

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

Sergio Casas^{*,1,2}, Abbas Sadat^{*,1}, Raquel Urtasun^{1,2}
Uber ATG¹, University of Toronto²

{sergio, urtasun}@cs.toronto.edu, abbas.sadat@gmail.com

Abstract

High-definition maps (HD maps) are a key component of most modern self-driving systems due to their valuable semantic and geometric information. Unfortunately, building HD maps has proven hard to scale due to their cost as well as the requirements they impose in the localization system that has to work everywhere with centimeter-level accuracy. Being able to drive without an HD map would be very beneficial to scale self-driving solutions as well as to increase the failure tolerance of existing ones (e.g., if localization fails or the map is not up-to-date). Towards this goal, we propose MP3, an end-to-end approach to mapless¹ driving where the input is raw sensor data and a high-level command (e.g., turn left at the intersection). MP3 predicts intermediate representations in the form of an online map and the current and future state of dynamic agents, and exploits them in a novel neural motion planner to make interpretable decisions taking into account uncertainty. We show that our approach is significantly safer, more comfortable, and can follow commands better than the baselines in challenging long-term closed-loop simulations, as well as when compared to an expert driver in a large-scale real-world dataset.

1. Introduction

Most modern self-driving stacks require up-to-date high-definition (HD) maps that contain rich semantic information necessary for driving such as the topology and location of the lanes, crosswalks, traffic lights, intersections as well as the traffic rules for each lane (e.g., unprotected left, right turn on red, maximum speed). These maps are a great source of knowledge that simplify the perception and motion forecasting tasks, as the online inference process has to mainly focus on dynamic objects (e.g., vehicles, pedestrians, cyclists). Furthermore, the use of HD maps significantly increases the safety of motion planning as knowing the lane topology and geometry eases the generation of potential trajectories for

*Denotes equal contribution

¹We note that by *mapless* we mean without HD maps. A coarse road network like the ones available in off-the-shelf services such as Google Maps or OpenStreetMap is assumed available for routing towards the goal.

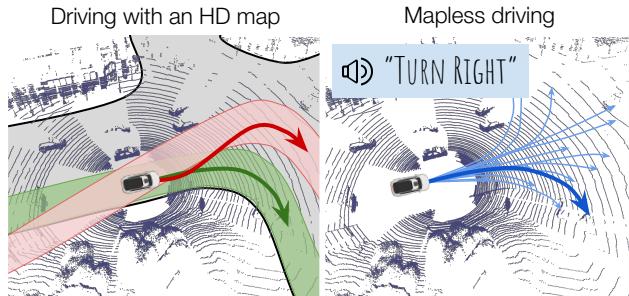


Figure 1: Left: a localization error makes the SDV follow a wrong route when using an HD map, driving into traffic. Right: mapless driving can interpret the scene from sensors and achieve a safe plan that follows a high-level command.

the ego-vehicle that adhere to the traffic rules. In addition, progressing towards a specific goal is much simpler when the desired route is defined as a sequence of lanes to traverse.

Unfortunately, building HD maps has proven hard to scale due to the complexity and cost of generating the maps and maintaining them. Furthermore, the heavy reliance on HD maps introduces very demanding requirements for the localization system, which needs to work at all times with centimeter-level accuracy or else unsafe situations like Fig. 1 (left) might arise. This motivates the development of mapless technology, which can serve as the fail-safe in the case of localization failures or outdated maps, and potentially unlock self-driving at scale at a much lower cost.

Self-driving without HD maps is a very challenging task. Perception can no longer rely on the prior that is more likely to find vehicles on the road and pedestrians on the sidewalk. Motion forecasting of dynamic objects becomes even more challenging without having access to the lanes that vehicles typically follow or the location of crosswalks for pedestrians. Most importantly, the search space to plan a safe maneuver for the SDV goes from narrow envelopes around the lane center lines [1, 45, 46, 50] to the full set of dynamically feasible trajectories as depicted in Fig. 1 (right). Moreover, without a well-defined route as a series of lanes to follow, the goal that the SDV is trying to reach needs to be abstracted into high-level behaviors such as going straight at an intersection, turning left or turning right [11], which require taking

different actions depending of the context of the scene.

Most mapless approaches [5, 11, 19, 37, 39], focus on imitating the controls of an expert driver (e.g., steering and acceleration), without providing intermediate interpretable representations that can help explain the self-driving vehicle decisions. Interpretability is of key importance in a safety-critical system particularly if a bad event was to happen. Moreover, the absence of a mechanism to inject structure and prior knowledge makes these methods very brittle to distributional shift [44]. While methods that perform online mapping to obtain lane boundaries or lane center lines have been proposed [2, 16, 18, 21, 37], they either are overly simplistic (e.g., assume lanes are close to parallel to the direction of travel), have only been demonstrated in highway scenarios which are much simpler than city driving, have not been shown to work when coupled with any existing planner, or involve information loss through discrete decisions such as confidence thresholding to output the candidate lanes. The latter is safety-critical as a lane can be completely missed in the worst case, and it makes it difficult to incorporate uncertainty about the static environment in motion planning, which is important to reduce risk.

To address these challenges, we propose an end-to-end approach to mapless driving that is interpretable, does not incur any information loss, and reasons about uncertainty in the intermediate representations. In particular, we propose a set of probabilistic spatial layers to model the static and dynamic parts of the environment. The static environment is subsumed in a planning-centric online map which captures information about which areas are drivable and which ones are reachable given traffic rules. The dynamic actors are captured in a novel occupancy flow that provides occupancy and velocity estimates over time. The motion planning module then leverages these representations without any postprocessing. It utilizes observational data to retrieve dynamically feasible trajectories, predicts a spatial mask over the map to estimate the route given an abstract goal, and leverages the online map and occupancy flow directly as cost functions for explainable, safe plans. We showcase that our approach is significantly safer, more comfortable, and can follow commands better than a wide variety of baselines in challenging closed-loop simulations, as well as when compared to an expert driver in a large-scale real-world dataset.

