MIXSIM: A Hierarchical Framework for Mixed Reality Traffic Simulation Supplementary Materials

Abstract

We present additional results and analyses (Appendix A), MIXSIM implementation details (Appendix B), and experiment details (Appendix C). We also attach a video that provides a brief outline of MIXSIM and additional qualitative results. See `supplementary_video.mp4`.

A. Additional Results

A.1. Comparison to Motion Forecasting Models

Motion forecasting is a related field that also requires accurate modeling of human driving behaviors. As such, we compare MIXSIM against state-of-the-art motion forecasting models on their ability to reconstruct a given scenario when re-simulating all interactive agents, mirroring the realism analysis presented in the main paper. In particular, we compare against our implementations of MTP [4], MultiPath [2], LaneGCN [10], and GoRela [3]. These models output multi-modal waypoint trajectories. Therefore, at each step of the simulation, we use each agent's highest scoring trajectory to determine which waypoint the agent will arrive at next.

Tab. 1 summarizes the results of our experiment on AV2. We observe that all motion forecasting models perform significantly worse than MIXSIM, particularly on reconstruction metrics. This result can be explained by: (1) these models cannot condition on ground truth paths; and (2) like BC, these models are trained in open-loop and suffer from significant covariate shift when deployed in closed-loop simulation. These conclusions are especially evident considering the fact that MIXSIM share a similar architecture with GoRela [3], differing only in terms of our training methodology (*i.e.*, closed-loop training) and controllability (*i.e.*, route-based hierarchy). This demonstrates the importance of these design choices for *mixed reality traffic simulation*.

A.2. Ablation Study

We conduct an ablation study of the two most prominent design choices in MIXSIM: (1) the route-based hierarchy (and its associated route-conditional policy); and (2) closed-loop training with an imitation objective. We have already partly demonstrated their significance via a comparison against the BC and IL baselines in the main paper. Here, we complete the picture by introducing an additional method: **BC-Route**. Concretely, BC-Route uses an *identical* architecture as MIXSIM (including our HeteroGNN based route-conditional policy decoder) and uses the same reconstructed route for reactive re-simulation. The only distinction is training methodology, where BC-Route is trained using one-step behavior cloning similar to BC.

Tab. 2 shows clear evidence that both route-conditioning and closed-loop training are critical for simulation performance. Notably, route-conditioning alone can significantly reduce compounding error, approximately halving reconstruction error and collision rate and preventing agents from driving off road in most situations. However, it remains critical to optimize the imitation objective on unrolled simulation states instead of one-step actions to further reduce the sim2real gap.

A.3. Reconstruction Error Across Time

In Figs. 2 to 3, we plot reconstruction metrics across time for HIGHWAY. As discussed in the main paper, MIXSIM achieves the best displacement error and along track error of the competing methods—a result that holds across all time steps. Unsurprisingly, the path-following baselines achieve the lowest cross track error since they constrain agents to follow their original paths. We note, however, that these baselines do not achieve zero cross track error since: (1) they fallback to their base model for steering control after fully traversing their original paths; and (2) they do not follow the agents' original trajectories exactly but instead plan against smooth driving paths built from the original trajectories (see Appendix C.1). Due to differences between the driving paths and the original trajectories, we observe that the path-following baselines exhibit slightly lower cross track error in the middle of a simulation than at the beginning or the end.

In Figs. 4 to 5, we show similar results hold in the urban domain of AV2. Notably, MIXSIM outperforms all baselines in all three reconstruction metrics, including against path-following baselines on cross track error! We attribute

Method	Ref. Path	Reconstruction FDE	ATE	CTE	Distribution JSD Speed	Accel.	Lead Dist.	Nearest Dist.	Common Sense Collision	Off Road
MTP [4]		13.88	13.54	1.21	0.21	0.28	0.04	0.06	1.40	2.12
MultiPath [2]		11.96	11.52	1.28	0.17	0.30	0.04	0.05	1.27	2.19
LaneGCN [10]		11.91	10.10	3.74	0.16	**0.18**	0.06	0.10	2.43	17.71
GoRela [3]		11.54	10.23	2.75	0.12	0.19	0.05	0.04	**1.06**	3.85
MIXSIM	✓	**5.04**	**4.86**	**0.67**	**0.04**	0.33	**0.03**	**0.03**	1.07	**1.29**

Table 1. Sim2real domain gap results on AV2: *Ref. Path* means method condition on $\mathbf{s}_{0:T}$, otherwise only $\mathbf{s}_0$.

Method	Ablation Route-cond.	CLT	Reconstruction FDE	ATE	CTE	Distribution JSD Speed	Accel.	Lead Dist.	Nearest Dist.	Common Sense Collision	Off Road
BC			26.10	17.00	15.60	0.11	0.32	0.18	0.08	11.40	61.60
BC-Route	✓		16.40	14.80	4.27	0.09	0.11	0.10	0.05	5.31	0.32
IL		✓	11.10	10.90	1.05	0.10	0.11	0.05	**0.03**	1.49	**0.00**
MIXSIM	✓	✓	**10.30**	**10.30**	**0.25**	**0.09**	**0.10**	**0.04**	**0.03**	**0.86**	**0.00**

Table 2. Sim2real domain gap results on HIGHWAY: Our route-conditional policy decoder (Route-cond.) and closed-loop training (CLT) are critical to achieving good simulation performance.

Finds Safety Critical Variation MIXSIM	BICYCLE	BICYCLE-F	Breakdown # Runs	% Runs
			190	47.50
		✓	26	6.50
	✓		27	6.75
	✓	✓	51	12.75
✓			5	1.25
✓		✓	5	1.25
✓	✓		4	1.00
✓	✓	✓	92	23.00

Table 3. Breakdown of 400 optimization runs to find safety critical variations on HIGHWAY-ADVERSARIAL. We report the number and percentage of runs for which a subset of the methods find a safety critical variation.

this advantage to using *soft* conditioning on the ground truth route instead of *hard* constraints on the driving path built from the original trajectory. This enables MIXSIM to be more robust to noise in the original trajectory (and also noise introduced by the driving path optimization). Another notable observation is that BC outperforms Heuristic in AV2 (but not in HIGHWAY). We attribute this to two main differences: (1) AV2 has more complex map topology, which is more difficult to handle with heuristics (especially for determining neighboring agents); and (2) AV2 has a larger and more diverse set of demonstration data, which improves the (relative) performance of BC.

A.4. Distributional Realism Histograms

Figs. 6 to 9 (see the end of this document) show a comparison of the histograms used to calculate our distributional realism metrics on HIGHWAY and AV2. From Fig. 6, we observe that methods learned using closed-loop training (*i.e.*, IL-Path and MIXSIM) induce distributions that best matches that of real data. In contrast, BC-Path, which uses open-loop training, fails to accurately model speed and acceleration, reflecting its deficiencies when deployed to closed-loop simulation. Heuristic-Path is unable to accurately recover the right tails of the speed and acceleration distributions due to its rigid encoding of traffic rules such as obeying speed limits.

Fig. 7 tells a similar story, but with the differences between the unconditional baselines and MIXSIM becoming more pronounced since the baselines no longer have access to the ground truth paths. Notably, BC fails to model the distance-to-lead-agent distribution and places significantly more mass on small distances—a reflection of its inability to properly reason about agent-to-agent interactions when deployed to closed-loop simulation.

We observe similar trends in Figs. 8 to 9 for experiments on AV2. We note that all competing methods exhibit lower distributional realism in acceleration on AV2 *vs.* HIGHWAY, which reflects the fact that ground truth acceleration is derived from noise finite difference approximations in AV2.

A.5. Safety Critical Variations Success Rates

We present a more detailed analysis comparing when MIXSIM, BICYCLE, and BICYCLE-F successfully finds a safety critical variation. In particular, we categorize our 400 optimization runs by the joint success or failure of the three methods to find a safety critical variation.

