Supplementary Material
MP3: A Unified Model to Map, Perceive, Predict and Plan

Sergio Casas[*,1,2], Abbas Sadat [*,1], Raquel Urtasun[1,2]
Uber ATG[1], University of Toronto[2]
{sergio, urtasun}@cs.toronto.edu, abbas.sadat@gmail.com

In this supplementary material we first explain our implementation details in depth, then showcase additional experiments to provide more insights from our model, and finally provide additional qualitative results.

Please check out our supplementary video for a short summary of our method as well as qualitative results with narrated comments.

1. Implementation details

1.1. Architecture

Backbone network: Fig. 1 shows an architecture diagram of the backbone network. We combine ideas from [1, 7] to build a multi-resolution backbone network that extracts geometric and semantic information from the past LiDAR sweeps and is able to aggregate them to reason about motion. Our backbone is composed of 4 convolutional blocks, and a convolutional header. The number of output features and kernel size is indicated in Fig. 1. All convolutional layers use Group Normalization [6] with 32 groups, and ReLU non-linearity. The scene context features after each residual block are $\mathcal{C}_{1x}, \mathcal{C}_{2x}, \mathcal{C}_{4x}, \mathcal{C}_{8x}$, where the subscript indicates the downsampling factor from the input in BEV. The features from the different blocks are then concatenated at 4x downsampling by max-pooling higher resolution ones $\mathcal{C}_{1x}, \mathcal{C}_{2x}$ and interpolating $\mathcal{C}_{8x}$. Finally, a residual block of 4 convolutional layers with no downsampling outputs the scene context $\mathcal{C}$. Since the LiDAR input is voxelized at 0.2 meters per pixel, $\mathcal{C}$ has a resolution of 0.8 meters per pixel.

Mapping architecture: In order to drive safely (e.g., narrow streets) we need a high resolution representation of our maps, and at the same time a big receptive field to help reduce uncertainty (e.g., around occlusions). To achieve this efficiently, we employ a multi-resolution architecture as shown in Fig. 2. It takes as input multiple feature maps $\mathcal{C}_{1x}, \mathcal{C}_{2x}, \mathcal{C}$ from the backbone network (see Fig. 1), and outputs the 6 channels of the online map at the original input resolution of 0.2 m/pixel. As a reminder, the six channels are: 1 for drivable area score, 1 for intersection score, 2 for truncated unsigned distance reachable lane (mean and variance of Gaussian distribution), 2 for the angle to closest reachable lane segment (location and concentration of Von Mises distribution).

Perception and Prediction architecture: The different dynamic classes (i.e. vehicles, pedestrians and bicyclists) are processed by different networks since they have very different geometry as well as motion. For each class, a 2-layer CNN processes $\mathcal{C}$ and upsamples it to 0.4 m/pixel. This is the same resolution as $\mathcal{C}_{2x}$, which gets processed by another 2-layer CNN. Then the two feature maps get concatenated to form the dynamic context. From this context, a 1-layer CNN outputs the current occupancy map, a 2-layer CNN the motion mode scores for all time steps, and another 2-layer CNN the motion vectors for all modes and future time steps. We employ dilation [8] to increase the receptive field while keeping the number of parameters low in order to be able to predict motion for voxels that are far away from the original position of actors for the long temporal horizons. To infer the future occupancy, we simply warp the initial occupancy with the temporal motion field as explained in the main manuscript's Fig. 3.4., and further detailed in Section 1.2. In practice, we found that $K = 3$ modes was expressive enough for the multi-modality of the temporal motion field. In all our experiments, $T = 11$ since we predict the future 5 seconds at 0.5-second intervals.

Routing architecture: In order to drive towards a goal, we would like to follow the driving commands. To do so, we predict a route spatial map, where each cell represents the probability that driving to it from the current location is aligned with the driving command. The architecture for the network that predicts such spatial map is detailed in 4. The high-level action in the command acts as a switch between 3 instantiations of the same network architecture (i.e., one for turning right, one for turning left, one for going

[*]Denotes equal contribution

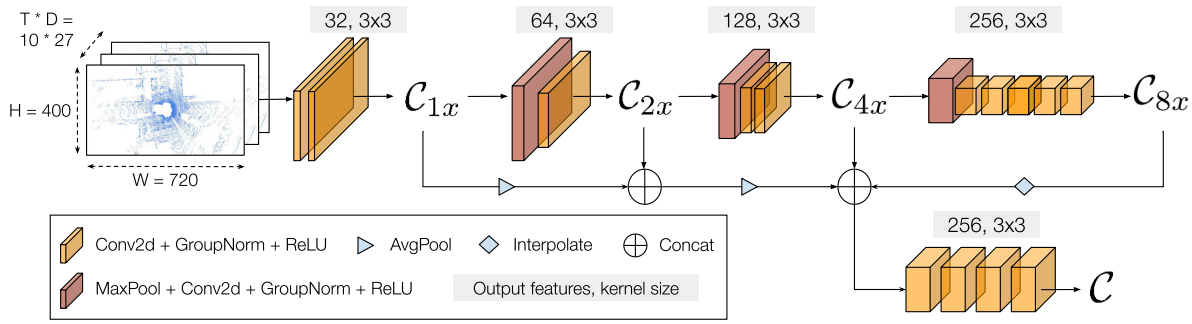

Figure 1: **Backbone Network**. The output features within one CNN block are fixed. All kernel strides and dilation are 1.

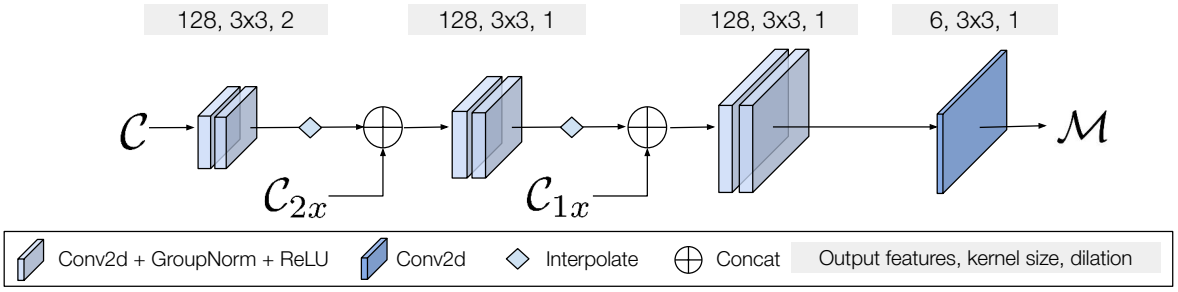

Figure 2: **Mapping Network**. The output features within one CNN block are fixed. All kernel strides are 1.

straight). The longitudinal distance to action is repeated spatially with the same resolution as the online map. Then, both are concatenated to form the input to a CNN that leverages Coordinate Convolutions (CoordConv) [2] in order to be able to reason about the distance to a particular grid cell from the SDV.

