieved}(\cdot)$ is 1 if the agent ever reaches within 1 metre of the ground-truth goal, and 0 otherwise. The vehicle-vehicle collision reward is defined by:

$$R_v(s_t, \mathbb{S}_t - \{s_t\}) = -10 \times \mathbb{1}_{\text{vehicle-vehicle collision}}(s_t, \mathbb{S}_t - \{s_t\}) + \frac{\min(\text{dist-nearest-vehicle}(s_t, \mathbb{S}_t - \{s_t\}), 15)}{15},$$

where $\text{dist-nearest-vehicle}(\cdot)$ computes the distance between the agent of interest and its nearest agent in the scene. Finally, the vehicle-road-edge collision reward is defined by:

$$R_e(s_t, m) = -10 \times \mathbb{1}_{\text{vehicle-road-edge collision}}(s_t, m) + \frac{\min(\text{dist-nearest-road-edge}(s_t, m), 5)}{5},$$

where $\text{dist-nearest-road-edge}(\cdot)$ computes the distance between the agent and the nearest road edge.

D Evaluation Metrics

The goal success rate is the proportion of evaluated agents across the evaluated test scenes that get within 1 metre of the ground-truth goal position at any point during the trajectory rollout. The final and average displacement errors are calculated for all evaluated agents across the test scenes and averaged. For a specific scene s , the collision rate and offroad rate of s are the proportion of evaluated agents in s that collide with another agent or road edge, respectively. These rates are then averaged across all tested scenes to define the overall collision and offroad rates.

We compute the Jensen Shannon Distance (JSD) between the distributions of features computed from the real and simulated rollouts. The Jensen Shannon Distance between two normalized histograms p and q is computed as:

$$\sqrt{\frac{D_{\text{KL}}(p||m) + D_{\text{KL}}(q||m)}{2}},$$

where m is the pointwise mean of p and q and D_{KL} is the KL-divergence. Unlike prior works that compute the Jensen Shannon Divergence [6, 7], we compute its square root – the Jensen Shannon Distance – so that values are not too close to 0. We compute the JSD over the following feature distributions: linear speed, angular speed, acceleration, and nearest distance. Since the acceleration values are discrete, for the acceleration JSD, we define one histogram bin for each valid acceleration

Method	JSD ($\times 10^{-2}$)				
	Lin. Speed	Ang. Speed	Accel.	Nearest Dist.	Meta-JSD
Replay-Physics*	0.1	11.5	17.4	1.2	7.6
Actions-Only [9]	4.1 ± 0.7	16.8 ± 0.3	15.6 ± 0.5	5.1 ± 0.5	10.4 ± 0.3
Imitation Learning	1.0 ± 0.1	13.4 ± 0.3	16.9 ± 0.4	2.1 ± 0.2	8.3 ± 0.1
DT (Max Return) [15]	2.7 ± 0.1	<u>13.4 ± 0.2</u>	15.4 ± 0.5	2.2 ± 0.3	8.4 ± 0.1
CTG++ [11]	3.2 ± 0.9	11.9 ± 0.9	12.4 ± 0.4	2.2 ± 0.3	7.4 ± 0.2
CtRL-Sim (No State Prediction)	1.2 ± 0.1	13.7 ± 0.2	15.9 ± 0.7	<u>2.0 ± 0.2</u>	8.2 ± 0.2
CtRL-Sim (Base)	<u>1.1 ± 0.2</u>	13.8 ± 0.2	15.6 ± 0.5	<u>2.0 ± 0.3</u>	8.1 ± 0.2
CtRL-Sim (Positive Tilting)	<u>1.4 ± 0.1</u>	13.6 ± 0.2	<u>14.8 ± 0.5</u>	1.8 ± 0.2	<u>7.9 ± 0.1</u>
DT* (GT Initial Return)	1.1 ± 0.2	13.4 ± 0.2	16.8 ± 0.6	2.1 ± 0.2	8.4 ± 0.1
CtRL-Sim* (GT Initial Return)	1.1 ± 0.2	13.8 ± 0.3	15.3 ± 0.6	2.2 ± 0.2	8.1 ± 0.2

Table 4: Breakdown of Meta-JSD in Table 1. For each metric, the best unprivileged method is **bolded** and second-best is underlined. * denotes a privileged method requiring the ground-truth future trajectory.

value, yielding 21 evenly spaced bins between -10 and 10. For the linear speed histogram, we use 200 uniformly spaced bins between 0 and 30. For the angular speed JSD, we use 200 uniformly spaced bins between -50 and 50. For the nearest distance JSD, we use 200 uniformly spaced bins between 0 and 40.