From Tab. 3, we observe that the success rates of all three methods are highly correlated, demonstrating that it is more

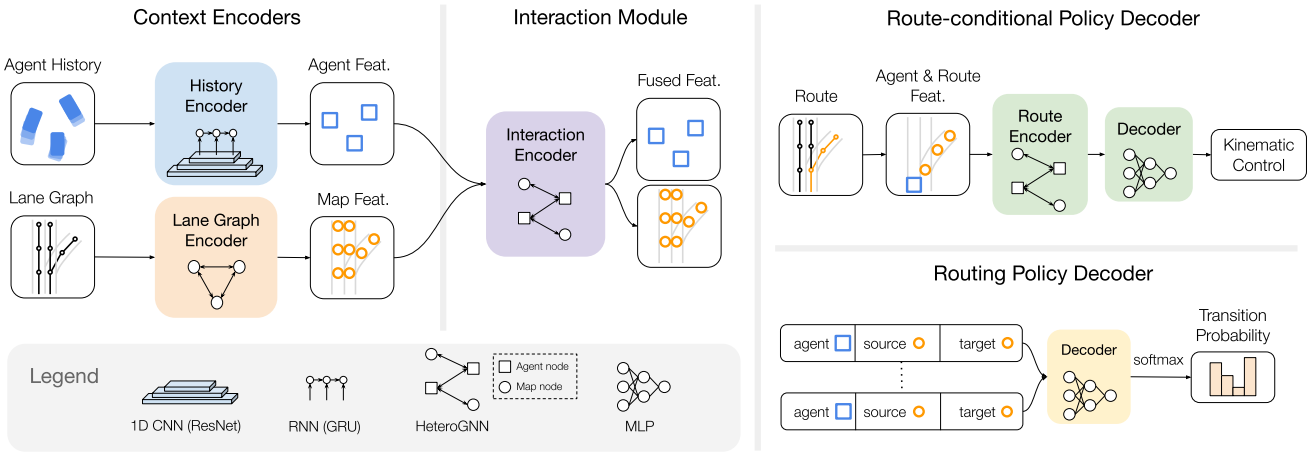

Figure 1. Overview of MIXSIM model architecture.

difficult to find safety critical behaviors for certain agents. At least one method was successful in 52.5% of runs. Almost all of the safety critical variations found by MIXSIM were also found by either or both of the baselines. This is expected, as any lane graph route can equivalently be expressed as a sequence of kinematic bicycle controls, so the latter is a superset of MIXSIM's parameterization. The cost of the more expressive parameter space appears elsewhere: the surplus of safety critical variations found by the baselines are lower in quality with respect to realism.

B. MIXSIM Implementation Details

B.1. Input Parameterization

Agent history: Following [3], we adopt an viewpoint invariant representation of an agent's past trajectory. More precisely, we encode the past trajectory as a sequence of pair-wise relative positional encodings between the past waypoints and the current pose. Each relative positional encoding is a sinusoidal encoding of the distance and heading difference between a pair of poses. See [3] for details.

Lane graph: We briefly describe how we construct our lane graph representation $G = (V, E)$. First, to obtain the lane graph nodes, we discretize the centerlines in the high-definition (HD) map into lane segments of fixed lengths (5m for AV2 and 10m for HIGHWAY). Each node is initialized with features such as length, width, curvature, speed limit, and lane boundary type (*e.g.*, solid, dashed). Following [10], we then connect nodes with 4 different relationships: successors, predecessors, left and right neighbors.

B.2. Model Architecture

We show an overview of MIXSIM model architecture in Fig. 1. Briefly, it is composed of three main building blocks: (1) context encoders for embedding lane graph and agent history inputs; (2) an interaction module for capturing scene-level interaction; and (3) policy decoders for parameterizing the route-conditional policy and routing policy. We discuss each in detail below.

History encoder: The *history encoder* is a 1D residual neural network (ResNet) followed by a gated recurrent unit (GRU) that extracts agent features $h_{\text{agent}} = f_{\text{agent}}(\mathbf{s}_t)$ from the joint agent states $\mathbf{s}_t$. Each agent's state $\mathbf{s}_{i,t}$ is a sliding window of its position, heading, 2D bounding box, and velocity over the past H time-steps. Intuitively, the 1D CNN captures local temporal patterns and the GRU aggregates them into a global feature. Unlike the map encoder, the agent history encoder is run at every step of the simulation.

Lane graph encoder: The *lane graph encoder* is a graph convolutional network (GCN) [10] that extracts map features $h_{\text{map}} = f_{\text{map}}(G)$ from a given lane graph G. We use hidden channel dimensions of [128, 128, 128, 128], layer normalization (LN), and max pooling aggregation. Since map features are static, they are computed once and cached between simulations thereafter.

Interaction module: To model scene-level interaction (*i.e.*, agent-to-agent, agent-to-map, and map-to-map), we first build a heterogeneous spatial graph G' by adding agent nodes to the original lane graph G. In addition to the original lane graph edges, we also connect agent nodes to their closest lane graph nodes and fully connect all agent nodes to each other. Then, we use a *scene encoder* parameterized by a heterogeneous graph neural network (HeteroGNN) [3] to process map features and agent features into fused features,

$$h'_{\text{map}}, h'_{\text{agent}} = f_{\text{scene}}(h_{\text{map}}, h_{\text{agent}}) \quad (1)$$

These contextual features forms the input to the route-conditional policy decoder and the routing policy decoder.

Route-conditional policy decoder: Our route-conditional policy decoder is independent per agent but computation is batched for efficiency. In the following, we describe inference for a single agent but omit the subscript i for brevity.

We first pre-process the route by truncating it to a sliding window of 10 lane graph nodes, starting at the node closest to the agent at time t. Then, we build a heterogeneous graph consisting of only the considered agent and lane graph nodes in its route. We use a HeteroGNN to fuse features into a single route feature,

$$h_{\text{route}} = \text{HeteroGNN}(\{h'_{\text{map}}(v) | v \in R\}, h'_{\text{agent}}) \quad (2)$$

and we concatenate these route features with the agent's state features h'_{agent}. Finally, we pass the concatenated features into a 4-layer MLP with hidden dimensions [128, 128, 128] to predict agent's acceleration and steering angle,

$$\mathbf{a}_t = \text{MLP}([h_{\text{route}}, h'_{\text{agent}}]) \quad (3)$$

Routing policy decoder: We use a simple MLP-based architecture to parameterize the transition probability between lane graph nodes for each agent. Concretely, for each pair of agent i and lane graph edge (u, v), we first concatenate the agent and edge features and then pass it through a 3-layer MLP with hidden dimensions [128, 64] to get a score,

$$f_{\text{score}}(u, v) = \text{MLP}([h'_{\text{map}}(u), h'_{\text{map}}(v), h'_{\text{agent}}]) \quad (4)$$

We decode the transition probability by taking a softmax over the logits of the outgoing edges,

$$h_i^{\text{edge}}(u_{j+1} | u_j, \mathbf{s}_0, \mathbf{m}; \phi) = \frac{\exp f_{\text{score}}(u_j, u_{j+1})}{\sum_{(u_j, v) \in E} \exp f_{\text{score}}(u_j, v)} \quad (5)$$

B.3. Inference

Route reconstruction for reactive re-simulation: We adapt a hidden Markov model (HMM) for map matching [12] to reconstruct each agent's route from its trajectory in the original scenario. Let $\mathbf{x}_t \in \mathbb{R}^2$ denote the agent's 2D bird's eye view position at time t. Given a lane graph $G = (V, E)$, let $\mathbf{z}_t$ denote the lane segment $u \in V$ that best explains the agent's position $\mathbf{x}_t$ at time t. We model the joint distribution over the agent's trajectory $\mathbf{x}_{0:T}$ and its associated lane segments $\mathbf{z}_{0:T}$ with an HMM, where $\mathbf{x}_{0:T}$ are the observations and $\mathbf{z}_{0:T}$ are the hidden variables,

$$p(\mathbf{x}_{0:T}, \mathbf{z}_{0:T}) = p(\mathbf{z}_0) p(\mathbf{x}_0 | \mathbf{z}_0) \prod_{t=1}^{T} p(\mathbf{z}_t | \mathbf{z}_{t-1}) p(\mathbf{x}_t | \mathbf{z}_t) \quad (6)$$