1.2. Dynamic occupancy

Here we provide a more detailed explanation of how our dynamic occupancy flow works than in the main manuscript. We first define a *flow event* from spatio-temporal grid cell (t, i_1) to $(t+1, i_2)$ as the intersection of the event that the original grid cell is occupied, and that the motion field moves this occupancy from (t, i_1) to $(t+1, i_2)$. Since we consider K motion modes, the final flow event considers the union of those, effectively marginalizing over modes:

$$\mathcal{F}_{(t,i_1)\to(t+1,i_2)} = \cup_k \{\mathcal{O}^t_{i_1} \cap \mathcal{K}^t_{i_1} = k \cap \mathcal{V}^{t,k}_{i_1} = i_2\}$$

Given this flow event definition and the assumptions in our probabilistic model, we can obtain its probability:

$$p(\mathcal{F}^c_{(t,i_1)\to(t+1,i_2)}) = \sum_k p(\mathcal{O}^c_{t,i_1}) p(\mathcal{K}^c_{t,i_1} = k) p(\mathcal{V}^c_{t,i_1,k} = i_2)$$

We can now calculate the future occupancy iteratively, starting from the occupancy predictions at $t = 0$. Specifically, to get the occupancy that flows into cell i at time $t+1$ from all cells j at time t, we can simply compute the probability that no occupancy flow event occurs, and take its complement

$$\begin{aligned} p(\mathcal{O}^c_{t+1,i}) &= p\big(\cup_j \mathcal{F}_{(t,j)\to(t+1,i)} : \forall k, l\big) \\ &= 1 - p\big(\cap_j \mathcal{F}'_{(t,j)\to(t+1,i)} : \forall k, l\big) \\ &= 1 - \prod_j \big(1 - p(\mathcal{F}^c_{(t,j)\to(t+1,i)})\big) \end{aligned}$$

where $\mathcal{F}'_{(t,j)\to(t+1,i)}$ denotes the complement of $\mathcal{F}_{(t,j)\to(t+1,i)}$.

1.3. Planning

In this section we provide more details about trajectory retrieval and the coring functions.

1.3.1 Trajectory Retreival

We use ~ 150 hours of manual driving data to create the dataset of expert trajectories. To group the trajectories into different bins, we use the initial velocity v, curvature κ, and acceleration a, with respective bin sizes of $2.0\frac{m}{s}$, $0.02\frac{1}{m}$, and $1.0\frac{m}{s^2}$. The trajectories in each bins are clustered into $3,000$ sets and the closest trajectory to each cluster prototype is kept. This creates a set fo diverse trajectories conditioned on the current state of SDV. Figure 5 shows example sets of trajectories retrieved base on the indicated initial state (initial acceleration is 0.0 in all the cases). We can see how the set

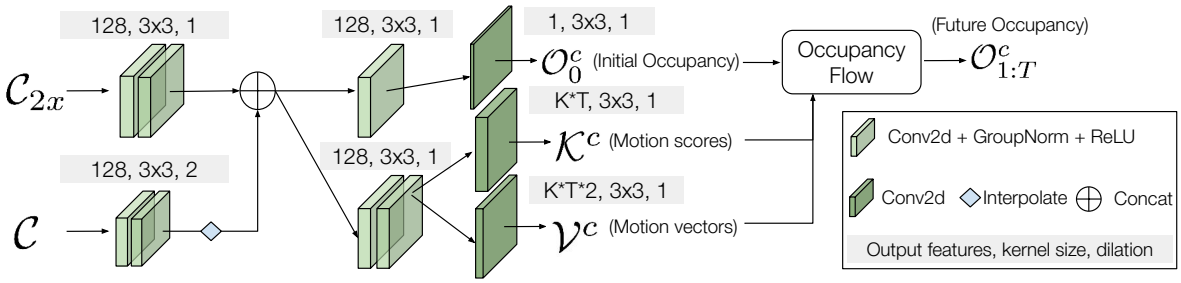

Figure 3: **Perception and Prediction Network**. It outputs the initial occupancy and current and future motion. The output features within one CNN block are fixed. All kernel strides are 1.

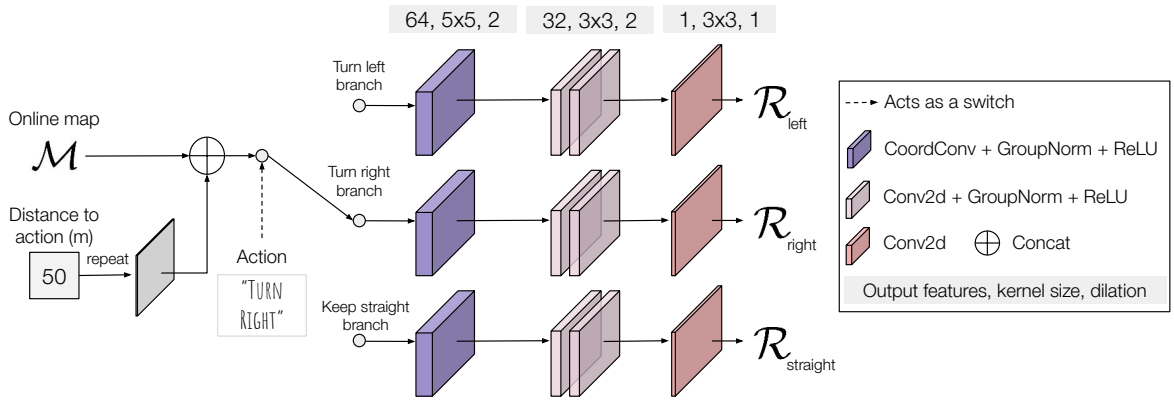

Figure 4: **Routing Network**. The output features within one CNN block are fixed. All kernel strides are 1. The architecture is the same for the 3 branches, but with different learnable parameters.

of trajectories are influenced by the initial state, resulting in kinematically-plausible trajectory sampling.

1.3.2 Trajectory Scoring

In the following, we provide more details about the cost function of the planner.

Reachable-lanes direction In addition to stay close to the lane centerlines, we promote trajectories that stay aligned with the direction of the lane. Towards this goal, we use the average difference of the trajectory point heading and the angles in $\mathcal{M}^\theta$ indexed at all BEV grid-cells that have overlap with SDV polygon.

$$f_d(\mathbf{x}, \mathcal{M}) = \mathop{\mathbb{E}}_{i \in m(\mathbf{x})} |\mathcal{M}_i^\theta - \mathbf{x}_\theta|$$

where $m(\mathbf{x})$ represents the spatial indices of BEV grid-cells of the map-layer prediction that overlaps with SDV polygon with state $\mathbf{x}$.

Lane uncertainty In order to promote cautious behavior when there is high uncertainty in $\mathcal{M}^D$ and $\mathcal{M}^\theta$, we use a cost function that is the product of the SDV velocity and the standard deviation of the probability distributions of cells overlapping with SDV in $\mathcal{M}^D$ and $\mathcal{M}^\theta$. This promotes slow maneuver in the presence of map uncertainty:

$$f_d(\mathbf{x}, \mathcal{M}^\theta, \mathcal{M}^D) = \sum_{i \in m(\mathbf{x})} \mathbf{x}_v (\sigma_i^D + \frac{1}{k_i^\theta})$$

Here σ_i^D denotes the standard deviation of the Gaussian distribution representing distance to closest reachable lane center, and k_i^θ is the concentration parameter of the von Mises distribution representing lane direction.