E Individual JSD Results

In Table 4, we report the per-feature JSD results for Table 1.

F CtRL-Sim Training and Inference Details

Training: The CtRL-Sim behaviour simulation model is trained using randomly subsampled sequences of length of $H \times N \times 3$, where $H = 32$ and $N = 24$. For the actions, we discretize the acceleration and steering into 20 and 50 uniformly quantized bins, respectively, yielding 1000 action tokens. For the return-to-gos, we discretize each return-to-go component $G_t^{c,i}$ into 350 uniformly quantized bins. All agents and the map context are encoded in global frame as in [28, 9] where we center and rotate the scene on a random agent during training. The map context is represented as a set of road segments $m := \{\mathbf{r}_l\}_{l=1}^L$, where each road segment is defined by a sequence of points $\mathbf{r}_l := (p_l^1, \dots, p_l^P)$, where L is the number of road segments and P is the number of points per road segment. We apply a per-point MLP to the points of each road segment \mathbf{r}_l . To produce road segment-level embeddings, we then apply attention-based pooling [45] on the embeddings of the points within each road segment, yielding L road segment embeddings of size d . We select the $L = 200$ closest lane segments within 100 metres of the centered agent as the map context, and select up to $N = 24$ closest agents within 60 metres of the centered agent as social context for the model. For each lane segment, we subsample $P = 100$ points. We use a hidden dimension size $d = 256$, where we use $E = 2$ Transformer encoder blocks and $D = 4$ Transformer decoder blocks, and we set $\alpha = \frac{1}{100}$ in the loss function. We supervise our model only on the trajectories of moving agents. We found it useful to employ *goal dropout* whereby the embeddings for 10% of agent goals are randomly set to 0 to prevent the model from overrelying on the goal information. We found goal dropout useful for learning an informative map representation. The state, return, and action embeddings for the missing timesteps are set to 0. To ensure that the model is permutation equivariant to the agent ordering [28, 27], we modify the standard temporally causal mask by additionally enforcing that each agent can only attend to its own action and return-to-go tokens at the present timestep while allowing access to all agents' state tokens at the present timestep and all agents' tokens in the past timesteps. The CtRL-Sim model is trained using a linear decaying learning rate schedule from $5e-4$ for 200k steps using the AdamW optimizer and a batch size of 64. At inference, we sample

actions with a temperature of 1.5. The CtRL-Sim architecture comprises 8.3 million parameters that we train in 20 hours with 4 NVIDIA A100 GPUs.

Inference: CtRL-Sim supports scenes with an arbitrary number of agents. As CtRL-Sim is trained with up to $N = 24$ agents, when the number of CtRL-Sim-controlled agents at inference time exceeds $N = 24$, we iteratively select 24-agent subsets at each timestep for processing until all agents have been processed. We first randomly select a CtRL-Sim-controlled agent, we normalize the scene to this agent and select the 23 closest context agents to the CtRL-Sim-controlled agent to comprise the first set of 24 agents. We then iteratively continue centering on a CtRL-Sim-controlled agent *that has not been processed in the previous sets of 24 agents* and select its 23 closest agents for context until all CtRL-Sim-controlled agents have been included in a 24-agent subset. If an agent belongs to multiple 24-agent subsets, we use the model’s first prediction of that agent. At inference time, the context length is set to training context length $H = 32$. At each timestep, we select $H = 32$ most recent timesteps as context and we found it useful to always center and rotate the scene on the centered agent at the oldest timestep in the context. For the first 10 timesteps (1s) of the simulated rollout, the states and actions are fixed to the ground-truth states and actions from the offline RL dataset, whereas the return-to-go is predicted at every timestep of the simulated rollout.