Under this formulation, we can determine the maximum a posteriori route $\mathbf{z}_0, \ldots, \mathbf{z}_T$ using the Viterbi algorithm,

$$\mathbf{z}^\star_{0:T} = \arg\max_{\mathbf{z}_{0:T}} p(\mathbf{z}_{0:T} | \mathbf{x}_{0:T}) \quad (7)$$

In our model, the emission probability $p(\mathbf{x}_t | \mathbf{z}_t)$ is proportional to the point-to-line distance from $\mathbf{x}_t$ to $\mathbf{z}_t$,

$$p(\mathbf{x}_t | \mathbf{z}_t) \propto -\min_{\mathbf{p} \in \mathbf{z}_t} \|\mathbf{x}_t - \mathbf{p}\|_2 \quad (8)$$

The transition probability $p(\mathbf{z}_t | \mathbf{z}_{t-1})$ is proportional to the absolute difference between the distance between $\mathbf{z}_{t-1}$ and $\mathbf{z}_t$ in G and the distance between $\mathbf{x}_{t-1}$ and $\mathbf{x}_t$,

$$p(\mathbf{z}_t | \mathbf{z}_{t-1}) \propto -|\text{d}_G(\mathbf{z}_{t-1}, \mathbf{z}_t) - \|\mathbf{x}_{t-1} - \mathbf{x}_t\|_2| \quad (9)$$

where $\text{d}_G(\mathbf{z}_{t-1}, \mathbf{z}_t)$ is the distance along the shortest directed path connecting $\mathbf{z}_{t-1}$ and $\mathbf{z}_t$ in G. Note that $\text{d}_G(\mathbf{z}_{t-1}, \mathbf{z}_t) = \infty$ for any pair of disconnected lane segments $\mathbf{z}_{t-1}$ and $\mathbf{z}_t$; thus, topologically infeasible transitions (and, by extension, routes) are impossible under our model.

Severity measure for finding safety critical variations: The severity measure $\mathcal{R}(\mathbf{s}_{0:T}, \mathbf{m})$ can be tuned to target different outcomes. For our experiments, we aim for simplicity and choose to target collisions between the SDV and any other agents. We use a continuous measure for collision: the "buffer" distance between (the surfaces of) the SDV and the nearest other agent,

$$\text{MinBuffer} = \min_{0:T} (\text{dist}_{\text{buffer}}(\text{SDV})) \quad (10)$$

If the minimum buffer distance during a scenario is zero, there is a collision.

However, minimum buffer distance is locally flat when the adversary A is never the closest object to the SDV, so it is difficult to optimize alone. Thus, in practice, an additional adversary-to-SDV distance term encourages the adversary to approach the SDV,

$$\text{AdvToSDV} = \max\left(\min_{0:T} \text{dist}(\text{SDV}, A) - 10, 0\right) \quad (11)$$

When close enough (within 10 metres) to produce a safety critical scenarios, the term becomes a constant.

Altogether, we minimize the sum of MinBuffer and AdvToSDV. In our formulation of finding safety critical variations as a maximization problem, the severity measure is,

$$\mathcal{R}(\mathbf{s}_{0:T}, \mathbf{m}) = -\text{MinBuffer} - \text{AdvToSDV} \quad (12)$$

Parameter	Value {*default*, *cautious*, *aggressive*}
Core IDM [14] parameters	
desired speed ($\mathrm{m\,s^{-1}}$)	speed limit
desired gap (s)	{1.5, 1.5, 0.75}
min clearance (m)	2.0
max idm accel. ($\mathrm{m\,s^{-2}}$)	{1.4, 1.4, 2.8}
comfortable decel. ($\mathrm{m\,s^{-2}}$)	{2.0, 1.4, 2.0}
free road exponent	4.0
Generalized IDM [9] parameters	
follow interaction weight	0.5
max accel. ($\mathrm{m\,s^{-2}}$)	3.0
max decel. ($\mathrm{m\,s^{-2}}$)	4.0
MOBIL [8] parameters	
politeness factor	{0.5, 1.0, 0.0}
changing threshold ($\mathrm{m\,s^{-2}}$)	0.1
max safe decel. ($\mathrm{m\,s^{-2}}$)	4.0

Table 4. Parameter setting for Heuristic and Heuristic-Path.

Feature	Bin size	Min	Max
Speed ($\mathrm{m\,s^{-1}}$)	0.5	0.0	50.0
Accel. ($\mathrm{m\,s^{-2}}$)	0.1	-10.0	10.0
Lead Dist. (m)	1.0	0.0	300.0
Nearest Dist. (m)	1.0	0.0	100.0
L. Accel. ($\mathrm{m\,s^{-2}}$)	0.1	-10.0	10.0
Curvature	0.0004	-0.02	0.02

Table 5. Histogram setting for our distributional realism metrics, used for both HIGHWAY and AV2.

C. Experiment Details

C.1. Baselines

Heuristic baselines: Our Heuristic baseline encodes normative driving behaviors (*e.g.*, collision avoidance, traffic rule compliance) by using IDM [14] for longitudinal control and MOBIL [8] for selecting target lanes. More concretely, at each planning step, we do the following: (1) associate the agent with its closest lane centerline; (2) resolve its neighbors on its current and adjacent lanes; (3) query MOBIL [8] for its target lane; (4) query IDM [14] for safe acceleration; and, lastly, (5) plan a constant acceleration lane-relative trajectory in Frenet space [16] that smoothly converges to the target lane centerline.

See Tab. 4 for detailed parameter settings for IDM and MOBIL. For our main sim2real domain gap results, we use the *default* behavior profile. To generate variations, we sample behavior profiles from {*default*, *cautious*, *aggressive*} for each agent independently at the start of the simulation.

Learned baselines: When benchmarking against learned baselines, we explicitly control for the effect of model architecture and focus our comparison to the route-based hierarchy and training methodology. Concretely, our BC and IL baselines share identical context encoders and interaction modules as MIXSIM, and they only differs from MIXSIM in their decoders. While MIXSIM uses a route-conditional decoder, BC and IL use an MLP decoder that is not conditioned on route. Concretely, we use a 2-layer MLP with hidden dimension of 128. To extend these methods for generating realistic variations, we found Monte Carlo Dropout [5] to be an effective approach without degrading performance. Concretely, we add dropout layers in only the decoder and train with dropout probability of 0.5. During inference, we enable dropout with same probability to sample actions at each simulation step.

Path-following variants: We extend the *unconditional* baselines, which only condition on the initial scenario context, to our setting where they have access to the full reference scenario. Concretely, we first fit a smooth *driving path* to the reference path from the original scenario using an ILQR-based path optimizer [1]. Then, we plan a constant acceleration lane-relative trajectory in Frenet space [16] that smoothly converges to the *driving path*. This approach has the benefit of being robust to noise in the reference paths from the original scenario and yields trajectory plans with reasonable kinematics.

C.2. Dataset

HIGHWAY: Our first dataset is a collection of simulated highway traffic scenarios. It consists of 1000 scenarios, which we split into a training set of 800 and an evaluation set of 200. Each scenario provides bounding box trajectories for every actor and a high definition map of the local road topology. Together, the scenarios cover a representative range of highway road topologies, traffic conditions, and actor interactions. In our experiments, the first 3s of each scenario is given as context to the simulation and our goal is to simulate the next 10s at 2Hz. Unless stated otherwise, every vehicle actor is *interactive* (*i.e.*, we simulate their behaviors).

