

Supplementary Material – TrafficSim: Learning to Simulate Realistic Multi-Agent Behaviors

Simon Suo^{1,2}, Sebastian Regalado³, Sergio Casas^{1,2}, Raquel Urtasun^{1,2}

¹Uber ATG, ²University of Toronto, ³University of Waterloo

{suo, sergio, urtasun}@cs.toronto.edu, sdregala@edu.uwaterloo.ca

In this supplementary material, we provide the following: additional details of our method in Section A, implementation details of baselines and metrics in Section B, and lastly, additional qualitative results in Section C.

A. Additional TRAFFICSIM Details

In this section, we describe additional details of input parameterization, model architecture, and learning methodology for TRAFFICSIM.

Input Parameterization: We use a rasterized map representation that encodes traffic elements into different channels of a raster. There are a total of 13 map channels consisting of intersections, lanes, roads, etc. We encode traffic control as additional channels, by rasterizing the lane segments controlled by the traffic light. We initialize each scenario with 3s of past actor states, with each actor history represented by 7 bounding boxes across time, each 0.5s apart. When an actor does not have the full history, we fill the missing states with NaNs. For simplicity, we keep the traffic light state fixed (at $t=0$) across the simulation horizon in training. This allows us to process the global map feature once, and accumulate gradients on the same features through back-propagation.

Global Map Module: We use a multi-scale backbone to extract map features at different resolution levels to encode both near and long-range map topology. The architecture is adapted from [12]: it consists of a sequence of 4 blocks, each with a single convolutional layer of kernel size 3 and [8, 16, 32, 64] channels. After each block, the feature maps are down-sampled using max pooling with stride 2. Finally, feature maps from each block are resized (via average-pooling or bilinear sampling) to a common resolution of 0.8m, concatenated, and processed by a header block with 2 additional convolutional layers with 64 channels.

Local Observation Module: We design the local observation modules to be lightweight and differentiable. This

enables the simulation to be fast and allows us to backpropagate gradient through the simulation. Works in motion forecasting (e.g., [8]) typically rasterize the bounding boxes of each actor, use a convolutional network to extract motion features, and rely on a limited receptive field to incorporate influences from its neighbors. In contrast, we directly encode the numerical values parameterizing the bounding boxes using a 4-layer GRU with 128 hidden states, and rely on the graph neural network based module for interaction reasoning. More concretely, we fill NaNs with zeros, and also pass in binary mask indicating missing values to the GRU. For extracting local map features from the pre-processed map features, we use Rotated Region of Interest Align with 70m in front, 10m behind, and 20m on each side. The extracted local features are further processed by a 3-layer CNN, and then max-pooled across the spatial dimensions. The final local context x_i for each actor i is a 192 dimensional vector formed by concatenating the map and motion features.

Scene Interaction Module: We leverage a graph neural network based scene interaction module to parameterize our joint actor policy. In particular, our scene interaction module is inspired by [6, 7], and is used in our Prior, Posterior, and Decoder networks. We provide an algorithmic description in Algorithm 1. The algorithm is written with for loops for clarity, but in practice our implementation is fully vectorized, since the only loop that is needed is that of the K rounds of message passing, but in practice we observe that $K = 1$ is sufficient. Our edge function $\mathcal{E}^{(k)}$ consists of a 3-layer MLP that takes as input the hidden states of the 2 terminal nodes at each edge in the graph at the previous propagation step as well as the projected coordinates of their corresponding bounding boxes. We use feature-wise max-pooling as our aggregate function $\mathcal{A}^{(k)}$. To update the hidden states we use a GRU cell as $\mathcal{U}^{(k)}$. Finally, to output the results from the graph propagations, we use another MLP as readout function \mathcal{O} .

Algorithm 1 SIM: Scene Interaction Module

Input: Initial hidden state for all of the actors in the scene $H^0 = \{h_0^0, h_1^0, \dots, h_N^0\}$. BEV coordinates of the actor bounding boxes $\{c_0, c_1, \dots, c_N\}$. Number of message propagations K (defaults to $K = 1$).

Output: Output vector per node $\{o_0, o_1, \dots, o_N\}$.

- 1: Construct actor interaction graph $G = (V, E)$
 - 2: Compute pairwise coordinate transformations $\mathcal{T}(c_u, c_v), \forall (u, v) \in E$
 - 3: **for** $k = 1, \dots, K$ **do** ▷ Loop over graph propagations
 - 4: **for** $(u, v) \in E$ **do** ▷ Compute message for every edge in the graph
 - 5: $m_{u \rightarrow v}^{(k)} = \mathcal{E}^{(k)}(h_u^{k-1}, h_v^{k-1}, \mathcal{T}(c_u, c_v))$
 - 6: **for** $v \in V$ **do** ▷ Update node states
 - 7: $a_v^{(k)} = \mathcal{A}^{(k)}(\{m_{u \rightarrow v}^{(k)} : u \in \mathbf{N}(v)\})$ ▷ Aggregate messages from neighbors
 - 8: $h_v^{(k)} = \mathcal{U}^{(k)}(h_v^{(k-1)}, a_v^{(k)})$ ▷ Update the hidden state
 - 9: **for** $v \in V$ **do**
 - 10: $o_v = \mathcal{O}(h_v^{(K)})$ ▷ Compute outputs
 - 11: **return** $\{o_0, o_1, \dots, o_N\}$
-