G Baseline Details

In this section, we describe the design decision of each baseline employed in our work. We note that for all models below, we scaled them in order for all models to have approximately the same number of learnable parameters as CtRL-Sim’s architecture.

Actions Only

The *actions-only* baseline is encoder-decoder architecture implemented in exactly the same way as CtRL-Sim with a few ablations. These include removal of states and returns from the decoder sequence, and no state rollout predictions. This model was inspired by [9] but differs in that the model also has access to the agents’ goals.

Imitation Learning

The *imitation learning* baseline is also based on the CtRL-Sim multi-agent behaviour simulation architecture but lacks factorized return information. It is a step better than the *actions-only* baseline since it considers the states in the decoder sequence, and this is corroborated by its improved performance on multi-agent simulation results of Table 1.

DT

The *Decision Transformer* baseline is based on the seminal work [15]. We adopt an identical architecture to CtRL-Sim’s with some minor difference based on the algorithm. One such decision is the lack of a return prediction based on states, and instead returns are chosen at inference time based on domain knowledge. Returns are the first token fed to the decoder, followed by states in order to predict actions. We make the strong argument that this is suboptimal for controllability (results in Figure 4) and does not provide intuitive mechanism for selecting the return values to target.

CTG++

The *CTG++* baseline is a diffusion model reimplementations of the recent work by [11]. We attempted to follow the architecture as closely as possible with a few minor differences. One such difference is that we diffuse over both states and actions, rather than diffusing over only actions and using an unicycle dynamics model to derive the states. We chose this approach because the underlying dynamics of the physics-enhanced Nocturne simulator is not necessarily governed by a unicycle dynamics model, and thus using a unicycle dynamics model would induce small errors in the derived states during training. We further note that we cannot replace the unicycle dynamics model with the Nocturne physics dynamics model as this forward model is not differentiable. We also condition on the present timestep and goal, to ensure fair comparison with CtRL-Sim. We note that at scene

Method	ADE (m) ↓	FDE (m) ↓	Goal Success Rate (%) ↑	JSD ($\times 10^{-2}$) ↓	Collision (%) ↓	Off Road (%) ↓	Per Scene Gen. Time (s) ↓
CTG++ [†]	1.72	3.97	41.7	7.7	6.4	17.4	44.0
CTG++ (128 hidden dim)	1.83	4.32	37.8	7.6	7.0	17.7	25.0
CTG++ (100 diffusion steps)	1.87	3.98	43.0	8.6	7.9	16.8	140.0

Table 5: **Ablations of CTG++ on Multi-agent simulation results over 1000 test scenes.** We report the results across all metrics of different configurations of the CTG++ baseline [11]. [†] indicates the original model results, with 256 hidden dimension and 50 diffusion steps at inference time.

generation time, we diffuse over actions and states at a rate of 2 Hz which is consistent with the original CTG++ model. In addition, although we train with 100 diffusion steps, we run evaluations with 50 diffusion steps. As showing in Table 5, the difference in performance is insignificant. As we do not have information on the size of the network used in the original manuscript, we explored different hidden dimension sizes of the transformer architecture of the diffusion model, also shown in Table 5. A final difference is the future relative encoding. While CTG++ use the ground-truth to compute the relative encoding during training and a constant velocity model at test time, we opted to use the final historical timestep’s relative encoding. We found this approach to be more stable.

H CAT Simulated Data Collection and CtRL-Sim Finetuning

The Waymo Open Motion dataset largely contains nominal driving scenes. To enhance control over the generation of safety-critical scenarios, we finetune CtRL-Sim on a simulated dataset of safety-critical scenarios generated by CAT [21]. CAT is a state-of-the-art collision generation method that involves fixing the agent’s future trajectory to a trajectory predicted by a DenseTNT trajectory predictor. CAT searches for a trajectory that has high likelihood of colliding with the log-replay future trajectory of the ego vehicle, while having high probability under the behaviour prior (DenseTNT). For more details, we refer readers to [21]. We note that a limitation of CAT is that the agent is non-reactive to the ego as the agent’s trajectory is fixed at the beginning of the simulation. Moreover, unlike CtRL-Sim, CAT does not have control over the degree to which the agent is adversarial.