2. Related Work

We cover previous works on online mapping, perception and prediction, and motion planning, particularly analyzing their fitness to the downstream task of end-to-end driving.

Online Mapping: While there are many offline mapping approaches [4, 22, 33], these rely on satellite imagery or multiple passes through the same scene with a data collection vehicle to gather dense information, and often involve

a human-in-the-loop. For these reasons, such approaches are not suitable for mapless driving. As a consequence, predicting map elements online has recently been proposed. In [16, 18] a network is presented to directly predict the 3D layout of lanes in a traffic scene from a single image. Conversely to the methods above, [2] argues that accurate image estimates do not translate to precise 3D lane boundaries, which are the input required by modern motion planning algorithms. To tackle this, LiDAR and camera are used to predict estimates of ground height and lanes directly in 3D space. Alternatively, [21] proposes a hierarchical recurrent neural network for extraction of structured lane boundaries from LIDAR sweeps. Notably, all the works above are geared toward highway traffic scenes and involve discrete decisions that could be unsafe when driving as they lose valuable uncertainty information. Contrary to these methods, we leverage dense representations of the map that do not involve information loss and are suitable for use in the motion planner as interpretable cost functions.

Perception and Prediction: Most previous works perform object detection [15, 25, 34, 53, 57] and actor-based prediction to reason about the current and future state of a driving scene. As there are multiple possible futures, these methods either generate a fixed set of trajectories [6, 8–10, 26, 28, 30, 36, 56], draw samples to characterize the distribution [7, 41, 47] or predict temporal occupancy maps [23, 27, 43]. However, these pipelines can be unsafe since the detection stage involves confidence thresholding and non-maximum suppression which can remove unconfident detections of real objects. In robotics, occupancy grids at the scene-level (in contrast to actor-level) have been a popular representation of free space. Different from the methods above, [13, 48] estimate occupancy probability of each grid-cell independently using range sensor data. More recently, [20] directly predicts an occupancy grid to replace object detection, but it does not predict how the scene might evolve in the future. [45] improves over such representation by adding semantics as well as future predictions. However, there is no way to extract velocity from the scene occupancy, which is important for motion planning. While [51] considers a dense motion field, their parameterization cannot capture multi-modal behaviors. We follow the philosophy of [13, 20, 45, 48, 51] in predicting scene-level representations, but propose an improved occupancy flow parameterization that can model multi-modal behavior and provides a consistent temporal motion field.

Motion Planning: There is a vast literature on end-to-end approaches for self-driving. The pioneering work of [39] proposes to use a single neural network that directly outputs a driving control command. Subsequent to the success of deep learning, direct control based methods have advanced

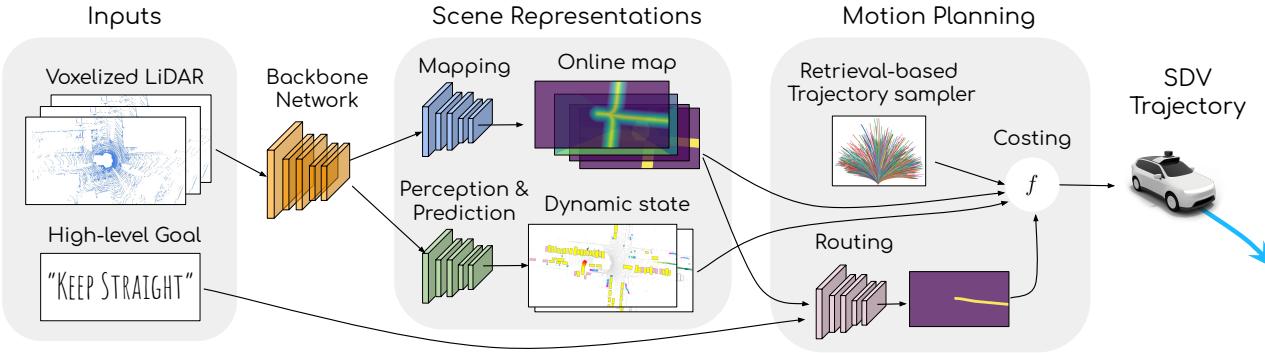


Figure 2: MP3 predicts probabilistic scene representations that are leveraged in motion planning as interpretable cost functions

with deeper networks, richer sensors, and scalable learning methods [5, 11, 24, 35]. Although simple and general, such methods of directly generating control command from sensor data may have stability and robustness issues [12]. More recently, cost map-based approaches have been shown to adapt better to challenging environments, which recover a trajectory by looking for local minima on the cost map. The cost map may be parameterized as a simple linear combination of hand crafted costs [14, 46], or in a general non-parametric form [55]. To bridge the gap between interpretability and expressivity, [45] proposed a model that leverages supervision to learn an interpretable nonparametric occupancy that can be directly used in motion planner, with hand-crafted sub-costs. In contrast to all methods above which rely on an HD map, [31] proposes to output a navigation cost map without localization under a weakly supervised learning environment. This work, however, does not explicitly predict the static and dynamic objects and hence lacks safety and interpretability. Similarly, [3] improves sampling in complex driving environments without the consideration of dynamic objects, and is only demonstrated in simplistic static scenarios. In contrast, our autonomy model leverages retrieval from expert demonstrations to achieve an efficient trajectory sampler that does not rely on the map, predicts a spatial route based on the probabilistic map predictions and a high-level driving command, and stays safe by exploiting an interpretable dynamic occupancy field as a summary of the scene free space and motion.