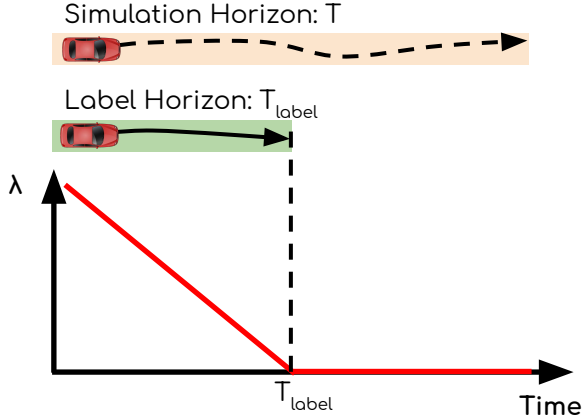


Figure 1: Our adaptive weight is a decreasing function of simulation timestep.

Simulating Traffic Scenarios: We provide an algorithmic description of the overall inference process of simulating traffic scenarios in Algorithm 2. While the algorithm describes the process of sampling a single scenario, and loop over actors in the scene, we can sample multiple scenarios with arbitrary number of actors in parallel by batching over samples and actors. Note that we do not directly regress heading of the actors. Instead, we approximate headings in a post-processing step by taking the tangent of segments between predicted waypoints. This ensures the headings are consistent with the predicted motion.

Time-Adaptive Multi-Task Loss: We use a multi-task loss to balance supervision from imitation and common

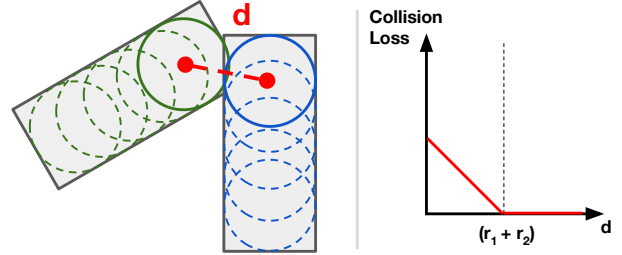


Figure 2: Differentiable relaxation of collision loss approximates each vehicle as 5 circles and considers distance between closest centroids.

sense:

$$\mathcal{L} = \sum_t \lambda(t) \mathcal{L}_{\text{imitation}}^t + (1 - \lambda(t)) \mathcal{L}_{\text{collision}}^t \quad (1)$$

Concretely, we define the time-adaptive weight as:

$$\lambda(t) = \min\left(\frac{T_{\text{label}} - t}{T_{\text{label}}}, 0\right) \quad (2)$$

where T_{label} is the label horizon (i.e., latest timestep which we have labels). Figure 1 illustrates this function. Intuitively, the weight on imitation must drop to zero at T_{label} as we no longer have access to labels. We note that our method is not sensitive to the choice of $\lambda(t)$. Experiments with other decreasing function of simulation timestep yields similar results.

Differentiable Relaxation of Collision: Figure 2 illustrates our proposed differentiable relaxation of collision.

More concretely, the loss is defined as follows:

$$\mathcal{L}_{\text{collision}}^t(\mathcal{Y}_{\text{prior}}^t) = \frac{1}{N^2} \sum_{i \neq j} \max(1, \sum_{\tau=t+1}^{t+P} \mathcal{L}_{\text{pair}}(y_i^\tau, y_j^\tau)) \quad (3)$$

$$\mathcal{L}_{\text{pair}}(y_i^\tau, y_j^\tau) = \begin{cases} 1 - \frac{d}{r_i + r_j}, & \text{if } d \leq r_i + r_j \\ 0, & \text{otherwise} \end{cases} \quad (4)$$

Intuitively, if there’s no overlap between any circles, the collision loss is 0. If two circles completely overlap, the collision loss is 1. We further reweight the collision loss with a factor of 0.01 to be on similar scale as the imitation loss.

B. Additional Experiment Details

In this section, we provide additional details on baselines, metrics, and experimental setup.