AV2: Our second dataset, Argoverse 2 Motion Forecasting [17], consists of 250,000 11s urban traffic scenarios which are split into training, validation, test sets on an 80/10/10 basis. In this dataset, the first 5s of each scenario is given as context to the simulation and our goal is to re-simulate the next 6s at 2Hz. Each scenario is automatically annotated with trajectories whose interesting-ness and annotation quality is categorized as follows: (1) *focal* denotes the maximally interesting fully observed trajectory; (2) *scored* denotes all fully observed tracks within 30m of the focal trajectory; and (3) *unscored* denotes all remaining trajectories. Since annotation quality is low for *unscored*,

we only consider focal and scored vehicles as *interactive* (*i.e.*, other actors replay their annotated trajectories).

C.3. Metrics

Distributional realism: We calculate distributional realism metrics in a three-step process.

1. *Collect features:* We start by collecting scenarios features for both the reference real world scenarios and simulated scenarios. Our scenario features focus on agent kinematics (speed, acceleration, lateral acceleration), agent-to-agent interactions (lead distance, nearest distance). We compute each feature across K traffic scenarios, each with N_k *interactive* agents and T simulation time-steps (excluding the initial scenario context). Then, we aggregate these features into a flattened list of size $\sum_{k=1}^{K} N_k T$.

2. *Build histograms:* We build histograms $P(x)$ and $Q(x)$ with feature-specific histogram settings. See Tab. 5 for detailed settings. We specify bin size manually for each feature. We set min and max automatically based on the data and clip values beyond reasonable range.

3. *Calculate divergence:* Lastly, we compute the Jensen-Shannon divergence (**JSD**) [7] between histograms,

$$\mathrm{JSD}(P \mid\mid Q) = \frac{1}{2}\,\mathrm{KLD}(P \mid\mid M) + \frac{1}{2}\,\mathrm{KLD}(Q \mid\mid M) \tag{13}$$

where $\mathrm{KLD}(P \mid\mid Q) = \mathbb{E}_{x\sim P}[\log P(x) - \log Q(x)]$ is the KL-divergence and $M = \frac{1}{2}(P+Q)$.

Diversity: We evaluate diversity with two metrics. First, we consider *final displacement diversity* (**FDD**), which measures the maximum distance between the final trajectory waypoints among the sampled scenario variations. In particular, given initial context from a real scenario, we sample K simulations and calculate,

$$\mathrm{FDD}\left(\{\tilde{\mathbf{s}}^{(k)}\}_{k=1}^{K}\right) = \frac{1}{N}\sum_{i=1}^{N}\max_{k,k'}\left\|\tilde{\mathbf{s}}_{i,T}^{(k)} - \tilde{\mathbf{s}}_{i,T}^{(k')}\right\|^2 \tag{14}$$

where $\tilde{\mathbf{s}}_{i,T}^{(k)}$ is the 2D position of the i-th *interactive* agent at the final time-step T of the simulation in the k-th sample. Furthermore, we report the minimum scenario FDE (**minSFDE**) [13], where the final displacement error (FDE) is computed for only the best matching sample.

$$\mathrm{MinSFDE}(\mathbf{s}, \tilde{\mathbf{s}}) = \min_{k}\frac{1}{N}\sum_{i=1}^{N}\left\|\mathbf{s}_{i,T} - \tilde{\mathbf{s}}_{i,T}^{(k)}\right\|^2 \tag{15}$$

Intuitively, this metric measures a combination of realism and diversity, since the simulation model need to recover a scenario variation that matches the ground truth from a limited number of samples ($K = 6$ in our experiments).

Controllability: We evaluate MIXSIM's controllability via a cyclic consistency metric [18]. Specifically, we compute route consistency (**RC**), which measures the path-to-path distance between an agent's desired route and the route reconstructed from its simulation trajectory. This metric implements the intuition that the more controllable an agent is, the better we can explain its simulation trajectory by its desired route. Thus, for an idealized controllable agent, the difference between its desired route and one reconstructed from its simulation trajectory should be small.

Given a lane graph $G = (V, E)$, let $\mathbf{g}_i = u_0, \ldots, u_L$ denote the i-th agent's desired route in G. Let $\tilde{\mathbf{g}}_i = \tilde{u}_0, \ldots, \tilde{u}_{L'}$ denote a route reconstructed from its simulation trajectory using the HMM-based algorithm described earlier. To compute route consistency, we first extrapolate $\mathbf{g}_i$ and $\tilde{\mathbf{g}}_i$ so that the two routes have similar length, assuming the agent keeps its final lane. Then, we compute route consistency as the symmetrized distance between $\mathbf{g}_i$ and $\tilde{\mathbf{g}}_i$,

$$\mathrm{RC}(\mathbf{g}_i, \tilde{\mathbf{g}}_i) = \frac{1}{2}\left(\mathrm{d}_{\mathrm{route}}(\mathbf{g}_i, \tilde{\mathbf{g}}_i) + \mathrm{d}_{\mathrm{route}}(\tilde{\mathbf{g}}_i, \mathbf{g}_i)\right) \tag{16}$$

where $\mathrm{d}_{\mathrm{route}}(\mathbf{g}_i, \tilde{\mathbf{g}}_i)$ is the average point-to-line distance between the mid-point of each lane segment in $\mathbf{g}_i$ to its nearest lane segment in $\tilde{\mathbf{g}}_i$,

$$\mathrm{d}_{\mathrm{route}}(\mathbf{g}_i, \tilde{\mathbf{g}}_i) = \frac{1}{|\mathbf{g}_i|}\sum_{u_\ell \in \mathbf{g}_i}\min_{\tilde{u}_k \in \tilde{\mathbf{g}}_i}\min_{\mathbf{p}\in\tilde{u}_k}\|\mathrm{mid}(u_\ell) - \mathbf{p}\|_2 \tag{17}$$

and likewise for $\mathrm{d}_{\mathrm{route}}(\tilde{\mathbf{g}}_i, \mathbf{g}_i)$. We report route consistency averaged over agents and scenarios.

C.4. Finding Safety Critical Variations

Dataset: To increase the experiment scale beyond the original evaluation set of 200 HIGHWAY scenarios, the adversarial experiment is conducted with an independently-generated set of 385 simulated highway scenarios. We refer to this dataset as HIGHWAY-ADVERSARIAL.

Actor selection: Given an 8 second log containing ground truth actor trajectories, we use the first 3 seconds as context. Over the remaining 5 seconds, we control the chosen adversarial actor while the remaining non-SDV vehicles are controlled by MIXSIM. A successful adversarial actor must be close enough to influence the SDV within the short attack window. We heuristically select candidate actors that: (1) are travelling in the same direction as the SDV;

and (2) are able to catch up to the SDV within the 5 second unroll window by accelerating or decelerating at $1.0\mathrm{m\,s^{-2}}$. In HIGHWAY-ADVERSARIAL, this produces a total of 400 candidates, ranging from 0 to 6 per scenario.

Baselines: The BICYCLE and BICYCLE-F baselines use the same kinematic bicycle model parameterization as [15]. The parameters do not provide an explicit trajectory but rather a perturbation that is added to the original trajectory of the actor. We review the details here.

- *Acceleration*: both the perturbation and the final trajectory are constrained to $\pm 2.0\mathrm{m\,s^{-2}}$.
- *Curvature Rate*: both the perturbation and the final rate of change of curvature are constrained to $\pm 0.05\mathrm{m^{-1}\,s^{-1}}$.

Different from [15], we do not reject trajectories that collide with the SDV's ground truth trajectory because we are not investigating solvability.

In BICYCLE-F, we implement the feasible set constraint by pre-sampling 20,000 trajectories from a uniform distribution and rejecting those which go off-road or collide with non-SDV agents. If the feasible set is smaller than 100 trajectories, we resample up to 9 more times. Over 99% of sampled trajectories are rejected, demonstrating how rare it is to find even mildly realistic actor behaviour. After this process, 82% of feasible sets contain no more than 200 trajectories.

The adversarial actor is controlled for 5 seconds at 2Hz, for a total of 10 control points resulting in a 20-dimensional search space.