3. Interpretable Mapless Driving

In this section, we introduce our end-to-end approach to self-driving that operates directly on raw sensor data. Importantly, our model produces intermediate representations that are designed for safe planning, decision-making and interpretability. Our interpretable representations estimate the current and future state of the world around the SDV, including the unknown map as well as the current and future location and velocity of dynamic objects. In the remainder of this section, we first describe our backbone network that extracts meaningful geometric and semantic features from the raw sensor data. We then introduce our intermediate

interpretable representations, and show how they can be exploited to plan maneuvers that are safe, comfortable, and explainable. An overview of our model can be seen in Fig. 2

3.1. Extracting Geometric and Semantic Features

Our model exploits a history of LiDAR point clouds to extract rich geometric and semantic features from the scene over time. Following [30], we voxelize $T_p=10$ past LiDAR point clouds in bird's eye view (BEV), equivalent to 1 second of history, with a spatial resolution of $a = 0.2$ meters/voxel. We exploit odometry to compensate for the SDV motion, thus voxelizing all point clouds in a common coordinate frame. Our region of interest is $W=140\text{m}$ long (70m front and behind of the SDV), $H=80\text{m}$ wide (40 to each side of the SDV), and $Z=5\text{m}$ tall. Following [9], we concatenate height and time along the channel dimension to avoid using 3D convolutions or a recurrent model, thus saving memory and computation. The result is a 3D tensor of size $(\frac{H}{a}, \frac{W}{a}, \frac{Z}{a} \cdot T_p)$, which is the input to our backbone network. This network combines ideas from [9, 53] to extract geometric, semantic and motion information about the scene. More details can be found in the appendix.

3.2. Interpretable Scene Representations

Human drivers are able to successfully navigate complex road topologies with high-density of traffic by exploiting their prior knowledge about traffic rules and social behavior such as the fact that vehicles should drive on the road, close to a lane centerline, in the direction of traffic and should not collide with other actors. Since we would like to incorporate such prior knowledge into the decisions of the SDV, and these to be explainable through interpretable concepts, it is important to predict intelligible representations of the static environment, which we refer here as an *online map*, as well as the dynamic objects position and velocity into the future, captured in our *dynamic occupancy field*. We refer the reader to Fig.3 for an example of these representations. Since the predicted online map and dynamic occupancy field are not going to be perfect due to limitations in the sensors, occlusions and the model, it is important to reason about uncertainty to assess the risk of each possible decision the

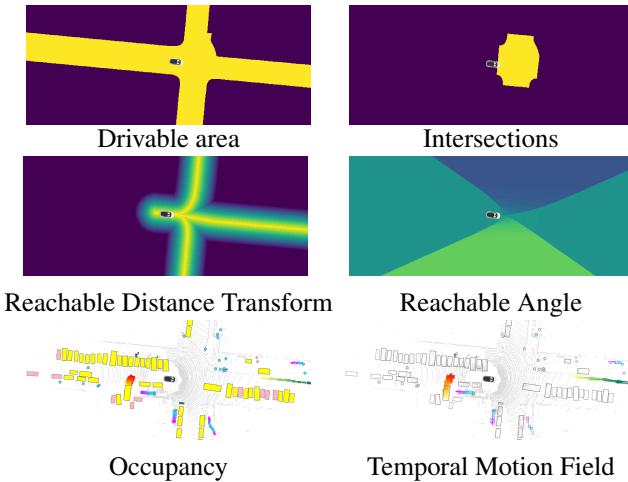


Figure 3: Interpretable Scene representations. For occupancy and motion, we visualize all time steps and classes in the same image to save space, differentiating with colors.

SDV might take. Next, we first describe the semantics in our interpretable representation of the world, and then introduce our probabilistic model.

Online map representation: In order to drive safely it is useful to reason the following elements in BEV:

- *Drivable area*: Road surface (or pavement) where vehicles are allowed to drive, bounded by the curb.
- *Reachable lanes*: Lane center lines (or motion paths) are defined as the canonical paths vehicles travel on, typically in the middle of 2 lane markers. We define the reachable lanes as the subset of motion paths the SDV can get to without breaking any traffic rules. When planning a trajectory, we would like the SDV to stay close to these reachable lanes and drive aligned to their direction. Thus, for each pixel in the ground plane we predict the unsigned distance to the closest reachable lane centerline, truncated at 10 meters, as well as the angle of the closest reachable lane centerline segment.
- *Intersection*: Drivable area portion where traffic is controlled via traffic lights or traffic signs. Reasoning about this is important to handle stop/yield signs and traffic lights. For instance, if a traffic light is red, we should wait to enter the intersection. Following [42], we assume a separate camera-based perception system detects the traffic lights and recognizes their state as this is not our focus.