Occupancy Given the state of the ego car $\mathbf{x}$ for a single time-step, we use the following cost function to penalize trajectories that overlap with occupied regions:

$$f_o(\mathbf{x}_t, \mathcal{O}) = \sum_c \max_{i \in m(\mathbf{x}_t)} P(\mathcal{O}_{t,i}^c)$$

where $m(\mathbf{x}_t)$ represents the BEV grid-cells, with semantic class c, that have overlap with the polygon of SDV with state $\mathbf{x}_t$.

Headway For the headway cost, we retrieve the set of BEV grid-cells $m(\mathbf{x}_t)$ that are within 20m in front of the

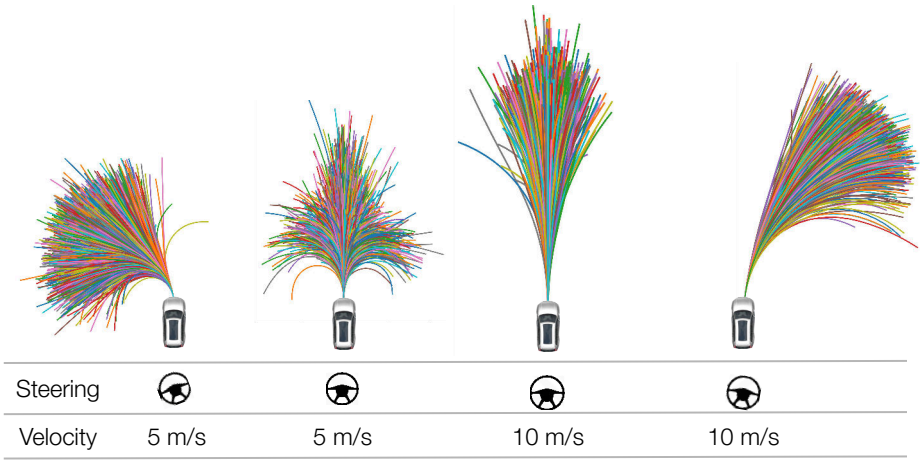

Figure 5: Sets of trajectories retrieved from the expert demonstrations. The initial state of the SDV is used to query trajectories that start with similar state (the initial acceleration is $0.0\frac{m}{s^2}$ in all the shown cases).

Loss	Success (%)↑	OffRoute (%)↓	L2 (m)↓	Progress per event (m) ↑					Comfort	
				any event	collision	off-road	off-route	oncoming	jerk($\frac{m}{s^3}$)↓	lat.acc.($\frac{m}{s^2}$)↓
Motion	68.90	**11.59**	13.10	171.53	545.59	991.81	**514.30**	1445.84	1.73	**0.10**
+ Occ.	**74.39**	14.63	**12.95**	**218.40**	**1037.08**	**1136.49**	409.34	**1465.27**	**1.64**	**0.10**

Table 1: **Loss ablation for future dynamic occupancy** (closed-loop simulation results). Adding supervision to the warped occupancy drastically improves our model, compared to supervising only the motion field.

SDV at time t. The headway cost is then computed over this set as follows:

$$f_h(\mathbf{x}_t, \mathcal{O}, \mathcal{K}, \mathcal{V}) = \sum_{i \in m(\mathbf{x}_t)} P(\mathcal{O}_{t,i}) \mathop{\mathbb{E}}_{P(\mathcal{K}_{t,i})} [h(\mathbf{x}_t, \mathcal{V}_{t,i})]$$

where the expectation is over the different motion modes. The function $h(\mathbf{x}_t, \mathcal{V}_{t,i})$ measures the violation of safety distance if the object at spatial index i with speed $\mathcal{V}_{t,i}$ stops with hard deceleration, and SDV with state $\mathbf{x}_t$ reacts with a comfortable deceleration.

1.4. Training details

Multi-task learning (1st stage): In the first stage we train the backbone, mapping, perception and prediction as well as routing networks. To do so, we linearly combine the mapping loss $\mathcal{L}_{\mathcal{M}}$, occupancy loss $\mathcal{L}_{\mathcal{O}}$, motion loss $\mathcal{L}_{\mathcal{K},\mathcal{V}}$, and routing loss $\mathcal{L}_{\mathcal{R}}$ described in the main manuscript.

$$\mathcal{L}_{\text{stage 1}} = \mathcal{L}_{\mathcal{O}} + \lambda_{\mathcal{K},\mathcal{V}} \mathcal{L}_{\mathcal{K},\mathcal{V}} + \lambda_{\mathcal{M}} \mathcal{L}_{\mathcal{M}} + \lambda_{\mathcal{R}} \mathcal{L}_{\mathcal{R}}$$

where $\lambda_{\mathcal{K},\mathcal{V}} = 0.1, \lambda_{\mathcal{M}} = 0.5, \lambda_{\mathcal{R}} = 2.0$ are hyperparameters that were found to work well in practice in our validation set. Within these task losses, all losses are summed with equal weight (e.g., for mapping, the negative-log likelihood for each map element has the same weight).

Trajectory Scoring (2nd stage): Since selecting the minimum-cost trajectory within a discrete set is non-differentiable, we use the max-margin loss [3,4] to penalize trajectories that have small cost but differ from the human demonstration or are unsafe. Let τ_h be the expert demonstration for a given example. The max-margin loss encourages the human driving trajectory to have smaller cost f than other trajectories.

$$\mathcal{L}_M = \max_{\tau} \left[f_r(\tau_h) - f_r(\tau) + l_{\text{im}} + \sum_t \left[f_o^t(\tau_h) - f_o^t(\tau) + l_{\text{o}}^t \right]_+ \right]_+$$

where f_o^t is the occupancy cost function at time step t, f_r are the rest of the planning subcosts, and $[]_+$ represents the ReLU function. Note that in above, we dropped the other inputs to the cost functions for brevity. The imitation task-loss l_{im} measures the ℓ_1 distance between trajectory τ and the ground-truth for the entire horizon, and the safety task-loss l_{o}^t accounts for collisions and their severity at each trajectory step. By imposing the task-loss per time-step and separate from the other non-safety subcosts, we make sure the margin in cost is achieved when the trajectory is in collision (high l_{o}^t) irrespective of other costs at other time-steps.