MIXSIM: For our experiment, we choose to map the discrete space of lane graph routes to a continuous space that is comparable to the one used by the BICYCLE and BICYCLE-F baselines. There are many ways to do this; we use a parameterization that is analogous to steering. Starting from a lane segment u in the lane graph $G = (V, E)$, we consider all lane segments $v \in V$ where: (1) v is a successor, left, or right neighbour of u; and (2) the relative heading from u to v is within $\frac{\pi}{3}\mathrm{rad}$ of the direction of u. We then break $[0, 1]$ into equally-sized sub-intervals and associate each sub-interval with one of the lane segments, ordered by the relative heading from u to that segment. Given a *decision parameter* $d \in [0, 1]$, we select the next lane segment corresponding to which sub-interval d is in. Then, given a starting node u_0 and a sequence of L decision parameters $d_1, \ldots, d_L$, we construct a lane graph route $u_0, \ldots, u_L$ by using d_i to decide u_i given u_{i-1}. This route is finally presented to the route-conditional policy.

In HIGHWAY-ADVERSARIAL, the adversarial actor is assumed to have a maximum speed of $30\mathrm{m\,s^{-1}}$. To control the adversary for the next 5 seconds, the constructed route must have a length of at least 150 meters. Using a lane graph with 10 meter lane segments, a decision parameter sequence of length 15 is sufficient, resulting in a 15-dimensional search space.

Parameter	BICYCLE & BICYCLE-F	MIXSIM
variance	564.03	620.71
lengthscale	0.1492	0.2124

Table 6. Fitted Matern kernel parameters for Bayesian Optimization.

Bayesian optimization: We use our own implementation of Bayesian Optimization built with [6] and [11]. To standardize the search space, the acceleration and curvature rate parameters used by BICYCLE and BICYCLE-F are normalized to $[0, 1]$. MIXSIM's decision sequence parameterization is already normalized. For all experiments, we use a Matern kernel with $\nu = \frac{5}{2}$ and the expected improvement acquisition function with $\epsilon = 0.01$. To find the optimal kernel parameters, we perform the same optimization experiment on HIGHWAY using default kernel parameters. The data from those experiments is used to fit separate kernels for the two parameterizations; see Tab. 6 for the fitted values. Our query budget of 75 matches that of [15]. We find this sufficient, as the vast majority of successful attacks are found within 25 queries.

References

[1] Dimitri Bertsekas. *Dynamic Programming and Optimal Control*. 2000. 5

[2] Yuning Chai, Benjamin Sapp, Mayank Bansal, and Dragomir Anguelov. MultiPath: Multiple probabilistic anchor trajectory hypotheses for behavior prediction. In *CoRL*, 2019. 1, 2

[3] Alexander Cui, Sergio Casas, Kelvin Wong, Simon Suo, and Raquel Urtasun. GoRela: Go relative for viewpoint-invariant motion forecasting. In *ICRA*, 2023. 1, 2, 3

[4] Henggang Cui, Vladan Radosavljevic, Fang-Chieh Chou, Tsung-Han Lin, Thi Nguyen, Tzu-Kuo Huang, Jeff Schneider, and Nemanja Djuric. Multimodal trajectory predictions for autonomous driving using deep convolutional networks. In *ICRA*, 2019. 1, 2

[5] Yarin Gal and Zoubin Ghahramani. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *ICML*, 2016. 5

[6] GPy. GPy: A gaussian process framework in python. http://github.com/SheffieldML/GPy, since 2012. 7

[7] Maximilian Igl, Daewoo Kim, Alex Kuefler, Paul Mougin, Punit Shah, Kyriacos Shiarlis, Dragomir Anguelov, Mark Palatucci, Brandyn White, and Shimon Whiteson. Symphony: Learning realistic and diverse agents for autonomous driving simulation. In *ICRA*, 2022. 6

[8] Arne Kesting. MOBIL: General lane-changing model for car-following models. 2007. 5

[9] Karsten Kreutz and Julian Eggert. Analysis of the generalized intelligent driver model (GIDM) for uncontrolled intersections. In *ITSC*, 2021. 5

[10] Ming Liang, Bin Yang, Rui Hu, Yun Chen, Renjie Liao, Song Feng, and Raquel Urtasun. Learning lane graph representations for motion forecasting. In *ECCV*, 2020. 1, 2, 3

[11] Richard Liaw, Eric Liang, Robert Nishihara, Philipp Moritz, Joseph E. Gonzalez, and Ion Stoica. Tune: A research platform for distributed model selection and training. *CoRR*, 2018. 7

[12] Paul Newson and John Krumm. Hidden markov map matching through noise and sparseness. In *SIGSPATIAL*, 2009. 4

[13] Simon Suo, Sebastian Regalado, Sergio Casas, and Raquel Urtasun. TrafficSim: Learning to simulate realistic multi-agent behaviors. In *CVPR*, 2021. 6

[14] Martin Treiber, Ansgar Hennecke, and Dirk Helbing. Congested traffic states in empirical observations and microscopic simulations. *Physical Review E*, 2000. 5

[15] Jingkang Wang, Ava Pun, James Tu, Sivabalan Manivasagam, Abbas Sadat, Sergio Casas, Mengye Ren, and Raquel Urtasun. AdvSim: Generating safety-critical scenarios for self-driving vehicles. In *CVPR*, 2021. 7

[16] Moritz Werling, Julius Ziegler, Sören Kammel, and Sebastian Thrun. Optimal trajectory generation for dynamic street scenarios in a Frenét frame. In *ICRA*, 2010. 5

[17] Benjamin Wilson, William Qi, Tanmay Agarwal, John Lambert, Jagjeet Singh, Siddhesh Khandelwal, Bowen Pan, Ratnesh Kumar, Andrew Hartnett, Jhony Kaesemodel Pontes, Deva Ramanan, Peter Carr, and James Hays. Argoverse 2: Next generation datasets for self-driving perception and forecasting. In *NeurIPS Datasets and Benchmarks*, 2021. 5

[18] Eric Zhan, Albert Tseng, Yisong Yue, Adith Swaminathan, and Matthew J. Hausknecht. Learning calibratable policies using programmatic style-consistency. In *ICML*, 2020. 6

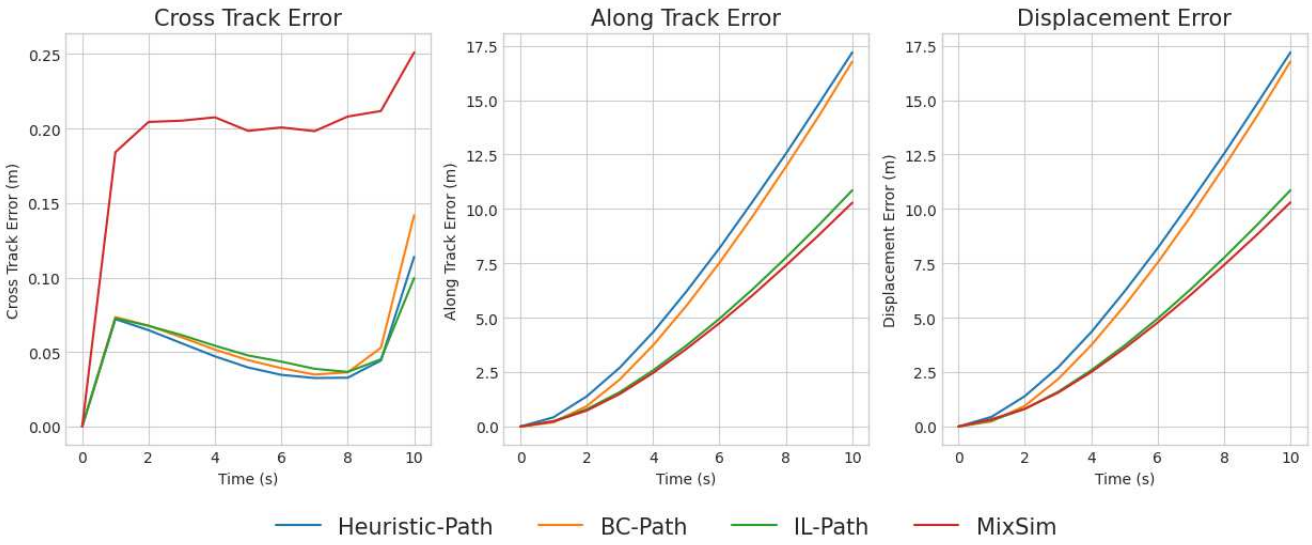

Figure 2. Comparison against path-following baselines in reconstruction realism metrics on HIGHWAY.