Dynamic occupancy field: Another critical aspect to achieve safe self-driving is to understand which space is occupied by dynamic objects and how do these move over time. Many accurate LiDAR-based object detectors have been proposed [25, 34, 53, 57] to localize dynamic obstacles followed by a motion forecasting stage [7, 10, 30, 41, 47] to predict the future state of each object. However, all these

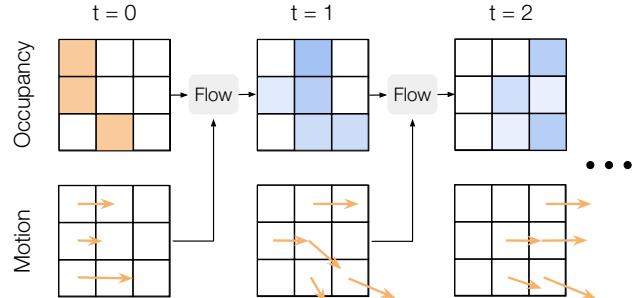


Figure 4: The motion field warps the occupancy over time. Transparency denotes probability. Color differences the predicted layers by the network and the future occupancy. We depict the particular case of unimodal motion ($K = 1$).

methods contain unsafe discrete decisions such as confidence thresholding and non-maximum suppression (NMS) that can eliminate low-confidence predictions of true objects resulting in unsafe situations. [45] proposed a probabilistic way to measure the likelihood of a collision for a given SDV maneuver by exploiting a non-parametric spatial representation of the world. This computation is agnostic to the number of objects. However, this representation does not provide velocity estimates, and thus it is not amenable to car-following behaviors and speed-dependent safety buffer reasoning. Moreover, the decision making algorithm cannot properly reason about interactions, since for a given future occupancy its origin cannot be traced back.

In contrast, in this paper we propose an occupancy flow parameterized by the occupancy of the dynamic objects at the current state of the world and a temporal motion field into the future that describes how objects move (and in turn their future occupancies), both discretized into a spatial grid on BEV with a resolution of 0.4 m/pixel, as depicted in Fig. 4:

- *Initial Occupancy*: a BEV grid cell is active (occupied) if its center falls in the interior of a polygon given by an object shape and its current pose.
- *Temporal Motion Field*: defined for the occupied pixels at a particular time into the future. Each occupied pixel motion is represented with a 2D BEV velocity vector (in m/s). We discretize this motion field into $T = 11$ time steps into the future (up to 5s, every 0.5s).

Since the SDV behavior should be adaptive to objects from different categories (e.g., extra caution is desired around vulnerable road users such as pedestrians and bicyclists), we consider vehicles, pedestrians and bikes as separate classes, each with their own occupancy flow.

Probabilistic Model: We would like to reason about uncertainty in our online map and dynamic occupancy field. Towards this goal, we model each semantic channel of the online map \mathcal{M} as a collection of independent variables per BEV grid cell. This assumption makes the model very simple and efficient. To simplify the notation, we use the letter i

to indicate a spatial index on the grid instead of two indices (row, column) from now on. We model each BEV grid cell in the drivable area and intersections channels as Bernoulli random variables, \mathcal{M}_i^A and \mathcal{M}_i^I respectively, as we consider a grid cell is either part these elements or not. We model the truncated distance transform to the reachable lanes centerline \mathcal{M}_i^D as a Laplacian, which we empirically found to yield more accurate results than a Gaussian, and the direction of the closest lane centerline in the reachable lanes \mathcal{M}_i^θ as a Von Mises distribution since it has support between $[\pi, \pi]$.

We model the occupancy of dynamic objects \mathcal{O}^c for each class $c \in \{\text{vehicle, pedestrian, bicyclist}\}$ as a collection of Bernoulli random variables $\mathcal{O}_{t,i}^c$, one for each spatio-temporal index t, i . Since an agent future behavior is highly uncertain and multi-modal (e.g., a vehicle going straight vs. turning right), we model the motion for each class at each spatio-temporal location as a categorical distribution $\mathcal{K}_{t,i}^c$ over K BEV motion vectors $\{\mathcal{V}_{t,i,k}^c : k \in 1 \dots K\}$. Here, each motion vector is parameterized by the continuous velocity in the x and y directions in BEV. To compute the probability of future occupancy under our probabilistic model, we first define the probability of occupancy flowing from location i_1 to location i_2 between two consecutive time steps t and $t + 1$ as follows:

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

where $p(\mathcal{V}_{t,i_1,k}^c = i_2)$ distributes the mass locally and is determined via bilinear interpolation if i_2 is among the 4 nearest grid cells to the head of the continuous motion vector, and 0 for all other cells, as depicted in Fig. 4. With this definition, we can easily calculate the future occupancy iteratively, starting from the occupancy predictions at $t = 0$. This parameterization ensures consistency by definition between future motion and future occupancy, and provides an efficient way to query how does some particular initial occupancy evolve over time, which will be used for interaction and right-of-way reasoning in our motion planner. 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

$$p(\mathcal{O}_{t+1,i}^c) = 1 - \prod_j (1 - p(\mathcal{F}_{(t,j) \rightarrow (t+1,i)}^c))$$

We point the reader to the appendix for further details on the mapping and perception and prediction network architecture.

3.3. Motion Planning

The goal of the motion planner is to generate trajectories that are safe, comfortable and progressing towards the goal. We design a sample-based motion-planner in which a set of kinematically-feasible trajectories are generated and then evaluated using a learned scoring function. The scoring function utilizes the probabilistic dynamic occupancy field to encode the safety of the possible maneuvers encouraging cautious behaviors that avoid occupied regions, and maintain

a safe headway to the occupied area in front of the SDV. The probabilistic layers in our online map are used in the scoring function to ensure the SDV is driving on the drivable area, close to the lane center and in the right direction, being cautious in uncertain regions, and driving towards the goal specified by the input high-level command. The planner evaluates all the sampled trajectories in parallel and selects the trajectory with the minimum cost:

$$\tau^* = \underset{\tau \in \mathcal{T}(\mathbf{x}_0)}{\operatorname{argmin}} f(\tau, \mathcal{M}, \mathcal{O}, \mathcal{K}, \mathcal{V}; \mathbf{w})$$

with f the scoring function, \mathbf{w} the learnable parameters of our models, \mathcal{M} the map layers, $\mathcal{O}, \mathcal{K}, \mathcal{V}$ the occupancy and motion mode-probability and vector layers respectively, and $\mathcal{T}(\mathbf{x}_0)$ represents the possible trajectories which are generated conditioned on the current state of the SDV \mathbf{x}_0 .