B.1. Baselines

IDM [11]: The Intelligent Driver Model (IDM) is a heuristic car following model that implement reactive keep lane behavior by following a specific headway vehicle. We resolve headway vehicle based on lane association, and also a narrow field of view in a 30° sector in front of each actor, with visibility up to 10 meters. We implement traffic control as a phantom actor that has zero size. We use a simulation frequency of 2.5Hz (0.4s per timestep), max deceleration of 3 m/s², reaction time of 0.1s, time headway of 1.5s. To generate diverse simulations, we sample max acceleration in the range of [0.6, 2.5] m/s², and desired speed in the range of [10, 20] m/s. Since the motion of IDM actors are parameterized against lane centerlines, it can trivially avoid traffic rule violations. However, it does suffer from an inherent limitation: unlike learned models, it cannot infer traffic flow from the initial actors states when given partially observed traffic light states, and thus results in occasional collisions at intersections.

MTP [8]: Multiple Trajectory Prediction (MTP) models uncertainty over actors’ future trajectories with mixture of Gaussians at each prediction timestep. It does not explicitly reason about interaction between actors as the future unrolls, and makes conditional independence assumption across actors. We use a mixture of Gaussian with 16 modes. Following the training methodology described in the original paper, we select the closest matching mode to compute loss, instead of directly optimizing the mixture density.

ESP [10]: ESP models multi-agent interaction by leveraging an autoregressive formulation, where actors influence each other as the future unrolls. Due to memory constraints, we limit the radii of the whiskers to [1, 2, 4] m while keeping the seven angle bins. We implement the social context

condition with a minor modification. The original paper specifies a fixed number of actors (since Carla has a small number of actors)., but this is not possible in ATG4D since traffic scenes contain many more actors. Thus, we use k-nearest neighbors to select $M = 4$ neighbors to gather social features.

ILVM [7]: We adapt ILVM from the joint perception and prediction setting by replacing voxelized LiDAR input by rasterized actor bounding boxes. Since processing noise-free actor bounding boxes require less model capacity than performing LiDAR perception, we reduce the number of convolutional layers in the backbone to improve inference speed. We noticed no degradation in performance in reducing the model capacity.

DataAug: We follow data augmentation technique described in [1], since it also leverages large-scale self-driving datasets and is closest to our setting. To factor out effects of model architecture, we use the best motion forecasting model ILVM as the base architecture. In particular, for each eligible trajectory snippet, we perturb the waypoint at the current timestep with a probability of 50%. we consider a trajectory to be eligible if has moved more than 16m in the 2s window around the perturbation (i.e. speed higher than 8m/s). We uniformly sample perturbation distance in the range of [0, 0.5] m, and sample a random direction to perturb the waypoint. Finally, we fit quadratic curves for both x and y as function of time, to smooth out the past and future trajectory. We only alter waypoints up to 2 seconds before and after the perturbation point.

AdversarialIL: Inspired by [9, 5, 4, 2, 3], we implement an adversarial imitation learning baseline by jointly learning a discriminator as the supervision for the policy. Similarly, to factor out the effects of model architecture, we leverage our differentiable observation modules and scene interaction module to parameterize the policy. In particular, the extracted scene context is directly fed to the scene interaction module to output a bivariate Gaussian of the next waypoint for each actor in the scene. The discriminator has a similar architecture with spectral normalization. By leveraging our differentiable components, we enable our adversarial IL baseline to also leverage back-propagation through simulation. This allows it to sidestep the challenge of policy optimization, and enables a more direct comparison with our method. For optimization, we use a separate Adam optimizer for the policy, and RMSProp optimizer for the discriminator. Furthermore, we found it’s necessary to periodically use behavior cloning loss to stabilize training. We use a replay buffer size of 200 and batch size 8 for optimizing the policy and the discriminator. Furthermore, following [2], we use

Algorithm 2 TRAFFICSIM: Simulating Traffic Scenarios

Input: Rasterized high definition map M . Initial actor states $Y^{-H:0} = \{Y^{-H}, \dots, Y^0\}$ where each $Y^t = \{y_1^t, y_2^t, \dots, y_N^t\}$ for the N actors in the scene.

Output: Simulated actor states $Y^{1:T} = \{Y^1, Y^2, \dots, Y^T\}$ for T simulation timesteps.

```
1:  $\tilde{M} \leftarrow \text{MapBackbone}(M)$  ▷ Extract global map feature once per environment
2: for  $t = 1, \dots, T$  do ▷ Simulate for requested number of timesteps
3:   for  $i = 1, \dots, N$  do ▷ Extract local context for each actor at each timestep
4:      $r_i \leftarrow \text{RRoiAlign}(y_i^t, \tilde{M})$ 
5:      $x_i^{\text{map}} \leftarrow \text{MaxPooling}(\text{CNN}(r_i))$ 
6:      $x_i^{\text{motion}} \leftarrow \text{GRU}(y_i^{-H:t})$ 
7:      $x_i \leftarrow x_i^{\text{map}} \oplus x_i^{\text{motion}}$ 
8:    $X = \{x_i : \forall i \in 1 \dots N\}$ 
9:    $\{Z_\mu, Z_\sigma\} \leftarrow \text{Prior}_\gamma(X)$  ▷ Use SIM modules to output latent prior distribution
10:   $Z \sim \mathcal{N}(\{Z_\mu, Z_\sigma \cdot I\})$  ▷ Sample a scene latent from diagonal Gaussian
11:   $H = \{\text{MLP}(x_i \oplus z_i) : \forall i \in 1 \dots N\}$ 
12:   $\mathcal{Y} \leftarrow \text{Decoder}_\theta(H)$  ▷ Use SIM module to decode actor plans
13:   $Y^{t+1:t+\kappa} \leftarrow \mathcal{Y}$  ▷ Update environment by taking the first  $\kappa$  steps of the actor plans
14: return  $Y^{1:T} = \{Y^1, Y^2, \dots, Y^T\}$ 
```