2. Additional Experiments

2.1. Dynamic occupancy parameterization ablation

Our network only predicts the occupancy at t=0, and the temporal motion field from t=0 to t=5s. The future occupancy at times $\{0.5, 1.0, \ldots, 5.0\}$s is computed by warping the initial occupancy with the motion field. In this ablation, we compare this with the parameterization in [4]. For behavior prediction metrics, the occupancy average F1 score over time (0-5s) in URBANEXPERT's test set is 73.7% for MP3, and 71.3% for [4] on vehicles, and 46.3% vs. 44.1%

Routing input	Success (%)↑	OffRoute (%)↓	L2 (m)↓	Progress per event (m) ↑					Comfort	
				any event	collision	off-road	off-route	oncoming	jerk($\frac{m}{s^3}$)↓	lat.acc.($\frac{m}{s^2}$)↓
None	46.95	44.51	22.04	93.44	**1614.09**	1106.78	116.52	1402.32	1.85	**0.07**
+ action	67.68	19.51	13.47	166.58	935.18	**1255.28**	296.68	1132.53	1.75	0.09
+ dist.	**74.39**	**14.63**	**12.95**	**218.40**	1037.08	1136.49	**409.34**	**1465.27**	**1.64**	0.10

Table 2: **Route ablation** (closed-loop simulation results). Naturally, adding a discrete route command (action) allows the SDV to better progress towards the goal. Moreover, complementing it with a noisy longitudinal distance to action is helpful.

Map & route	Success (%)↑	OffRoute (%)↓	L2 (m)↓	Progress per event (m) ↑					Comfort	
				any event	collision	off-road	off-route	oncoming	jerk($\frac{m}{s^3}$)↓	lat.acc.($\frac{m}{s^2}$)↓
GT	84.80	0.0	9.19	415.41	684.61	1822.48	∞	3689.89	1.53	0.15
Pred.	74.39	14.63	12.95	218.40	1037.08	1136.49	409.34	1465.27	1.64	0.10

Table 3: **Upper bound performance** in closed-loop simulation when using an HD map to feed the motion planner the true online map and route.

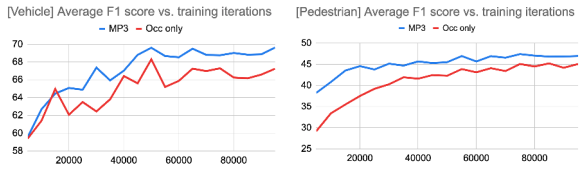

Figure 6: **Dynamic occupancy parameterization ablation**. Validation during training curves that showcase the power of our dynamic occupancy when compared against that of [4].

on pedestrians. Moreover, the inductive bias of our flow allows the model to learn faster, or equivalently with less data (see validation curves in Fig. 6). These gains in occupancy, together with the additional motion field output (used to compute the headway score), result in better closed loop driving with improved safety (>2x m. per collision) and closer imitation to human ($\sim 10\%$ in L2).

2.2. Dynamic occupancy loss ablation

Our proposed model predicts multi-modal motion vectors and their categorical distribution for each spatio-temporal BEV grid cell, which is then used to flow the initial predicted occupancy to the future steps. The most direct way of supervising the predicted elements, i.e., initial occupancy and temporal motion fields, is to have a separate loss for each as they both have supervision from the labels, similar to the loss in [5]. However, this has 2 main drawbacks. First, this training strategy does not optimize future occupancy, which is critical for safety in motion planning. Because the future occupancy is obtained as a resulting of flowing (or warping) the initial occupancy with the temporal motion field, small errors in motion can accumulate over time. Second, only the motion vector that is closest to the ground-truth gets supervision, leaving the rest of the distribution over possible motions unsupervised. We implement this strategy and ablate it against our training, where we also apply supervision to the future occupancy after flow, thus optimizing for what we care about for safe driving, and also giving signal to the full motion distribution. The results shown in Table 1 show a clear advantage of our training strategy. We would like to highlight that this improvement in the loss makes the model roughly twice as safe, as indicated by the progress per collision metric. While it is true that we suffer a minor regression in route following probably to avoid some collisions by deviating laterally, safety takes precedence.

2.3. Routing ablation

In order to show the importance of route prediction, we consider (i) no route prediction, (ii) predicting the route map based only on the input discrete high-level action coming from the command through switching between different NNs, (iii) also taking into account the approximate longitudinal distance to action as explained in Fig. 4. Our results in Table 2 show several interesting aspects. First, we see that only by using the discrete high-level action (Action only), which is the same information as the CIL and CNMP baselines use, our model is able to achieve a much better route following than those. In particular, while the best baseline, CNMP, went 47.56% of the times off-route, our *MP3 - Action only* ablation only goes out of route 19.51% of the times. Finally, we see that adding the approximate distance to action and using CoordConv [2] such that the routing network can reason about the distance from SDV further reduces the out-of-route events to 14.63%. We note that the decrease in progress per collision we observe when adding the route is due to the fact that the base model is not constrained to the route, and therefore can swerve outside the route without cost to avoid collisions. In that case, our metrics only record an off-route event for the breakdown. However, doing this consistently would make any travel very frustrating since it would make it impossible to reach the destination in time, and thus this is not a reasonable option.

2.4. Upper bound with access to HD map

Table 3 showcases the results of our model when feeding the motion planner with the ground-truth online map and route map, instead of the predicted ones. For this experiment we only retrain the motion planner weights for trajectory scoring (i.e. the second stage in our training described in the main manuscript). We note that the planner is still not receiving a perfect representation of the scene, as it needs to infer the dynamic occupancy from sensor data. This experiment shows several interesting aspects. We can see that with a perfect map and route prediction, the proposed motion planner (including the retrieval-based trajectory sampler) always follows the route. This experiment also confirms what we have observed in the other ablations: sometimes diverging from the route is an easier way to avoid collisions than a change in the speed profile, as shown by the progress per collision metric. Finally, this experiment confirms that the proposed representations for the online map and the route, as well as the motion planning costs are adequate, and motivates future work to improve the prediction of the static part of the environment.

3. Qualitative results

3.1. MP3 outputs in closed-loop simulation

Figures 7, 8 showcase a solid understanding of both the static and dynamic parts of the environment through the online map and dynamic occupancy predictions. These translate into good routings and safe maneuvers from the motion planner that are close to the expert demonstrations even after unrolling our own plans for several seconds and therefore deviating from the expert state.

Moreover, Figs. 9, 10, 11, 12 provide further insight into the motion planner by additionally showing (a random subset of) the retrieved trajectory samples as well as their cost in a color map ranging from blue (lowest cost) to red (highest cost) in the top right image. The optimal trajectory (i.e. the one with the lowest cost) is plotted separately in the top left image.

3.2. Plan comparison against baselines in closed-loop simulation

Figures 13, 14, 15 compare the plans from our method and those of the baselines in 3 different scenarios from our closed-loop simulations. In these figures, the expert driver state is shown in black for reference, and the plans in blue. The path that should be followed by obeying the high-level commands is shown in orange. Each column is a method, and the rows represent a sequence of frames from a video (1 snapshot every 2.5 seconds of execution).

Please see our supplementary video for more comparisons.