3.3.1 Trajectory Sampling

The lane centers and topology are strong priors to construct the potential trajectories to be executed by the SDV. When an HD map is available, the lane geometry can be exploited to guide the trajectory sampling process. A popular approach, for example, is to sample trajectories in Frenet-frame of the goal lane-center, limiting the samples to motions that do not deviate much from the desired lane [1, 45, 46, 50]. However, in mapless driving we need to take a different approach as the HD map is not available. We thus use retrieval from a large-scale dataset of real trajectories. This approach provides a large set of trajectories from expert demonstrations while avoiding random sampling or arbitrary choices of acceleration/steering profiles [36, 55]. We create a dataset of expert demonstrations by binning based on the SDV initial state, clustering the trajectories of each bin, and using the cluster prototypes for efficiency. During online motion planning, we retrieve the trajectories of the bin specified by $(v_{\mathbf{x}}, a_{\mathbf{x}}, \kappa_{\mathbf{x}})$ with \mathbf{x} the current state of the SDV. However, the retrieved trajectories may have marginally different initial velocity and steering angle than the SDV. Hence, instead of directly using those trajectories, we use the acceleration and steering rate profiles, $(a, \kappa)_t, t = 0, \dots, T$, to rollout a bicycle model [38] starting from the initial SDV state. This process generates trajectories with continuous velocity and steering. This is in contrast to the simplistic approach of, e.g., [37] where a fixed set of template trajectories is used, ignoring the initial state of SDV.

3.3.2 Route Prediction

When HD maps are available, the input route is typically given in the form of a sequence of lanes that the SDV should follow. In mapless driving however, this is not possible. Instead, we assume we are given a driving command as a tuple $c = (a, d)$, where $a \in \{\text{keep lane, turn left, turn right}\}$ is a discrete high-level action, and d an approximate longitudinal distance to the action. This information is similar to

what an off-the-shelf GPS-based navigation system provides to human drivers. To simulate GPS errors, we randomly sample noise from a zero-mean Gaussian with 5m standard deviation. We model the route as a collection of Bernoulli random variables, one for each grid cell in BEV. Given the driving command c and the predicted map \mathcal{M} , a routing network predicts a dense probability map \mathcal{R} in BEV. The routing network is composed of 3 CNNs that act like a switch for the possible high-level actions a . Note that only the one corresponding to the given driving command will be run at inference. Together with the predicted map layers, we “rasterize” the longitudinal distance d to the action as an additional channel (i.e., repeated spatially), and leverage CoordConv [29] to break the translation invariance of CNNs.

3.3.3 Trajectory Scoring

We use a linear combination of the following cost functions to score the sampled trajectories. More detailed explanations about the individual costs can be found in the appendix.

Routing and Driving on Roads: In order to encourage the SDV to perform the high-level command, we use a scoring function that encourages trajectories that travel a larger distance in regions with high probability in \mathcal{R} . We use the following score function:

$$f_r(\tau, \mathcal{R}) = -m(\tau) \min_{i \in m(\tau)} \mathcal{R}_i$$

where $m(\tau)$ is the BEV grid-cells that overlap with SDV polygon in trajectory τ . This score function makes sure the SDV stays on the route and is only rewarded when moving within the route. We introduce an additional cost-to-go that considers the predicted route beyond the planning horizon. This is important when there is a turn at the end of the horizon and the SDV velocity is high. Specifically, we compute the average value of $1 - \mathcal{R}_j$ for all BEV grid-cells j that have overlap with SDV beyond the trajectory horizon, assuming that the SDV maintains constant velocity and heading.

The SDV needs to always stay close to the center of the reachable lanes while on the road. Hence we use the predicted reachable lanes distance transform \mathcal{M}^D to penalize distant trajectory points. In order to promote cautious behavior when there is high uncertainty in \mathcal{M}^D and \mathcal{M}^θ , 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}^θ . This promotes slow maneuver in the presence of map uncertainty.

The SDV is also required to stay on the road and avoid encroaching onto the side-walks or the curb. Hence, we use the predicted drivable area \mathcal{M}^A to penalize trajectories that go off the road:

$$f_a(\mathbf{x}, \mathcal{M}) = \max_{i \in m(\mathbf{x})} [1 - P(\mathcal{M}_i^A)]$$

where $m(\mathbf{x})$ is the set of BEV grid-cells that overlap with SDV at trajectory point \mathbf{x} . Similarly, the SDV needs to avoid junctions with red-traffic lights. Hence, we use the predicted junction probability map \mathcal{M}^J to penalize maneuvers that violate red-traffic light, similar to the routing cost.

Safety: The predicted occupancy layers and motion predictions are used to score the trajectory samples with respect to safety. We penalize trajectories where the SDV overlaps occupied regions. For each trajectory point, we use the BEV grid-cell with maximum probability among all the grid-cells that overlap with SDV polygon and use this probability directly as collision cost. The max operator ensures that the worst-case occupancy is considered over the region SDV occupies.