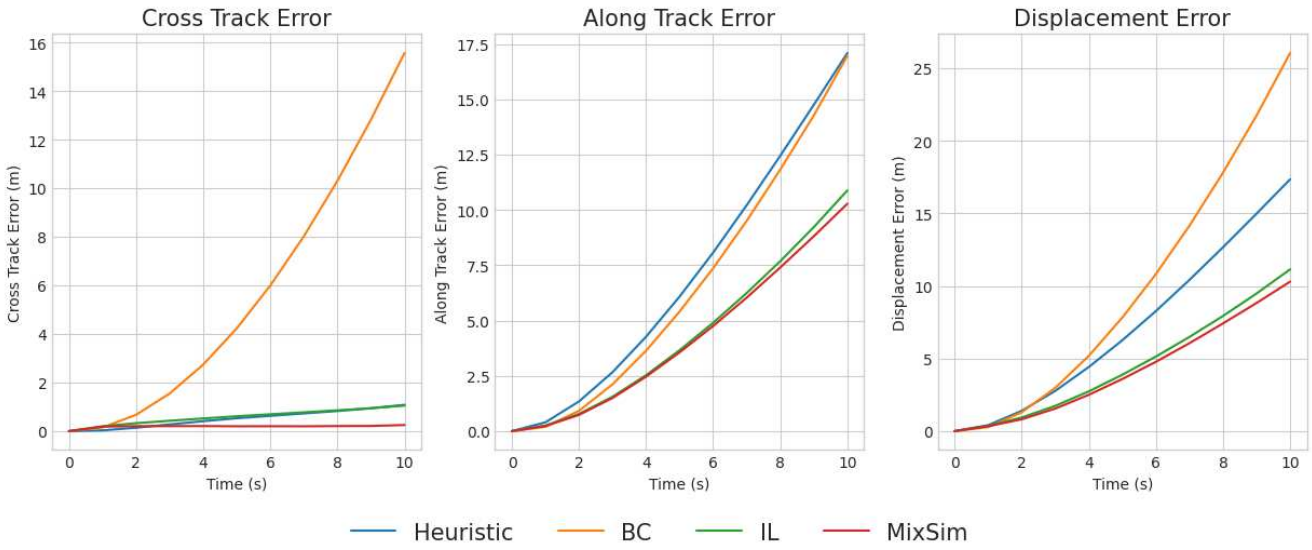

Figure 3. Comparison against non path-following baselines in reconstruction realism metrics on HIGHWAY.

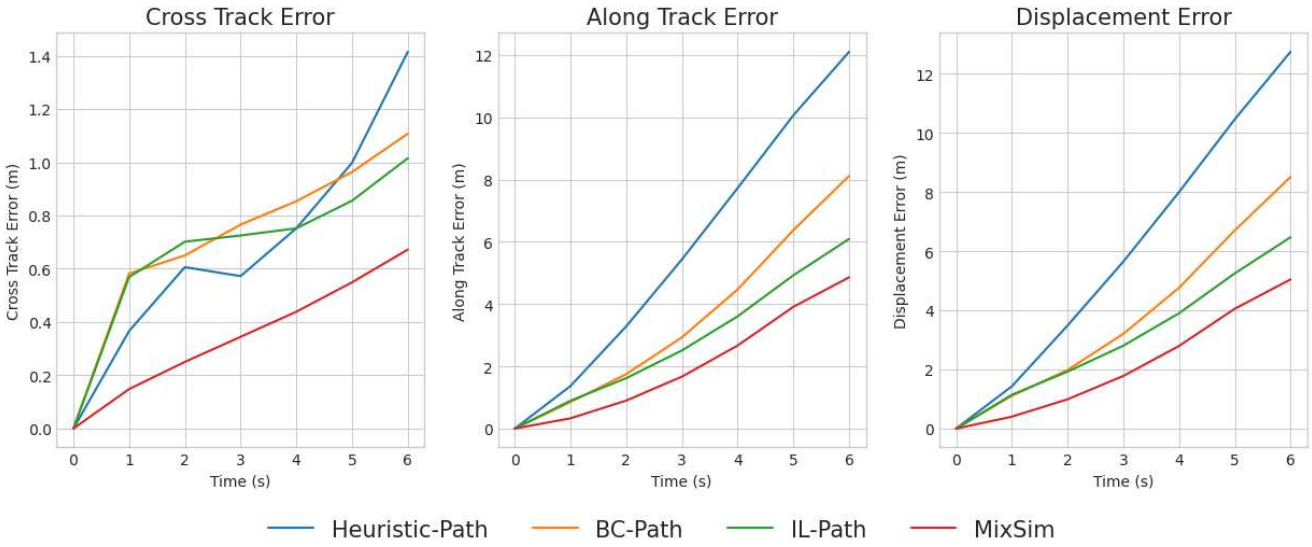

Figure 4. Comparison against path-following baselines in reconstruction realism metrics on AV2.

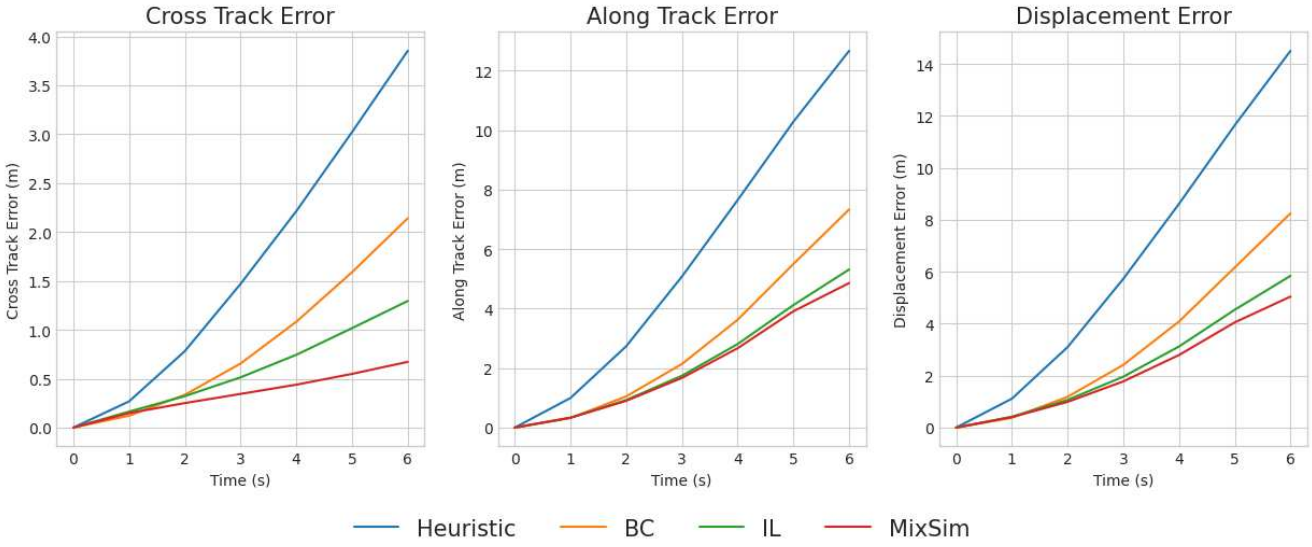

Figure 5. Comparison against non path-following baselines in reconstruction realism metrics on AV2.