References

[1] Sergio Casas, Wenjie Luo, and Raquel Urtasun. Intentnet: Learning to predict intention from raw sensor data. In *CoRL*, 2018. 1

[2] Rosanne Liu, Joel Lehman, Piero Molino, Felipe Petroski Such, Eric Frank, Alex Sergeev, and Jason Yosinski. An intriguing failing of convolutional neural networks and the coordconv solution. In *Advances in Neural Information Processing Systems*, pages 9605–9616, 2018. 2, 5

[3] Nathan D Ratliff, J Andrew Bagnell, and Martin A Zinkevich. Maximum margin planning. In *ICML*, 2006. 4

[4] Abbas Sadat, Sergio Casas, Mengye Ren, Xinyu Wu, Pranaab Dhawan, and Raquel Urtasun. Perceive, predict, and plan: Safe motion planning through interpretable semantic representations. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2020. 4, 5

[5] Pengxiang Wu, Siheng Chen, and Dimitris N Metaxas. Motionnet: Joint perception and motion prediction for autonomous driving based on bird's eye view maps. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11385–11395, 2020. 5

[6] Yuxin Wu and Kaiming He. Group normalization. In *Proceedings of the European conference on computer vision (ECCV)*, pages 3–19, 2018. 1

[7] Bin Yang, Wenjie Luo, and Raquel Urtasun. Pixor: Real-time 3d object detection from point clouds. In *CVPR*, 2018. 1

[8] Fisher Yu and Vladlen Koltun. Multi-scale context aggregation by dilated convolutions. *arXiv preprint arXiv:1511.07122*, 2015. 1

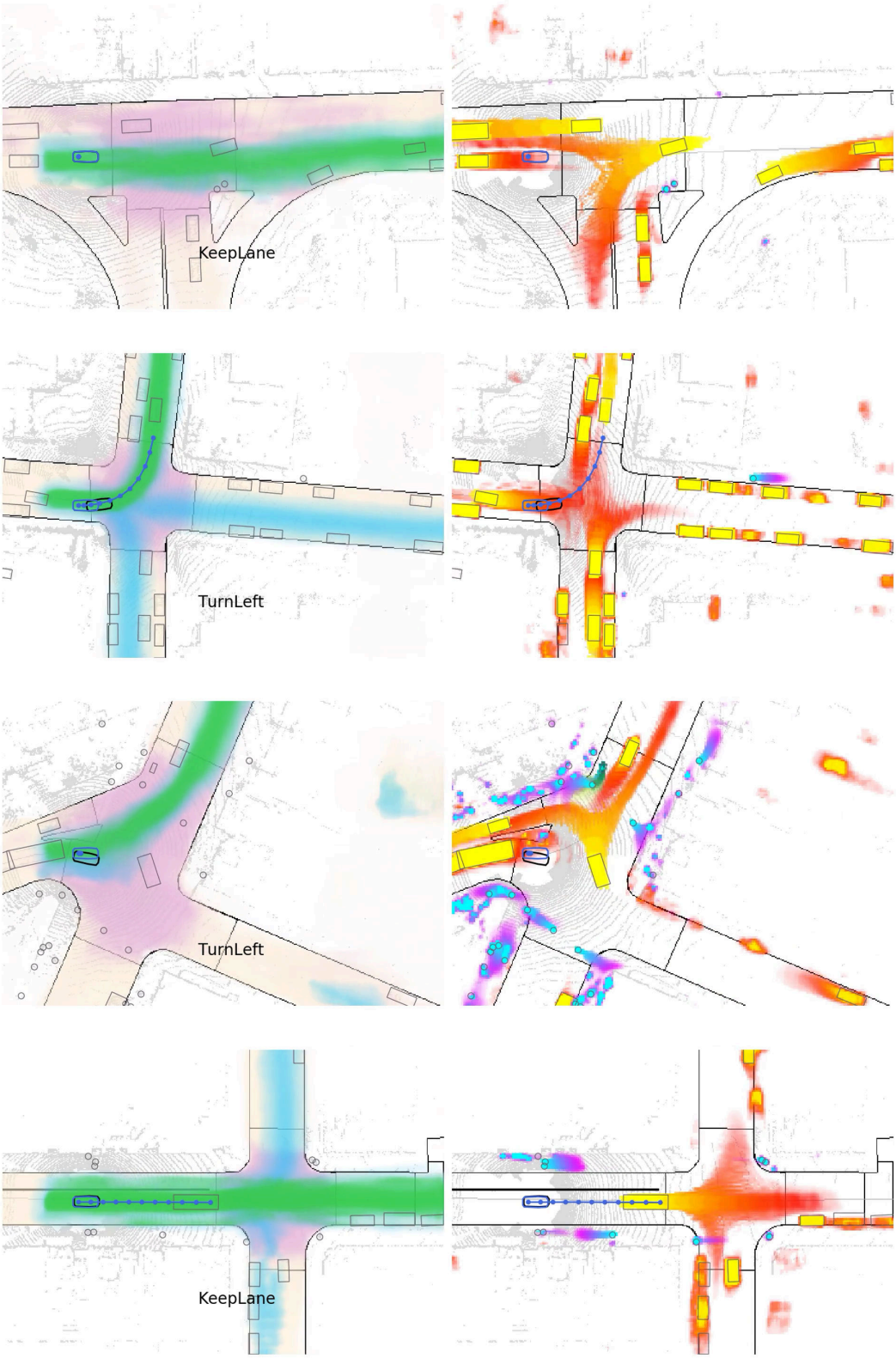

Figure 7: **Qualitative results of MP3 in closed-loop.** We show our predicted scene representations and motion plan in different interesting scenarios. The black bounding box represents the state of the expert driver at that point, and we can see it's very close to the planner state (in blue)