a curriculum of increasing simulation horizon to ease optimization. More concretely, we first pre-train with behavior cloning for 25k steps, then follow a schedule of [2,4,6,8] simulation timesteps increasing every 25k steps. We found increasing simulation horizon further does not improve the model.

B.2. Metrics

Scenario Reconstruction: For each simulation environment (i.e., with a given map, traffic control, and initial actor states), we define the following scenario reconstruction metrics over the K traffic scenarios sampled from the model:

$$\begin{aligned} \text{minSADE} &= \min_{k \in 1 \dots K} \frac{1}{NT_{\text{label}}} \sum_{n=1}^N \sum_{t=1}^{T_{\text{label}}} \|y_{n,GT}^t - y_{n,(k)}^t\|^2 \\ \text{minSFDE} &= \min_{k \in 1 \dots K} \frac{1}{N} \sum_{n=1}^N \|y_{n,GT}^{T_{\text{label}}} - y_{n,(k)}^{T_{\text{label}}}\|^2 \\ \text{meanSADE} &= \frac{1}{KN T_{\text{label}}} \sum_{k=1}^K \sum_{n=1}^N \sum_{t=1}^{T_{\text{label}}} \|y_{n,GT}^t - y_{n,(k)}^t\|^2 \\ \text{meanSFDE} &= \frac{1}{KN} \sum_{k=1}^K \sum_{n=1}^N \|y_{n,GT}^{T_{\text{label}}} - y_{n,(k)}^{T_{\text{label}}}\|^2 \end{aligned}$$

In particular, we only evaluate up to $T_{\text{label}} = 8s$ due to the availability of ground truth actor trajectories.

Interaction Reasoning: Our scenario level collision rate metric is implemented as follows:

$$\text{SCR} = \frac{1}{NS} \sum_{s=1}^S \sum_{i=1}^N \min \left(1, \sum_{j>i}^N \sum_{t=1}^T \mathbb{1}[IoU(b_{i,s}^t, b_{j,s}^t) > \varepsilon] \right)$$

In particular, we consider two actors to be in collision if their bounding boxes overlap each other with IOU greater than a small ϵ of 0.1. This threshold is necessary since the labeled bounding boxes are slightly larger than true vehicle shape, thus sometimes resulting in collisions even in ground truth scenarios. Furthermore, we count a maximum of 1 collision per actor. In other words, we count number of actors in collision, rather than number of total collisions between pairs of actors.

Traffic Rule Compliance: We leverage our high definition map with precise lane graph annotations to evaluate traffic rule compliance. More concretely, we first obtain the drivable areas of each actor in the scenario by first performing lane association. Then, we traverse the lane graph to derive a set of reachable lane segments from the initial location, including neighbours and successors. Furthermore, we cut off any connection influenced by traffic control (i.e. red traffic light). We rasterize the drivable area with binary values: 1 for drivable and 0 for violation. This allows us to efficiently index and calculate traffic rule violations. To handle actors that begin initially outside of mapped region (i.e. parked vehicles on the side of the road or in a parking lot), we ignore actors that do not have initial lane associations.

Diversity: To calculate our map-aware diversity metric, we leverage the same drivable area raster employed by the traffic rule compliance metric. More concretely, we first filter out actor trajectory samples that violate traffic rule, then we measure the average distance (across time) between