To collect the simulated safety-critical dataset, we run CAT on a subset of the interactive validation split of the Waymo Open Motion Dataset, which involves two interacting agents. Following CAT, we select one of the two interacting agents to be the ego (whose trajectory is fixed to the log-replay trajectory) and the other interacting agent to be the CAT adversary. In total, we collect 3577 CAT scenarios for finetuning, of which around 60% contain ego-adversary collisions.

To encourage CtRL-Sim to learn how to generate safety-critical scenarios without forgetting how to generate good driving behaviour, we adopt a continual pre-training strategy for finetuning [46] where we randomly sample 3577 real training scenarios from the offline RL dataset in each training epoch, or a 50% replay ratio. We rewarm the learning rate to the maximum learning rate of $5e-4$ over 500 steps and follow a linear decay learning rate schedule to 0 over 20 epochs. We expect that the finetuned CtRL-Sim model will be more capable of generating long-tail scenarios as it is more exposed to such scenarios during finetuning. Finetuning takes roughly 30 minutes on 1 NVIDIA A100-Large GPU.

Method	Times Preferred
CtRL-Sim	15
CAT	8
Tie	2

Table 6: **The tally of votes for the pilot study.** We show the breakdown of the votes for the conducted pilot study. Strong preference in terms of plausibility is shown for scenarios generated using CtRL-Sim.

I Adversarial Scenario Generation User Study

We first conduct a pilot study with 5 participants to evaluate which method (CtRL-Sim vs. CAT) generates more plausible adversarial behaviours. Each participant was tasked with evaluating the

Survey

What collision scenario do you find more plausible given the behaviour of the pink agent?



Figure 6: **User Study Example Scenario.** We show an example of a pair of scenarios along with the question users are asked to answer.

624 same set of 25 pairs of scenarios that were randomly selected from a pool of 259 scenarios, where
 625 one scenario employed a CtRL-Sim adversarial agent and the other was a CAT adversarial agent. In
 626 both instances, the planner was a positively tilted CtRL-Sim planner. Table 6 shows the tally of the
 627 votes, where we observed strong preference for adversarial scenarios generated using CtRL-Sim.

628 We conduct a larger study including a total of 22 partic-
 629 ipants. We did not record any identifying information of
 630 the participants, and participants were invited on a vol-
 631 untary basis. Users were presented with the user inter-
 632 face shown in Figure 6, where we randomize the order
 633 between both videos. Each participant responded to a to-
 634 tal of 30 pairs of scenarios, which took an average 12
 635 minutes to complete. These pairs were randomly selected
 636 from the same pool of 259 scenarios as the pilot study. Ta-
 637 ble 7 shows the tally of this larger study. As we can see,
 638 the results of this larger study are inconclusive compared
 639 to the pilot study. We hypothesize that the framing of the
 640 question may have been ambiguous for a larger audience,
 641 and that the pairs of videos may have included less interesting scenarios (e.g, adversary rear-ending
 642 planner), resulting in many comparisons that were very difficult to judge, thus adding noise to the
 643 survey results. We plan to further study the question of plausibility in a future user study with better
 644 scenario selection.

Method	Times Preferred
CtRL-Sim	97
CAT	96
Tie	66

Table 7: **The tally of votes for the larger study.** We show the breakdown of the votes for the conducted larger study. We observe that the results of this study are less conclusive than the pilot study.