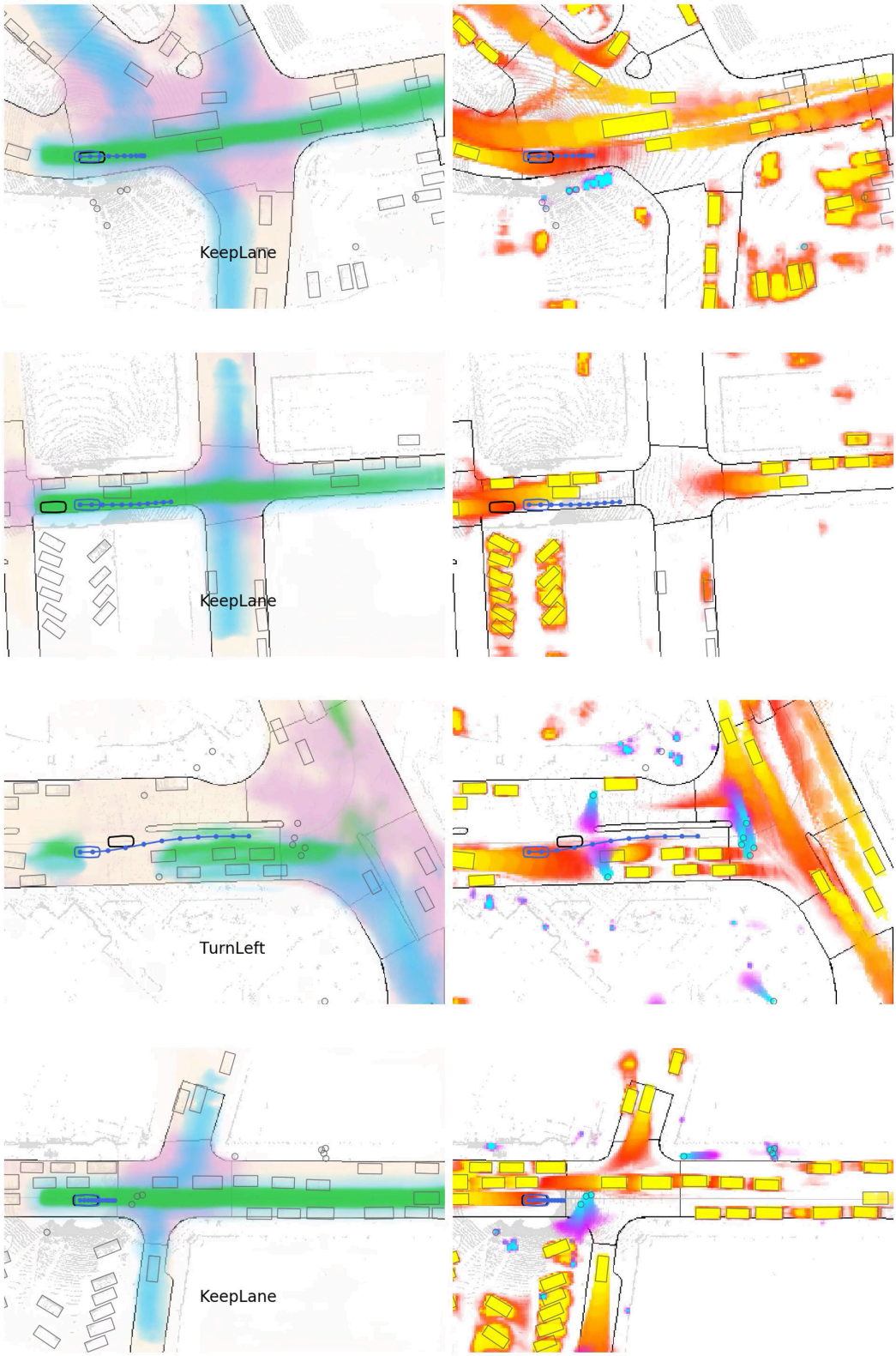

Figure 8: **More qualitative results of MP3 in closed-loop.**

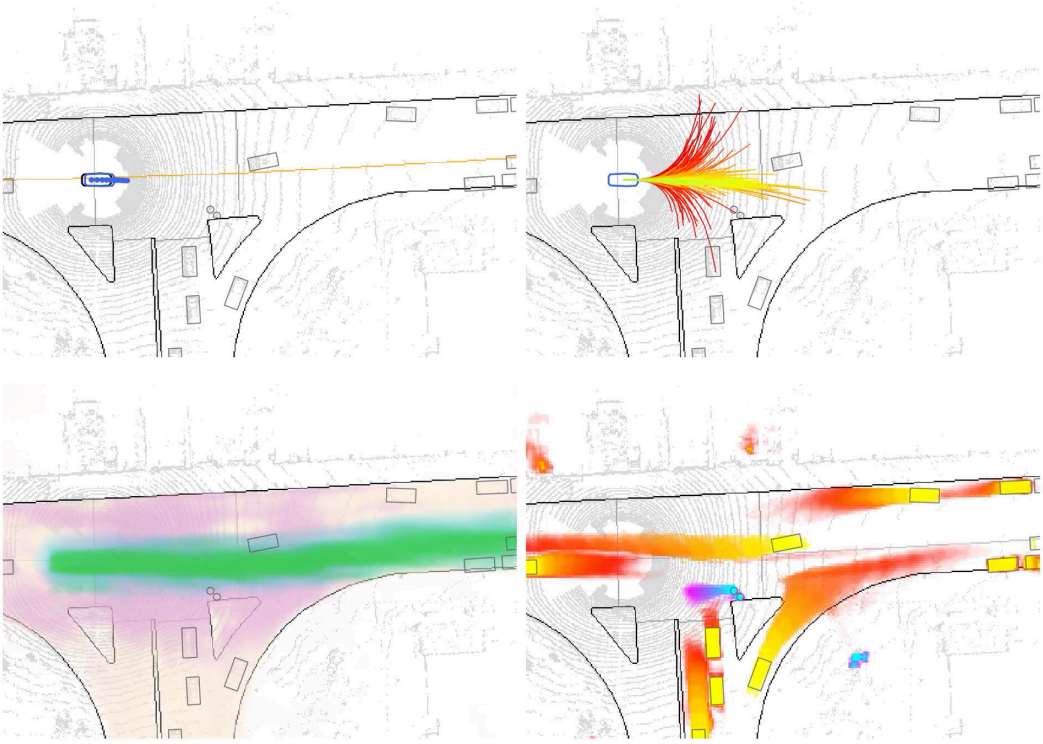

Figure 9: **Yielding scenario at an intersection**. Top right showcases a visualization of the cost of the retrieved trajectory samples. The warmer the color the higher the cost.

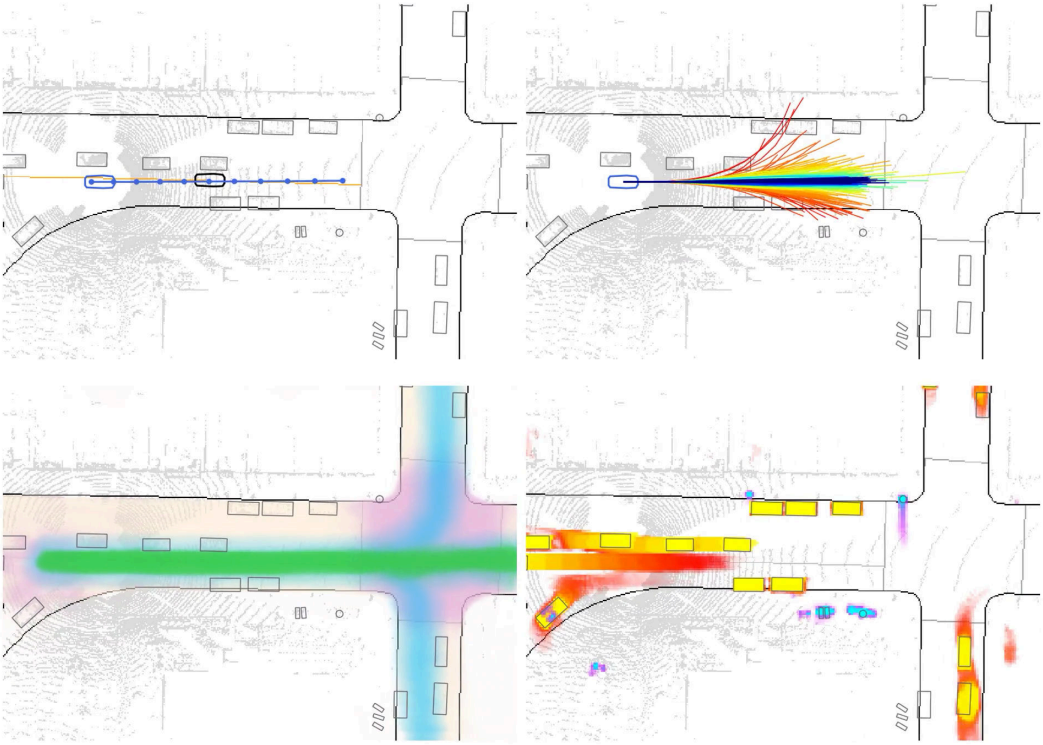

Figure 10: **Cruising scenario**. As can be seen in the top right, the fast moving straight trajectories achieve the lowest score as there is no one in front of the SDV.