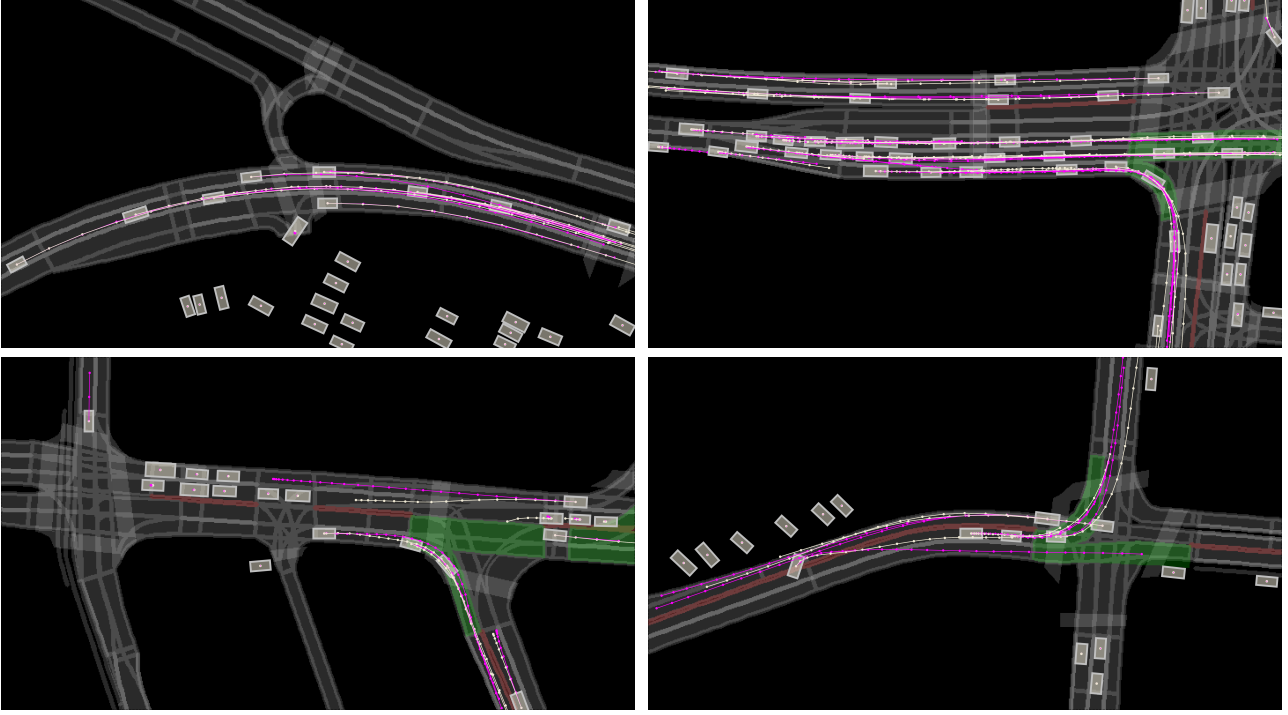


Figure 3: Comparison between simulated trajectories (purple) sampled from TrafficSim and real trajectories (white) from ATG4D

the two most distinct trajectory samples for each actor.

$$\text{MASD} = \max_{k, k' \in 1 \dots K} \frac{1}{NT} \sum_{n=1}^N \sum_{t=1}^T \|y_{n,(k)}^t - y_{n,(k')}^t\|^2$$

B.3. Experimental Setup

TRAFFICSIM for Data Augmentation: For synthetic data generation, we generate approximately 15k examples by initializing from a subset of training scenarios used to train the traffic simulation models. We use the same amount of real data for fair comparison. We use a simple imitation planner, which takes rasterized map and actor bounding box history as input, and directly regresses the future plan. The planning horizon is 5s, with 0.5s per waypoint, for a total of 10 waypoints. The imitation planner is learned with supervision from all actor trajectories in the synthetic scenarios. For evaluation, we test on the same set of ground truth scenarios that are used for evaluating traffic simulation metrics.

Incorporating Constraints at Simulation-Time: For rejection sampling at simulation-time, we resample at most 10 times to keep the runtime bounded. If we cannot generate enough collision-free plans after 10 re-sampling steps, we sort all the generated samples by collision loss, then return

the minimum cost plans. To reason about potential collision in the future, we evaluate our collision loss on the first 5 timesteps of the sampled actor plans (i.e. 2.5s into the future). For gradient-based optimization, we leverage our differentiable relaxation of collision loss. Similarly, we evaluate the collision loss on the first 5 timesteps of the actor plans (i.e. 2.5s into the future). While keeping our model frozen, we backpropagate the gradient to optimize the latent samples Z^t . Performing the optimization in the latent space allows us to influence the actor plans while remaining in the model distribution. More concretely, we take 5 gradient steps with a learning rate of 1e-2.

C. Additional Qualitative Results

In this section, we showcase additional qualitative results. Please refer to the supplementary video for animated sequences. Figure 3 shows examples from ATG4D and overlays the most likely sample from TRAFFICSIM (given the same map and initial states). Since ATG4D is observational, it only captures a single reality. Our simulation recovers that reality in most cases. And when it does not, it generates highly sensible alternatives. Figure 4 and 5 show additional traffic scenarios sampled from our model. Figure 6 shows comparison between baselines and our model.