645 J Additional Qualitative Results

646 Figure 7 shows more qualitative examples demonstrating the effects of positive exponential tilting
 647 on each of the three reward components. In the left panels, CtRL-Sim with no tilting produces a

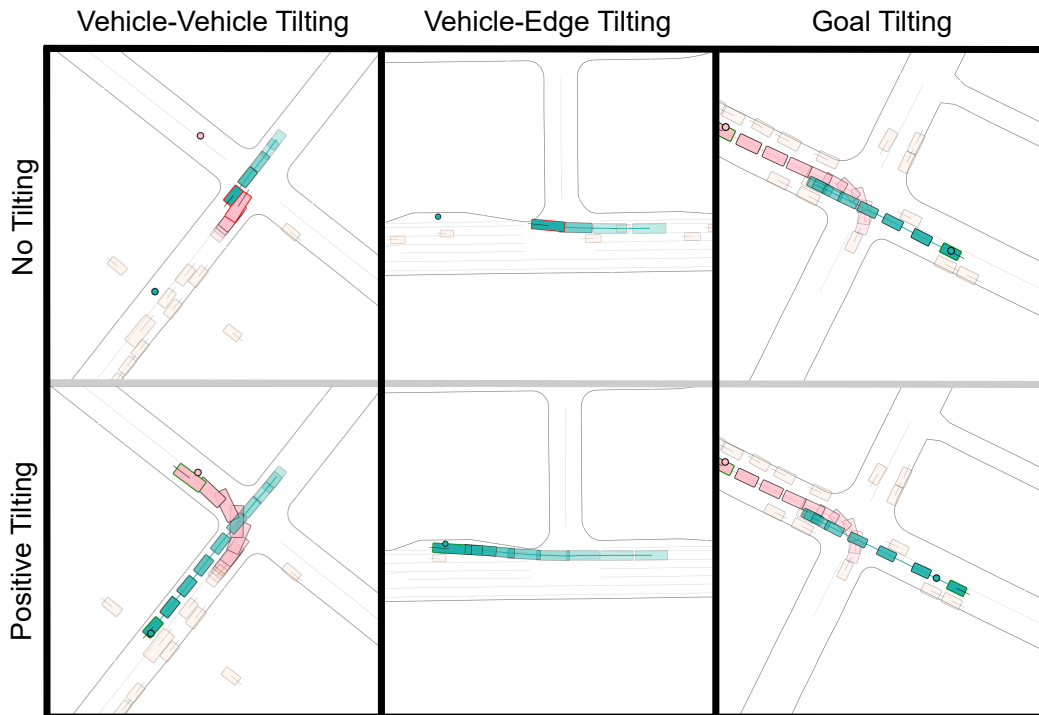


Figure 7: **Qualitative results of the effects of positive tilting.** We show the evolution of three traffic scenes with the top panels applying no exponential tilting to the CtRL-Sim-controlled agent (shown in teal) and the bottom panels applying positive tilting to the same CtRL-Sim-controlled agent. Bounding boxes outlined in red contain a traffic violation. All other agents are set to log-replay through physics, with the agent interacting with the CtRL-Sim-controlled agent denoted in pink. Goals are denoted by small circles.

648 vehicle-vehicle collision between two interacting agents at a left-turn. With positive vehicle-vehicle
649 tilting, the CtRL-Sim-controlled agent moves more to the right-hand side of the lane to avoid the
650 collision. In the middle panels, CtRL-Sim with no tilting produces a vehicle-edge collision as the bus
651 pulls into the curb. With positive vehicle-edge tilting, the CtRL-Sim-controlled agent pulls into the
652 curb at a safer distance from the curb. In the right panels, CtRL-Sim with no tilting reaches the goal.
653 With positive goal tilting, the CtRL-Sim-controlled agent reaches the goal much faster and nearly
654 avoids collision with the turning vehicle. In Figure 8, we show two more examples of adversarial
655 collision scenarios generated with negative vehicle-vehicle tilting. We refer the interested reader to
656 the supplementary video for more examples.

657 K Multi-Agent Simulation Results with Higher Temperature Sampling

658 In Table 8, we report results from the same experiments as Table 1 except with a higher action
659 sampling temperature, set to 1.5.