The above objective promotes trajectories that do not overlap with occupied regions. However, the SDV needs to also maintain a safe distance from objects that are in the direction of SDV motion. This headway distance is a function of the relative speed of the SDV wrt the other objects. To compute this cost for each trajectory point \mathbf{x} , we retrieve all the BEV grid-cells in front of the SDV at \mathbf{x} and measure the violation of safety distance if the object at each of those grid-cells stops with hard deceleration, and SDV with state \mathbf{x}_t reacts with a comfortable deceleration.

Comfort: We also penalize jerk, lateral acceleration, curvature and its rate of change to promote comfortable driving.

3.4. Learning

We optimize our driving model in two stages. We first train the online map, dynamic occupancy field, and routing. Once these are converged, in a second stage, we keep these parts frozen and train the planner weights for the linear combination of scoring functions. We found this 2-stage training empirically more stable than training end-to-end.

Online map: We train the online map using negative log-likelihood (NLL) under the data distribution. That means, Gaussian NLL for reachable lanes distance transform \mathcal{M}^D , Von Mises NLL for direction of traffic \mathcal{M}^θ and binary cross-entropy for drivable area \mathcal{M}^A and junctions \mathcal{M}^J .

Dynamic occupancy field: To learn the occupancy \mathcal{O} of dynamic objects at the current and future time stamps, we employ cross entropy loss with hard negative mining to tackle the high imbalance in the data (i.e., the majority of the space is free). To learn the probabilistic motion field, the motion modes \mathcal{K} are learned in an unsupervised fashion via a categorical cross-entropy, where the true mode is defined as the one which associated motion vector is closest to the ground-truth motion in ℓ_2 distance. Then, only the associated motion vector from the true mode is trained via a Huber loss.

Model	Success (%)↑	OffRoute (%)↓	L2 (m)↓	any event	Progress per event (m) ↑				Comfort	
					collision	off-road	off-route	oncoming	jerk($\frac{m}{s^3}$) ↓	lat.acc. ($\frac{m}{s^2}$) ↓
IL	0.00	99.39	39.10	15.69	44.49	36.40	30.28	65.18	98.99	0.91
CIL	0.00	99.39	35.53	15.85	38.50	34.68	35.64	54.58	52.88	0.81
TC	12.80	67.07	30.35	51.17	127.87	288.07	105.26	329.90	3.15	0.25
NMP	22.56	64.02	27.95	69.83	331.81	721.74	104.70	1229.82	3.04	0.14
CNMP	21.34	47.56	27.45	74.85	158.85	646.49	198.28	543.32	2.96	0.26
MP3	74.39	14.63	12.95	218.40	1037.08	1136.49	409.34	1465.27	1.64	0.10

Table 1: Closed-loop simulation results

Model	Collisions (%)		L2 (m)		Progress(m) 0-5s	OffRoute(%) 0-5s	OffRoad(%) 0-5s	Oncoming(%) 0-5s	lat.acc.($\frac{m}{s^2}$) 0-5s	Jerk ($\frac{m}{s^3}$) 0-5s
	0-3s	0-5s	@3s	@5s						
IL	2.17	9.54	1.36	3.77	23.62	5.05	4.46	3.05	1.00	2.47
CIL	2.20	10.15	1.38	3.79	23.58	5.16	5.28	3.64	1.10	2.60
TC	1.72	6.95	2.02	4.34	22.26	2.68	0.28	0.62	1.47	7.48
NMP	0.83	5.18	1.75	4.47	23.09	1.59	0.00	0.21	1.14	3.98
CNMP	1.03	5.45	1.62	4.02	22.99	0.14	0.07	0.14	1.28	3.97
MP3	0.21	2.07	1.71	4.54	25.15	0.15	0.42	0.09	1.23	1.88

Table 2: Large-scale evaluation against expert demonstrations

Note that because the occupancy at future time steps $t > 1$ is obtained by warping the initial occupancy iteratively with the motion field, the whole motion field receives supervision from the occupancy loss. This is important in practice.

Routing: We train the route prediction with binary cross-entropy loss. To learn a better routing model, we leverage supervision for all possible commands given a scene, instead of just the command that the SDV followed in the observational data. This does not require additional human annotations, since we can extract all possible (command, route) pairs from the ground-truth HD map.

Scoring: Since selecting the minimum-cost trajectory within a discrete set is non-differentiable, we use the max-margin loss [40, 45] to penalize trajectories that have small cost but differ from the human demonstration or are unsafe.

4. Experimental Evaluation

In this section we first describe our experimental setup, and then present quantitative results in both closed-loop and open-loop. Closed-loop evaluations are of critical importance since as the execution unrolls, the SDV finds itself in states induced by its own previous motion plans, and thus it is much more challenging than open-loop and closer to the real task of driving. We defer the ablations of several components from our model to the appendix.

Dataset: We train our models using our large-scale dataset URBANEXPERT that includes challenging scenarios where the operators are instructed to drive smoothly and in a safe manner. It contains 5000 scenarios for training, 500 for

validation and 1000 for the test set. Each scenario is 25 seconds. Compared to KITTI [17], URBANEXPERT has 33x more hours of driving. Note that the train/validation/test splits are geographically non-overlapping which is crucial to evaluate generalization.

Baselines: We compare against many SOTA approaches. *Imitation Learning (IL)*, where the future positions of the SDV are predicted directly from the scene context features, and is trained using L2 loss. *Conditional Imitation Learning (CIL)* [11], which is similar to *IL* but the trajectory is conditioned on the driving command. *Neural Motion Planner (NMP)* [55], where a planning cost-volume as well as detection and prediction are predicted in a multi-task fashion from the scene context features, and *Trajectory Classification (TC)* [37], where a cost-volume is predicted similar to NMP, but the trajectory cost is used to create a probability distribution over the trajectories and is trained by optimizing for the likelihood of the expert trajectory. Finally, we extend NMP to consider the high-level command by learning a separate costing network for each discrete action (*CNMP*).