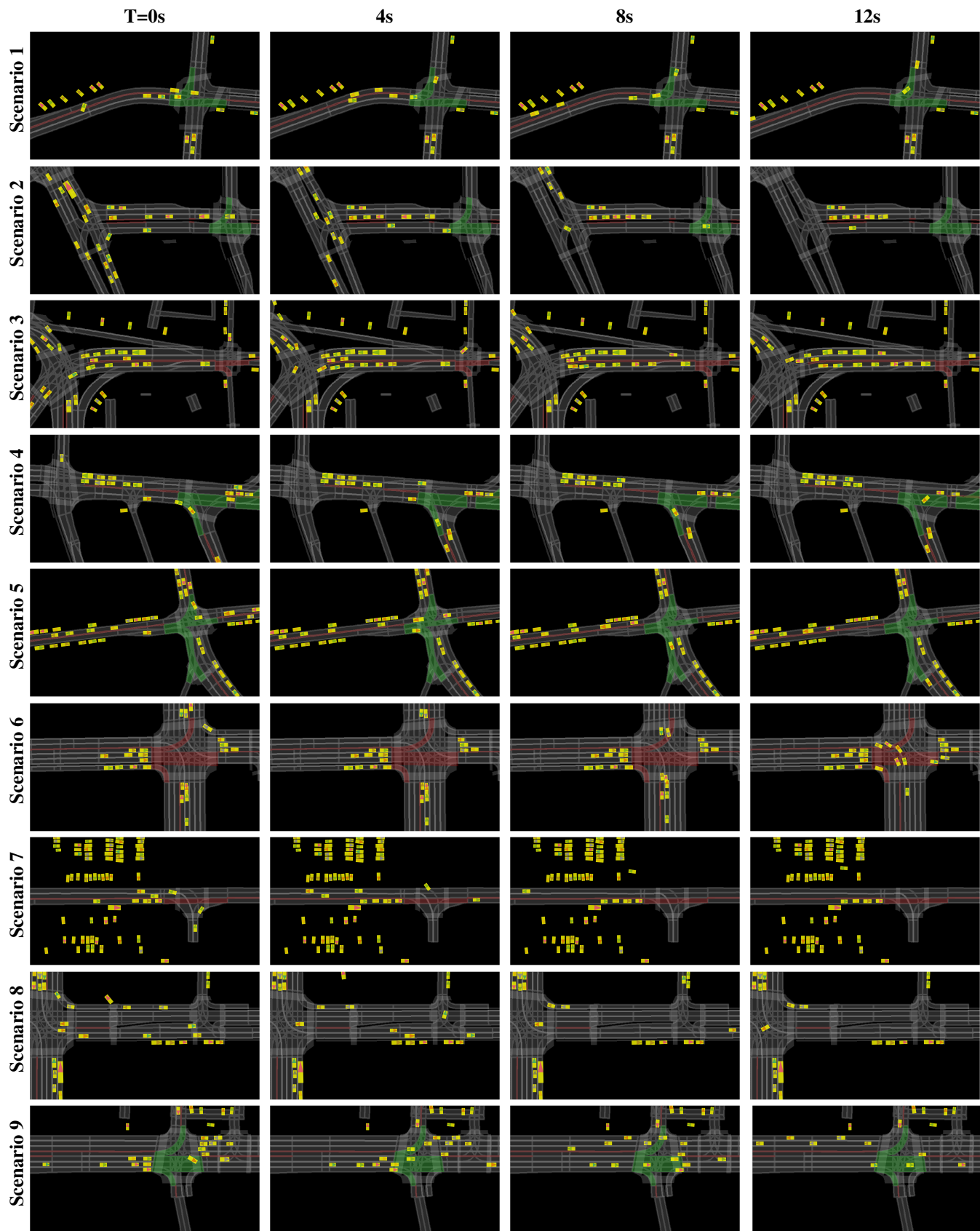


Figure 4: Simulated traffic scenarios sampled from TRAFFICSIM: colored triangle shows heading and tracks instances across time