660 L Fine-tuning CtRL-Sim on CtRL-Sim Scenarios

661 Instead of finetuning on CAT scenarios, we explore finetuning CtRL-Sim on adversarial scenar-
662 ios generated by CtRL-Sim. We first collect a simulated dataset of scenes either containing a
663 vehicle-vehicle collision or an offroad infraction. Specifically, we generate rollouts of a single
664 agent with the negatively tilted base CtRL-Sim model where the other agents are set to log re-
665 play through physics, and we save the scenario only if the generated rollout yields a vehicle-vehicle

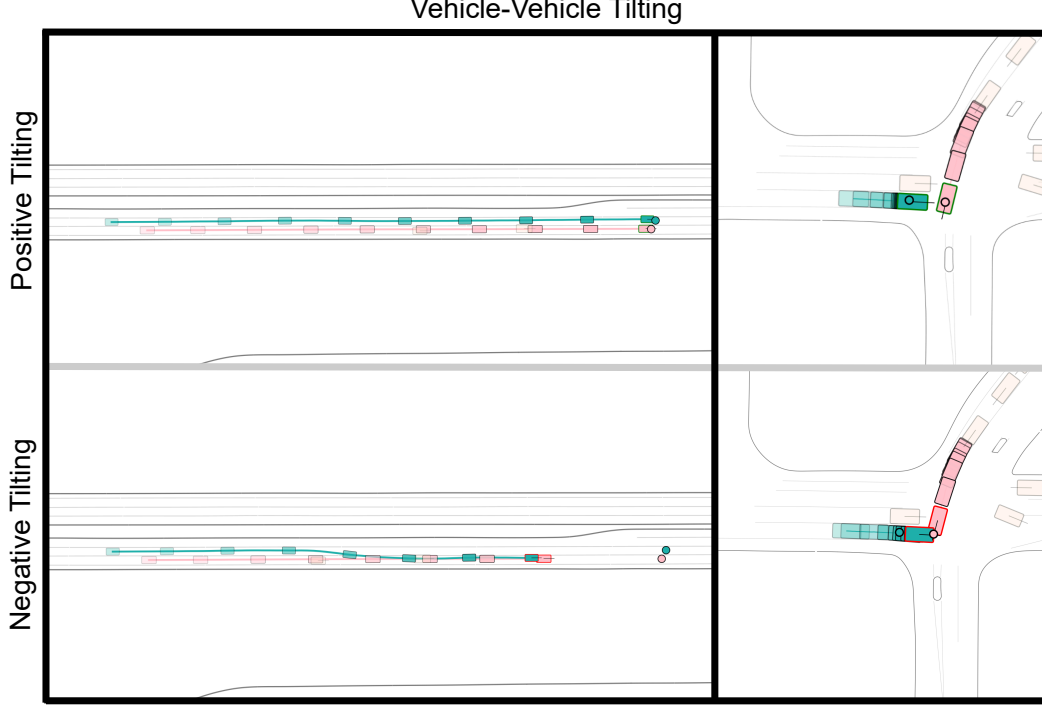


Figure 8: **Qualitative results of vehicle-vehicle tilting.** We show the evolution of two traffic scenes with the top panels applying positive exponential tilting to the CtRL-Sim-controlled agent (shown in teal) and the bottom panels applying negative tilting to the same CtRL-Sim-controlled agent. Bounding boxes outlined in red contain a traffic violation. All other agents are set to log-replay through physics, with the agent interacting with the CtRL-Sim-controlled agent denoted in pink. Goals are denoted by small circles.