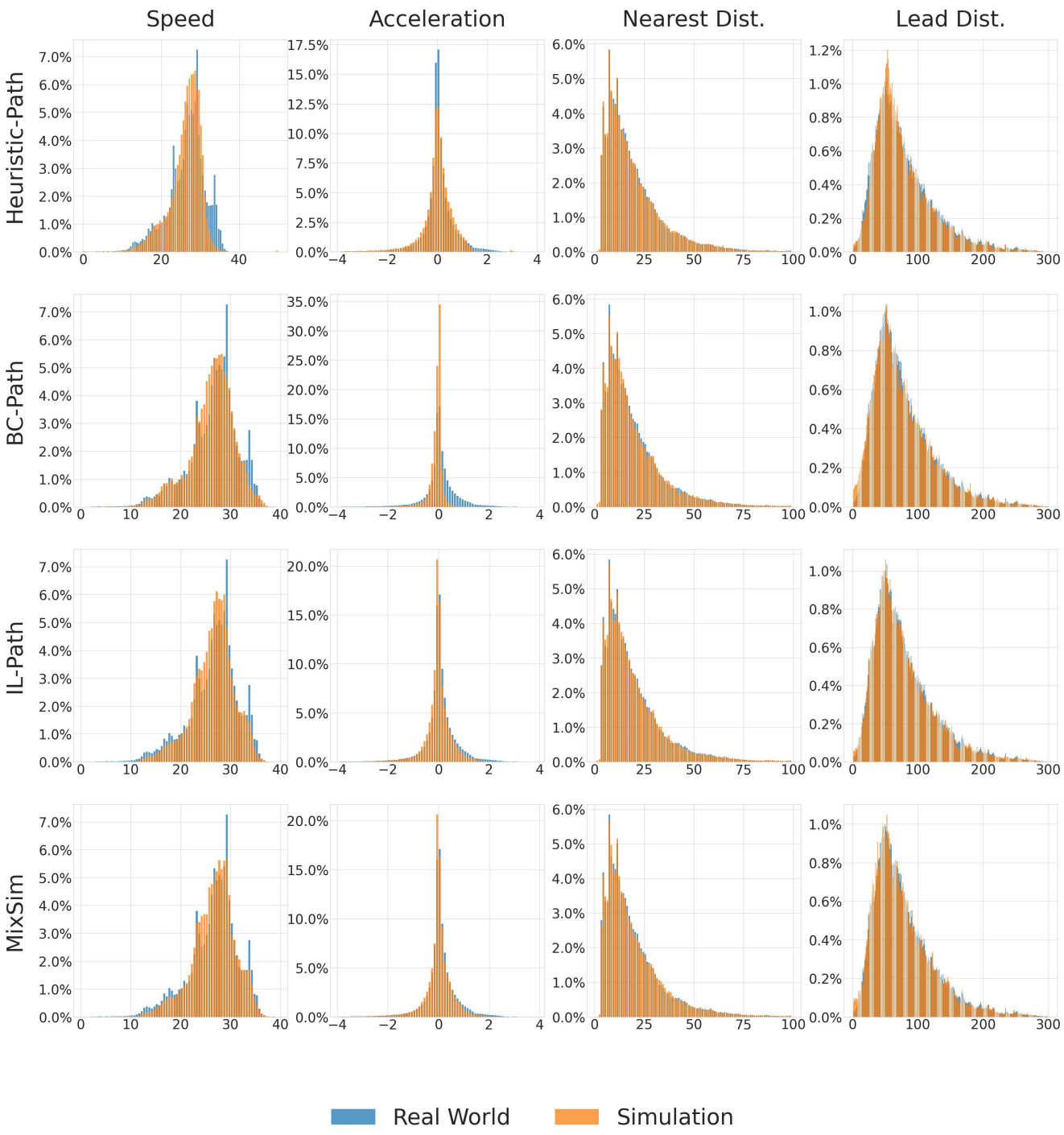

Figure 6. Comparison against path-following baselines in distributional realism metrics on HIGHWAY.

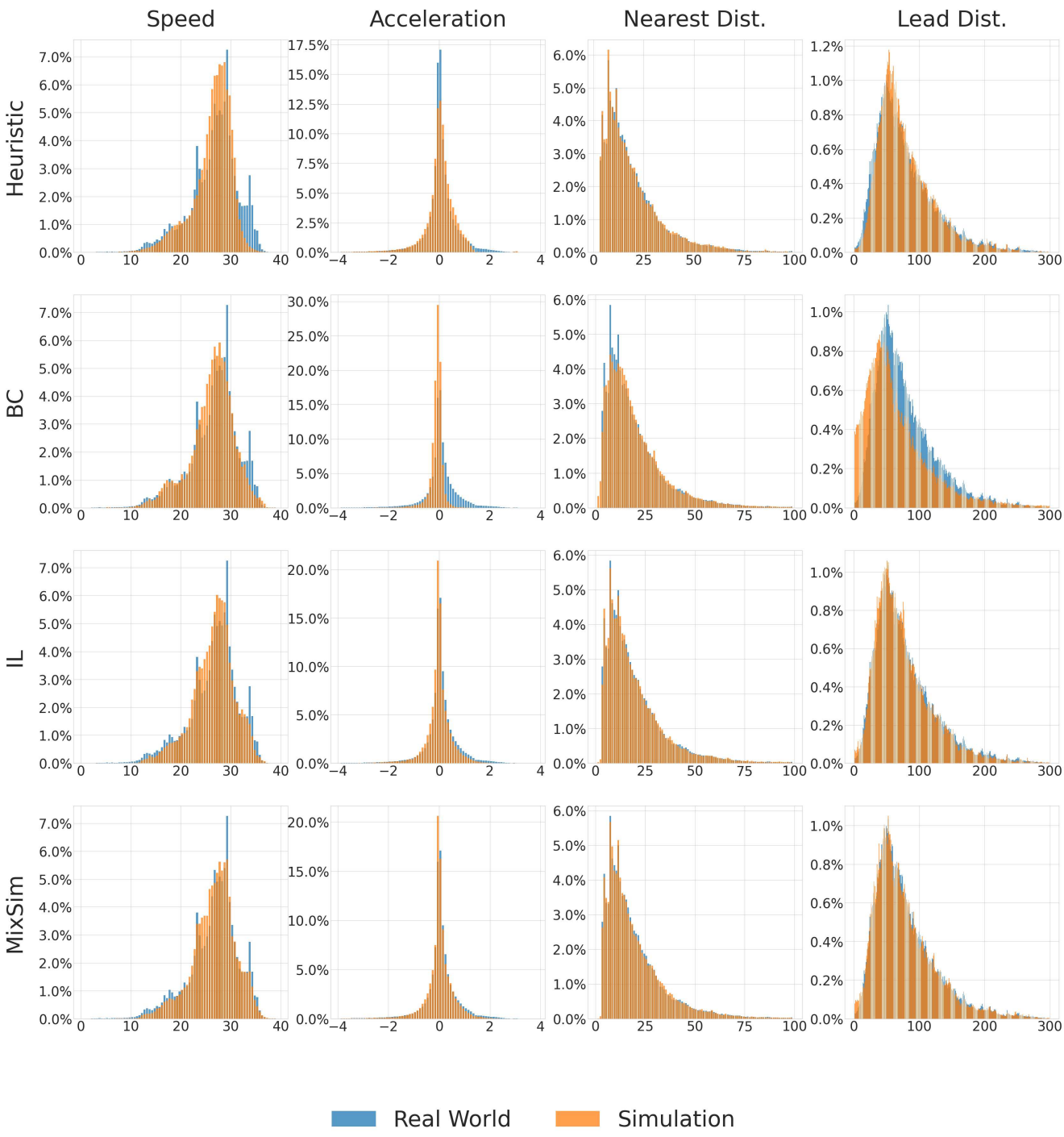

Figure 7. Comparison against non path-following baselines in distributional realism metrics on HIGHWAY.