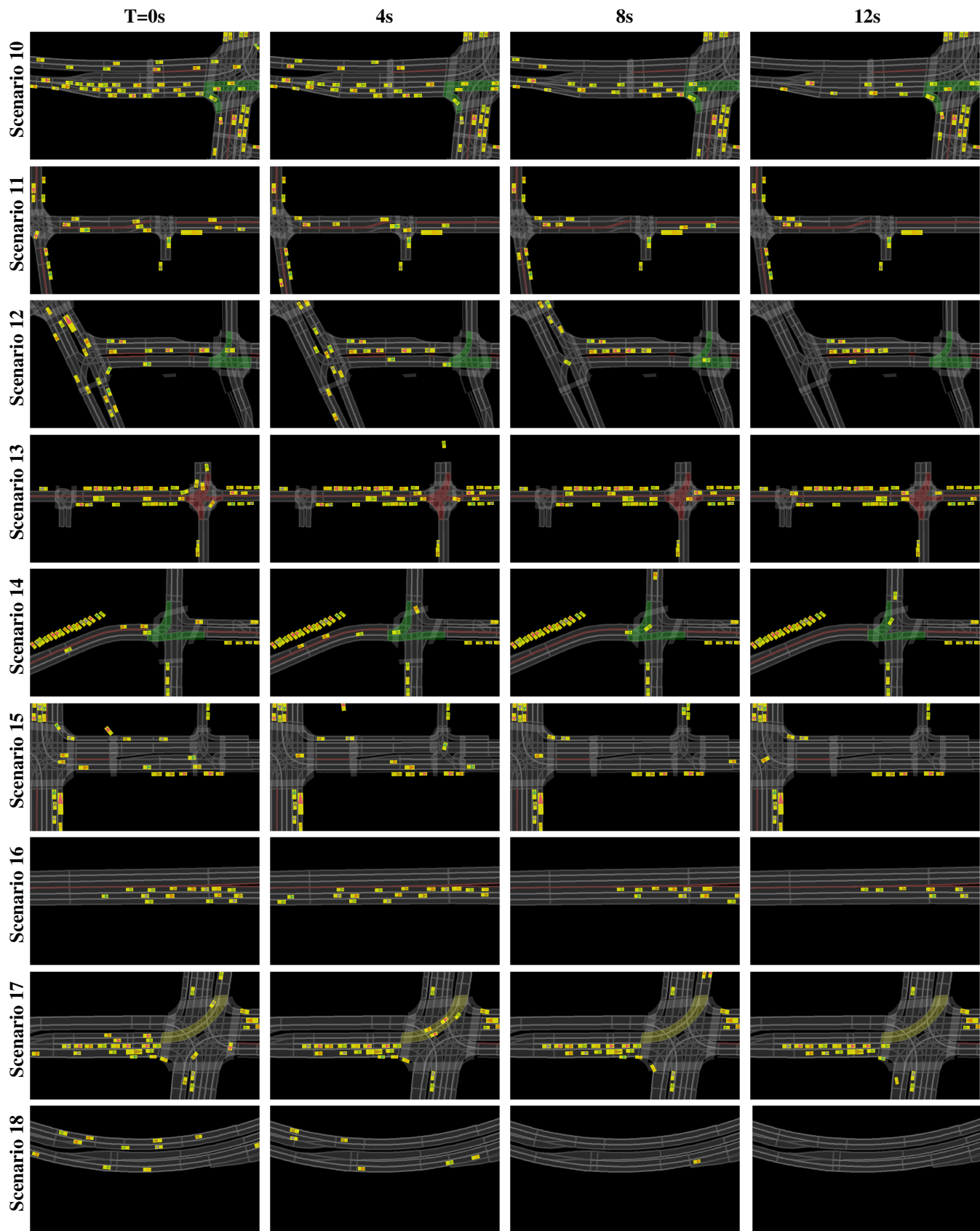


Figure 5: Simulated traffic scenarios sampled from TRAFFICSIM: colored triangle shows heading and tracks instances across time