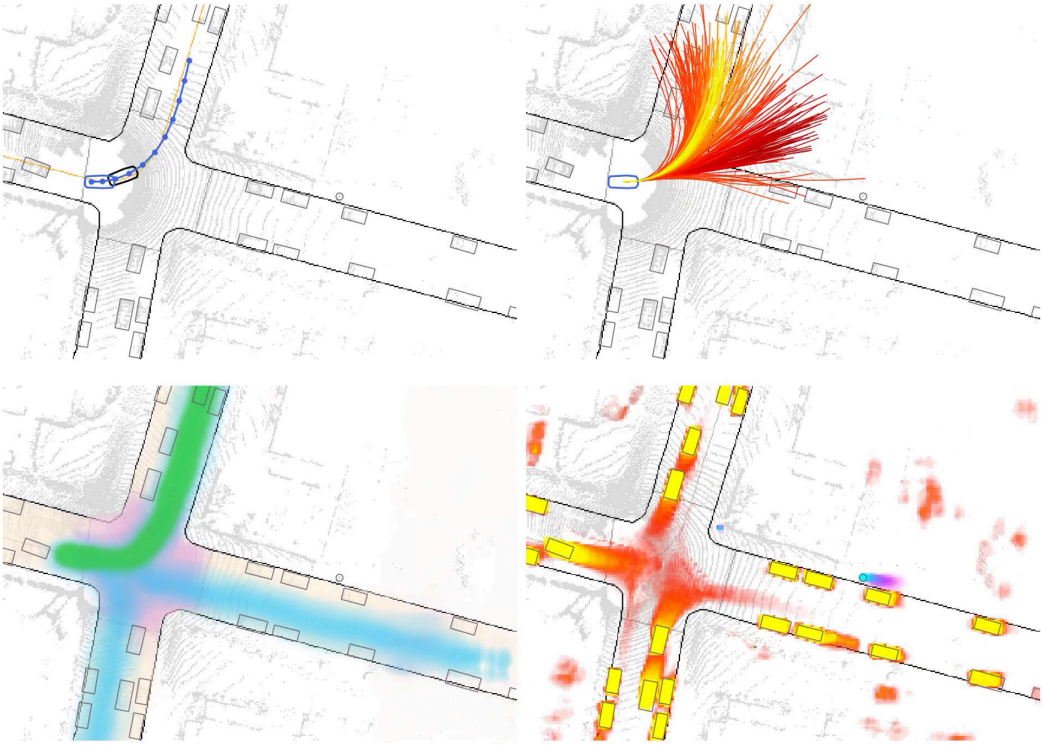

Figure 11: **Left turn scenario** in which the SDV progresses fast.

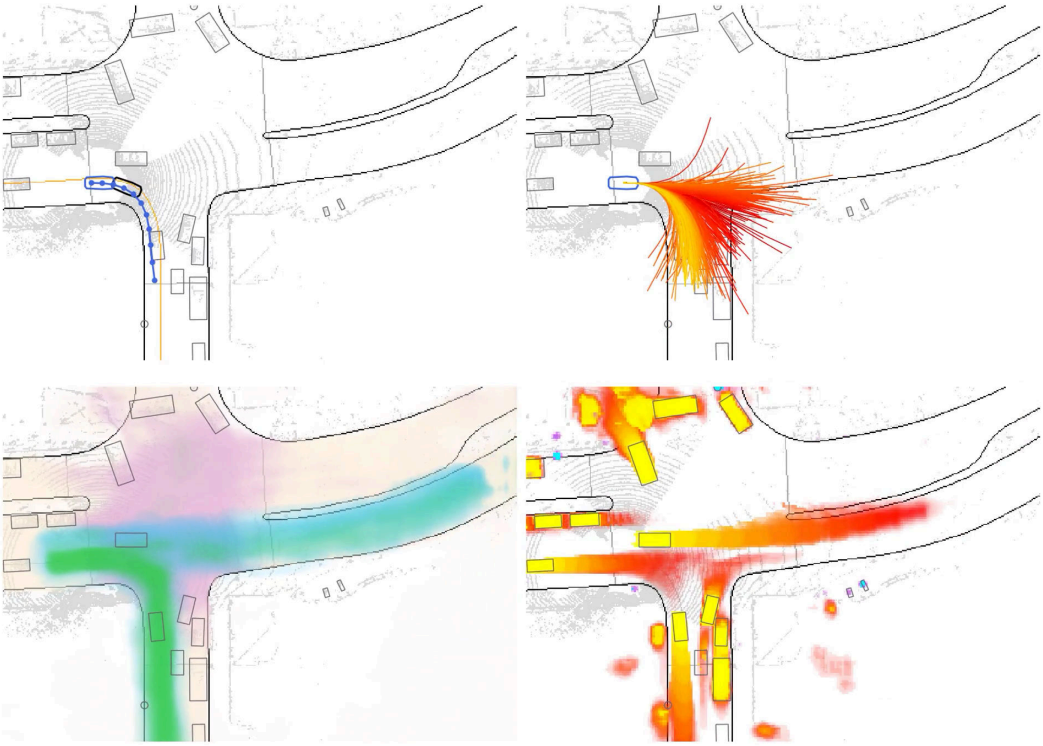

Figure 12: **Right turn scenario.** We can see how the trajectory samples adapt to the current SDV velocity.

Figure 13: **Planning comparison in closed-loop** MP3 is the only method that follows the route and does not collide in this example.