Method	FDE (m)	Reconstruction ADE (m)	Goal Suc. Rate (%)	Distributional Realism Meta JSD($\times 10^{-2}$)	Common Sense Collision (%)	Off Road (%)
Replay-Physics*	0.97	0.47	87.3	7.6	2.8	10.7
Actions-Only [9]	11.70 ± 1.12	4.78 ± 0.42	34.4 ± 1.3	14.3 ± 0.3	22.8 ± 0.7	29.7 ± 1.7
Imitation Learning	2.42 ± 0.17	1.47 ± 0.07	73.8 ± 1.2	12.3 ± 0.5	7.3 ± 0.6	13.1 ± 0.4
DT (Max Return) [15]	3.25 ± 0.17	1.67 ± 0.05	60.5 ± 1.2	12.3 ± 0.4	6.1 ± 0.7	11.6 ± 0.3
CtRL-Sim (No State Prediction)	2.57 ± 0.16	1.52 ± 0.07	66.2 ± 1.0	12.3 ± 0.3	7.6 ± 0.7	13.1 ± 0.3
CtRL-Sim (Base)	2.49 ± 0.10	1.50 ± 0.04	67.9 ± 1.2	12.2 ± 0.2	7.6 ± 0.3	13.1 ± 0.5
CtRL-Sim (Positive Tilting)	2.38 ± 0.08	1.44 ± 0.03	67.2 ± 1.0	12.1 ± 0.1	6.7 ± 0.4	12.3 ± 0.3
DT* (GT Initial Return)	1.94 ± 0.07	1.28 ± 0.02	73.7 ± 1.5	12.2 ± 0.3	6.6 ± 0.4	12.6 ± 0.4
CtRL-Sim* (GT Initial Return)	1.97 ± 0.08	1.30 ± 0.03	71.1 ± 0.9	12.2 ± 0.1	7.2 ± 0.5	13.1 ± 0.3

Table 8: **Multi-agent simulation results over 1000 test scenes with action temperature = 1.5 over 3 seeds.** This table presents the results from the same experiments as Table 1, but with an action sampling temperature of 1.5 instead of 1.0. This allows for a comparison of the impact of the temperature hyperparameter. Overall, an action sampling temperature of 1.0 yields better results.

collision or vehicle-road-edge collision. For tilting, we uniformly sample $\kappa_{\text{veh-veh}} \sim \mathcal{U}(-25, 0)$ and $\kappa_{\text{goal}} \sim \mathcal{U}(-25, 0)$ when generating vehicle-vehicle collision scenarios, and we uniformly sample $\kappa_{\text{veh-edge}} \sim \mathcal{U}(-25, 0)$ and $\kappa_{\text{goal}} \sim \mathcal{U}(-25, 0)$ when generating vehicle-road-edge collision scenarios. By additionally negatively tilting the goal, this grants the model more flexibility when generating traffic violations as the agents are not trying to reach its prescribed goal. We collect 5000 scenarios of each type of traffic violation derived from the training set, which comprises the simulated dataset of safety-critical scenarios. To encourage CtRL-Sim to learn how to generate safety-critical scenarios without forgetting how to generate good driving behaviour, we adopt the same finetuning strategy as in Appendix H, except we randomly sample 90000 real training scenarios from the of-

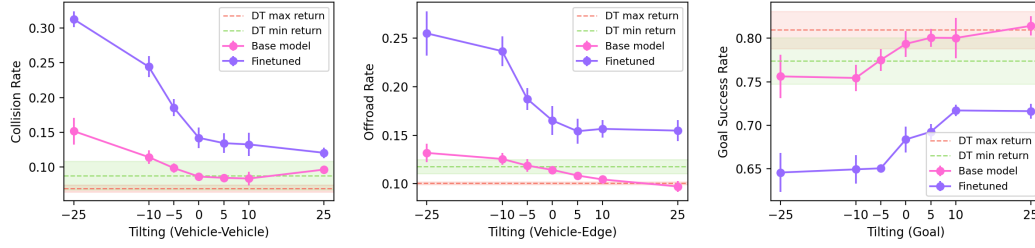


Figure 9: **Effects of exponential tilting.** Comparison of CtRL-Sim base model (magenta) and a CtRL-Sim model fine-tuned on adversarial CtRL-Sim scenarios (purple). As opposed to Figure 4, this fine-tuned model does not involve using CAT to select the adversarial scenarios. Rewards range from -25 to 25 for vehicle-vehicle collision (left), vehicle-edge collision (middle), and goal reaching (right). Results show smooth controllability, with fine-tuning enhancing this effect. Mean and std are reported over 5 seeds.

675 fine RL dataset in each training epoch, or a 90% replay ratio. We find it useful to use a larger replay
676 ratio when finetuning on CtRL-Sim scenarios. The controllability results over 5 seeds are shown in
677 Figure 9, demonstrating similar control over adversarial behaviours as the CAT-finetuned CtRL-Sim
678 model.