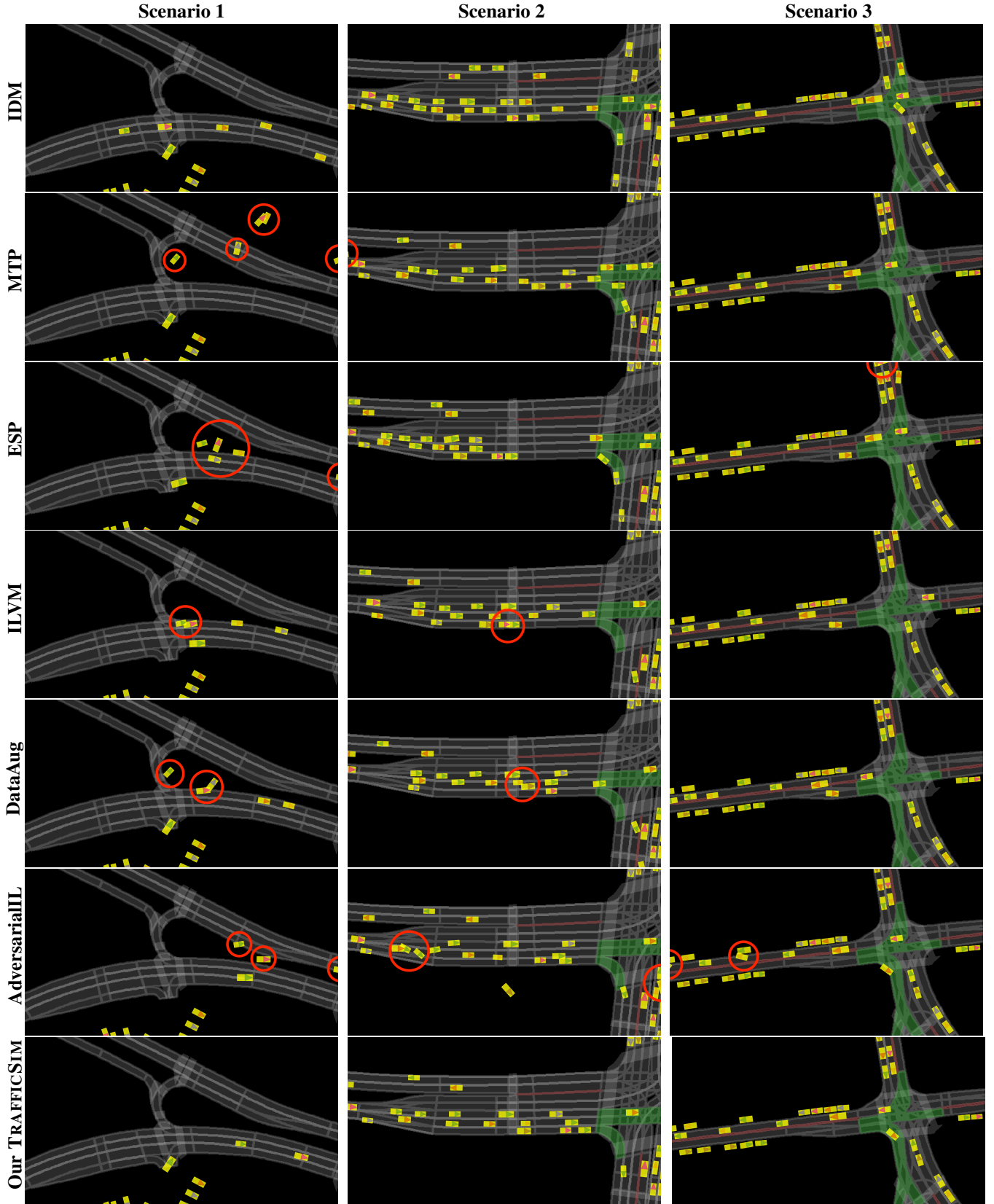


Figure 6: Comparison between traffic scenarios simulated by the baselines and our model. We show a snapshot at 6s after the start of the simulation. Red circles highlight collisions and traffic rule violations

References

- [1] Mayank Bansal, Alex Krizhevsky, and Abhijit Ogale. Chauffeurnet: Learning to drive by imitating the best and synthesizing the worst. In *Proceedings of Robotics: Science and Systems*, Freiburg/Breisgau, Germany, June 2019. 3
- [2] F. Behbahani, K. Shiarlis, X. Chen, V. Kurin, S. Kasewa, C. Stirbu, J. Gomes, S. Paul, F. A. Oliehoek, J. Messias, and S. Whiteson. Learning from demonstration in the wild. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 775–781, 2019. 3
- [3] R. P. Bhattacharyya, D. J. Phillips, C. Liu, J. K. Gupta, K. Driggs-Campbell, and M. J. Kochenderfer. Simulating emergent properties of human driving behavior using multi-agent reward augmented imitation learning. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 789–795, 2019. 3
- [4] Raunak P. Bhattacharyya, Derek J. Phillips, Blake Wulfe, Jeremy Morton, Alex Kuefler, and Mykel J. Kochenderfer. Multi-agent imitation learning for driving simulation. *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1534–1539, 2018. 3
- [5] Raunak P. Bhattacharyya, Blake Wulfe, D. Phillips, Alex Kuefler, J. Morton, Ransalu Senanayake, and M. Kochenderfer. Modeling human driving behavior through generative adversarial imitation learning. *ArXiv*, abs/2006.06412, 2020. 3
- [6] Sergio Casas, Cole Gulino, Renjie Liao, and R. Urtasun. Spaghn: Spatially-aware graph neural networks for relational behavior forecasting from sensor data. *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 9491–9497, 2020. 1
- [7] Sergio Casas, Cole Gulino, Simon Suo, Katie Luo, Renjie Liao, and Raquel Urtasun. Implicit latent variable model for scene-consistent motion forecasting. In *Proceedings of the European Conference on Computer Vision (ECCV)*. Springer, 2020. 1, 3
- [8] 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. *2019 International Conference on Robotics and Automation (ICRA)*, pages 2090–2096, 2019. 1, 3
- [9] Jonathan Ho and Stefano Ermon. Generative adversarial imitation learning. In *Proceedings of the 30th International Conference on Neural Information Processing Systems*, pages 4572–4580, 2016. 3
- [10] N. Rhinehart, R. Mcallister, K. Kitani, and S. Levine. Precog: Prediction conditioned on goals in visual multi-agent settings. In *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 2821–2830, 2019. 3
- [11] Martin Treiber, Ansgar Hennecke, and Dirk Helbing. Congested traffic states in empirical observations and microscopic simulations. *Physical Review E*, 62(2):1805–1824, Aug 2000. 3
- [12] Bin Yang, Wenjie Luo, and Raquel Urtasun. Pixor: Real-time 3d object detection from point clouds. In *Proceedings of the IEEE CVPR*, 2018. 1