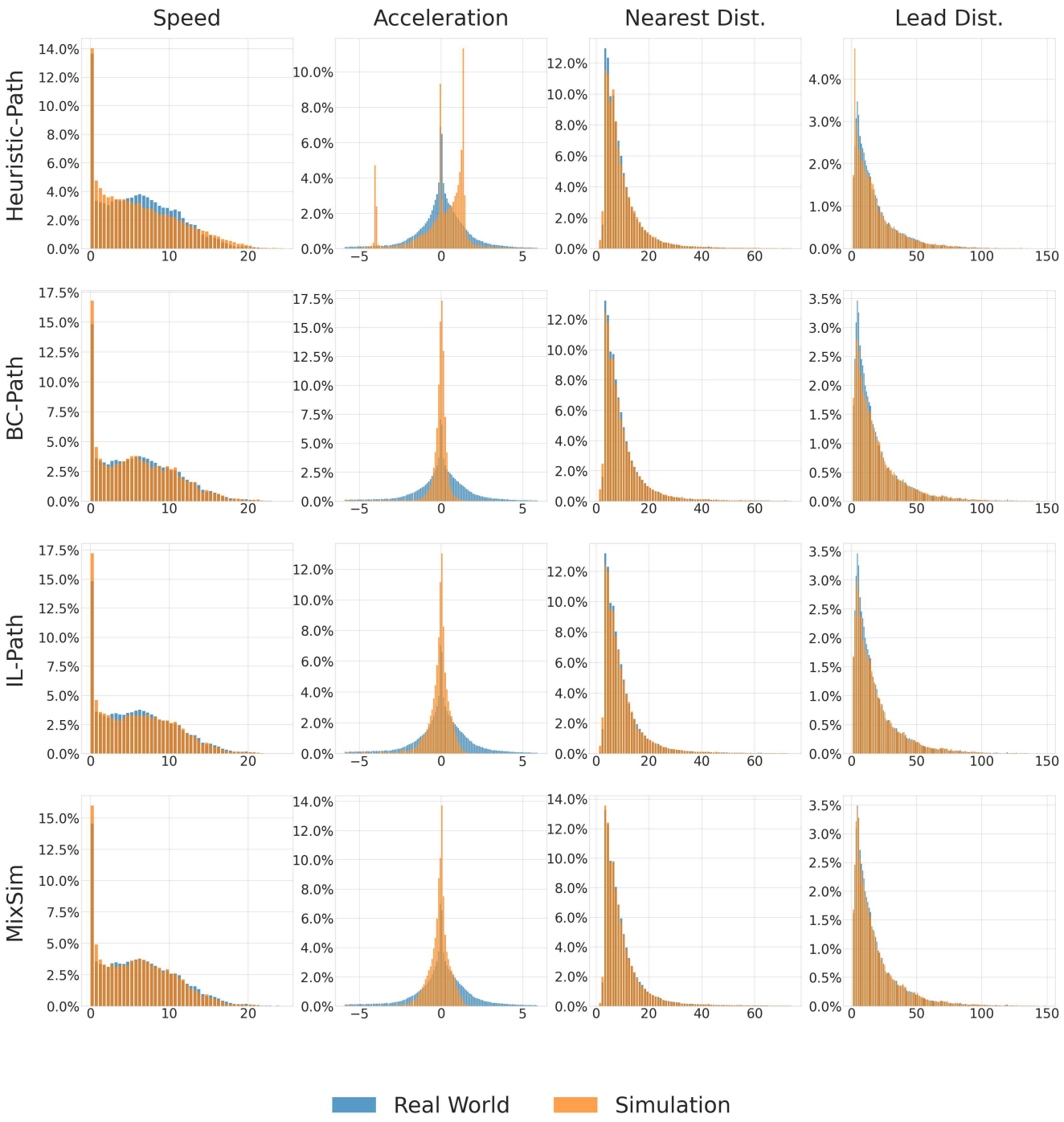

Figure 8. Comparison against path-following baselines in distributional realism metrics on AV2.

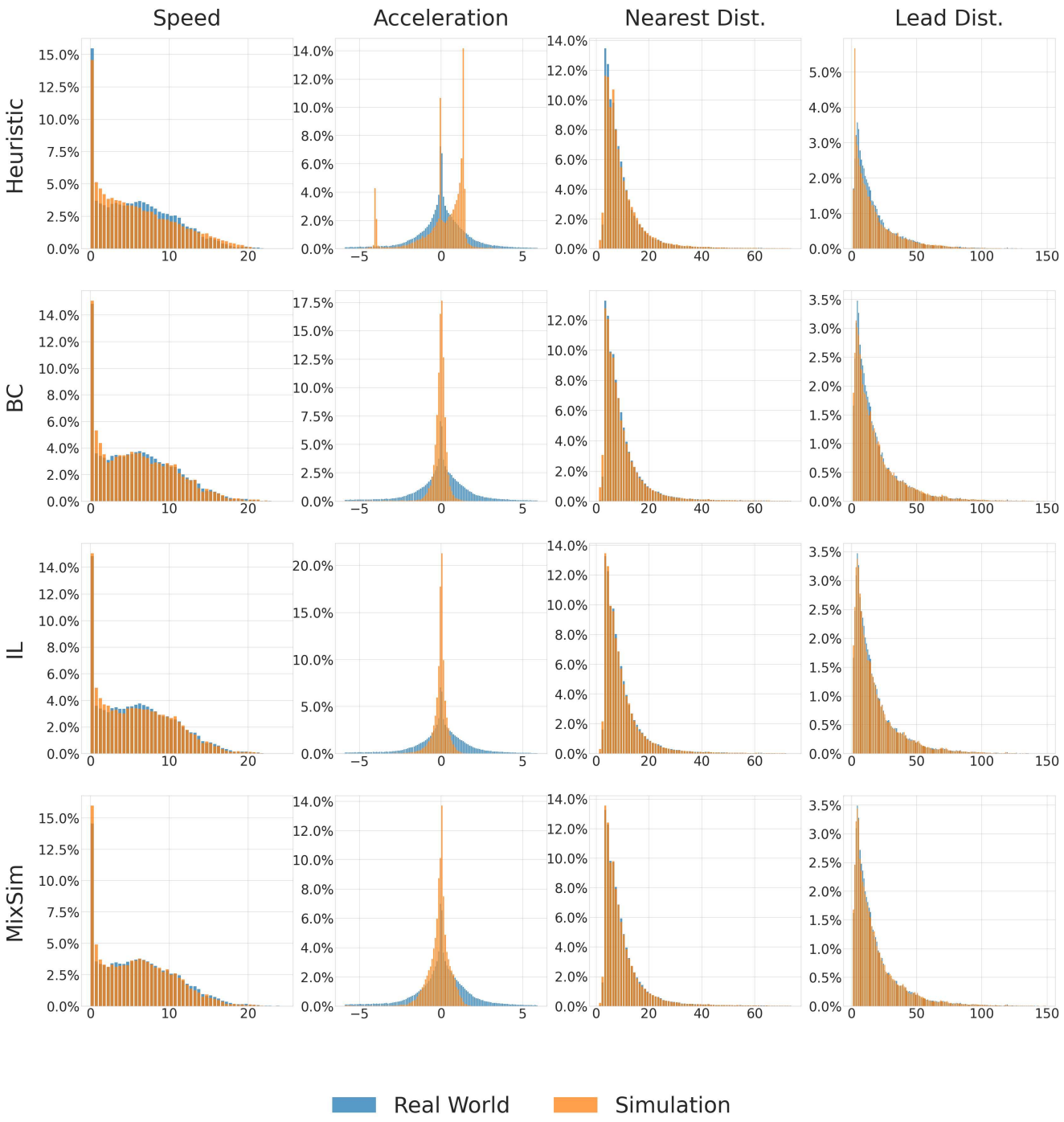

Figure 9. Comparison against non path-following baselines in distributional realism metrics on AV2.