Closed-loop Simulation Results: Our simulated environment leverages a state-of-the-art LiDAR simulator [32] to recreate a virtual world from previously collected real static environments and a large-scale bank of diverse actors. We use a set of 164 curated scenarios (18 seconds each) that are particularly challenging and require complex decision making and motion planning. The simulation starts by replaying the motion of the actors as happened during the real-world capture. In case the scenario diverges from the original one due to SDV actions (e.g., SDV moving slower), the affected

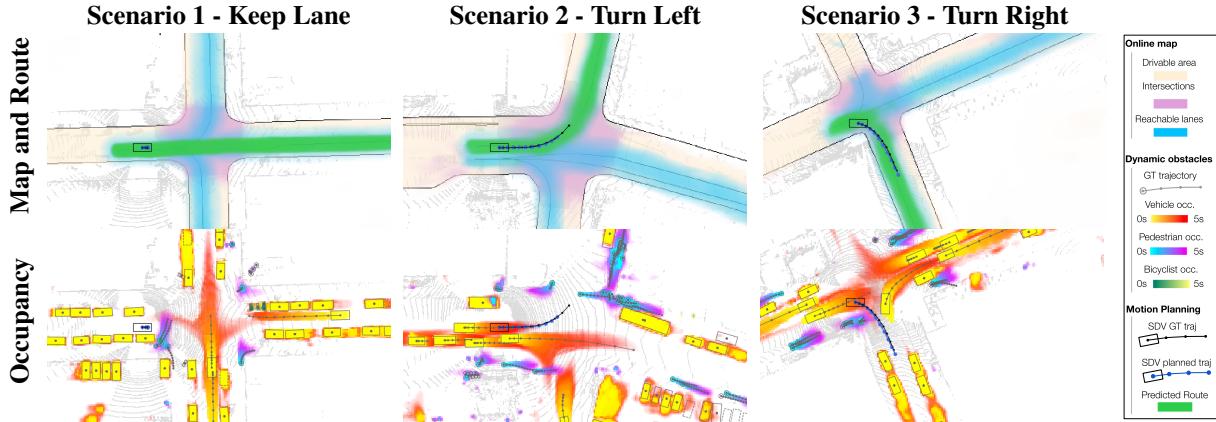


Figure 5: **Qualitative results.** We show our predicted scene representations and motion plan for different high-level actions.

actors (e.g., rear vehicles) switch to the *Intelligent Driver Model* [49] for the rest of the simulation in order to be reactive. We stop the simulation if the execution diverges too far from the commanded route. A scenario is a success iff there are no events, i.e., the SDV does not collide with other actors, follows the route, does not get out of the road nor into opposite traffic. We report the *Success rate*. Because the goal of an SDV is to reach a goal by following the driving commands, we report *Off-route (%)*, which measures the percentage of scenarios the SDV goes outside the route. Since all simulated scenarios are initialized from a real log, we measure the average *L2* distance to the trajectory demonstrated by the expert driver. Progress is measured by recording the meters traveled until an event happens. We summarize this in the metric *meters per event*, and show a breakdown per event category. Table 1 shows that our method clearly outperforms all the baselines across all metrics. MP3 achieves over 3x the success rate, diverges from the route a third of the times, imitates the human expert driver at least twice as close, and progresses 3x more per event than any baseline, while also being the most comfortable.

Open-Loop Evaluation: We evaluate our method against human expert demonstrations on URBANEXPERT. We measure the safety of the planner via the *% of collisions* with the ground-truth actors up to each trajectory time step. *Progress* measures how far the SDV advanced along the route for the 5s planning horizon, and *L2* the distance to the human expert trajectory at different time steps. To illustrate the map and route understanding, we compute the *road violation rate*, *oncoming traffic violation rate*, and *route violation rate*. Finally, *jerk* and *lateral acceleration* show how comfortable the produced trajectories are. As shown in Table 2 our MP3 model produces the safest trajectories that in turn achieve the most progress and are the most comfortable. In terms of imitation, IL and CIL outperform the rest since they are optimized for this metric, but are very unsafe. Our model achieves similar map-related metrics than the best performing baselines (NMP/CNMP) in open-loop. We want to stress

the fact that these experiments are open-loop, and thus the SDV always plans from an expert state. Because of this, it is very unusual to diverge from the route/road. We consider this a secondary evaluation that does not reflect very well the actual performance when executing these plans, but include it for completeness since previous methods [37, 55] benchmark this way. Comparing these results to closed-loop, we can see that MP3 is much more robust than the baselines to the distributional shift incurred by the SDV unrolling its own plans over time.

Qualitative Results: Fig. 5 showcases the outputs from our model. *Scenario 1* shows the predictions when our model is commanded to keep straight at the intersection. Our model recognizes and accurately predicts the future motion of pedestrians near the SDV that just came out of occlusion, and plans a safe stop accordingly. Moreover, we can appreciate the high expressivity of our dynamic occupancy field at the bottom, which can capture highly multimodal behaviors such as the 3 modes of the vehicle heading north at the intersection. *Scenario 2* and *Scenario 3* show how our model accurately predicts the route when given turning commands, as well as how planning can progress through crowded scenes similar to the human demonstrations. See Appendix C for visualizations of the retrieved trajectory samples from the motion planner together with their cost, as well as a comparison of closed-loop rollouts against the baselines.