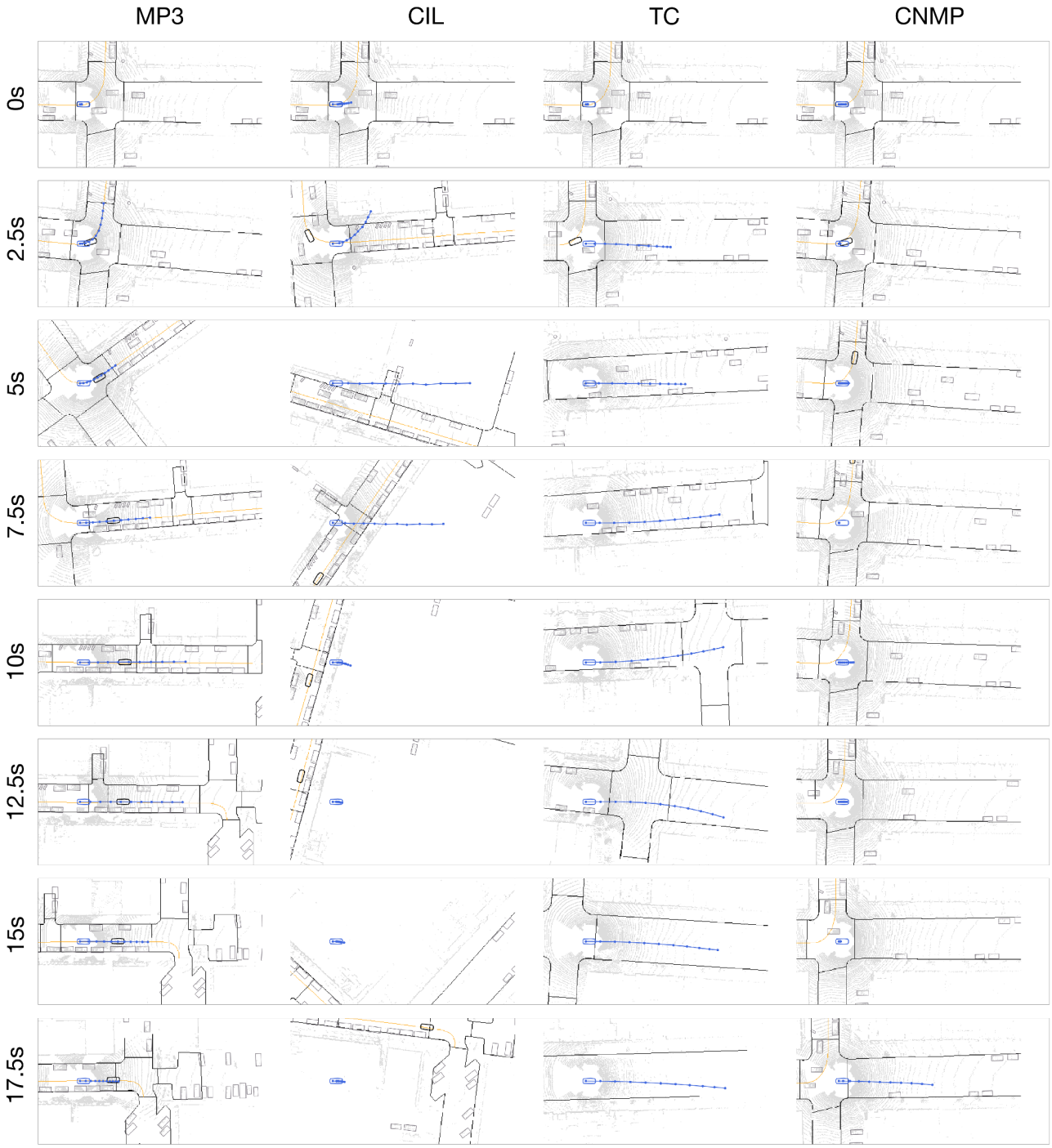

Figure 14: **More planning comparison in closed-loop**. MP3 is the only method that manages to effectuate the unprotected left turn

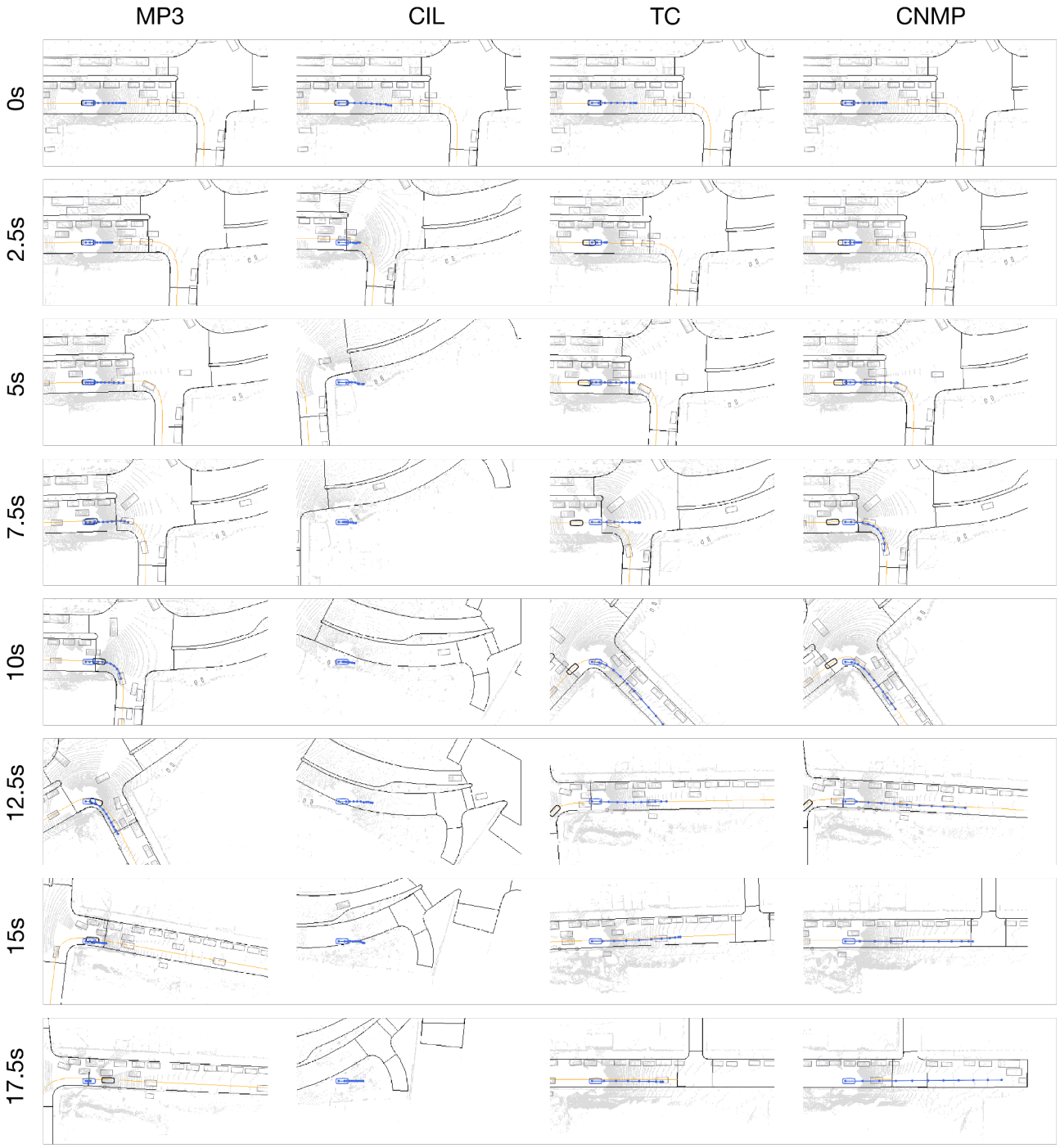

Figure 15: **More planning comparison in closed-loop**. MP3 closely imitates the expert. CIL diverges from the route. TC and CNMP stay on the route but collide with other vehicles.