5. Conclusion

In this paper, we have proposed an end-to-end model for mapless driving. Importantly, our method produces probabilistic intermediate representations that are interpretable and ready-to-use as cost functions in our neural motion planner. We showcased that our driving model is safer, more comfortable and progresses the most among SOTA approaches in a large-scale dataset. Most importantly, when we evaluate our model in a closed-loop simulator without any additional training it is far more robust than the baselines, achieving very significant improvements across all metrics.

References

- [1] Zlatan Ajanovic, Bakir Lucevic, Barys Shyrokau, Michael Stolz, and Martin Horn. Search-based optimal motion planning for automated driving. In *IROS*, 2018. 1, 5
- [2] Min Bai, Gellert Mattyus, Namdar Homayounfar, Shenlong Wang, Shrinidhi Kowshika Lakshmikanth, and Raquel Urtasun. Deep multi-sensor lane detection. In *IROS*, pages 3102–3109. IEEE, 2018. 2
- [3] Holger Banzhaf, Paul Sanzenbacher, Ulrich Baumann, and J. Marius Zöllner. Learning to predict ego-vehicle poses for sampling-based nonholonomic motion planning. *IEEE RA-L*, 4(2):1053–1060, 2019. 3
- [4] Favyen Bastani, Songtao He, Sofiane Abbar, Mohammad Alizadeh, Hari Balakrishnan, Sanjay Chawla, Sam Madden, and David DeWitt. Roadtracer: Automatic extraction of road networks from aerial images. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4720–4728, 2018. 2
- [5] Mariusz Bojarski, Davide Del Testa, Daniel Dworakowski, Bernhard Firner, Beat Flepp, Prasoon Goyal, Lawrence D Jackel, Mathew Monfort, Urs Muller, Jiakai Zhang, et al. End to end learning for self-driving cars. *arXiv preprint arXiv:1604.07316*, 2016. 2, 3
- [6] Sergio Casas, Cole Gulino, Renjie Liao, and Raquel Urtasun. Spatially-aware graph neural networks for relational behavior forecasting from sensor data. *arXiv preprint arXiv:1910.08233*, 2019. 2
- [7] Sergio Casas, Cole Gulino, Simon Suo, Katie Luo, Renjie Liao, and Raquel Urtasun. Implicit latent variable model for scene-consistent motion forecasting. *arXiv preprint arXiv:2007.12036*, 2020. 2, 4
- [8] Sergio Casas, Cole Gulino, Simon Suo, and Raquel Urtasun. The importance of prior knowledge in precise multimodal prediction. *arXiv preprint arXiv:2006.02636*, 2020. 2
- [9] Sergio Casas, Wenjie Luo, and Raquel Urtasun. Intentnet: Learning to predict intention from raw sensor data. In *CoRL*, 2018. 2, 3, 11
- [10] Yuning Chai, Benjamin Sapp, Mayank Bansal, and Dragomir Anguelov. Multipath: Multiple probabilistic anchor trajectory hypotheses for behavior prediction. *arXiv preprint arXiv:1910.05449*, 2019. 2, 4
- [11] Felipe Codevilla, Matthias Müller, Antonio López, Vladlen Koltun, and Alexey Dosovitskiy. End-to-end driving via conditional imitation learning. In *ICRA*, 2018. 1, 2, 3, 7
- [12] Felipe Codevilla, Eder Santana, Antonio M López, and Adrien Gaidon. Exploring the limitations of behavior cloning for autonomous driving. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 9329–9338, 2019. 3
- [13] Alberto Elfes. Using occupancy grids for mobile robot perception and navigation. *Computer*, 22(6):46–57, 1989. 2
- [14] Haoyang Fan, Zhongpu Xia, Changchun Liu, Yaqin Chen, and Qi Kong. An auto-tuning framework for autonomous vehicles. *arXiv preprint arXiv:1808.04913*, 2018. 3
- [15] Davi Frossard, Simon Suo, Sergio Casas, James Tu, Rui Hu, and Raquel Urtasun. Strobe: Streaming object detection from lidar packets. *CoRL*, 2020. 2
- [16] Noa Garnett, Rafi Cohen, Tomer Pe'er, Roee Lahav, and Dan Levi. 3d-lanenet: End-to-end 3d multiple lane detection. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2921–2930, 2019. 2
- [17] Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pages 3354–3361. IEEE, 2012. 7
- [18] Yuliang Guo, Guang Chen, Peitao Zhao, Weide Zhang, Jing-hao Miao, Jingao Wang, and Tae Eun Choe. Gen-lanenet: A generalized and scalable approach for 3d lane detection. *arXiv*, pages arXiv–2003, 2020. 2
- [19] Jeffrey Hawke, Richard Shen, Corina Gurau, Siddharth Sharma, Daniele Reda, Nikolay Nikolov, Przemyslaw Mazur, Sean Micklethwaite, Nicolas Griffiths, Amar Shah, et al. Urban driving with conditional imitation learning. *arXiv preprint arXiv:1912.00177*, 2019. 2
- [20] Stefan Hoermann, Martin Bach, and Klaus Dietmayer. Dynamic occupancy grid prediction for urban autonomous driving: A deep learning approach with fully automatic labeling. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2056–2063. IEEE, 2018. 2
- [21] Namdar Homayounfar, Wei-Chiu Ma, Shrinidhi Kowshika Lakshmikanth, and Raquel Urtasun. Hierarchical recurrent attention networks for structured online maps. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3417–3426, 2018. 2
- [22] Namdar Homayounfar, Wei-Chiu Ma, Justin Liang, Xinyu Wu, Jack Fan, and Raquel Urtasun. Dagmapper: Learning to map by discovering lane topology. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2911–2920, 2019. 2
- [23] Ajay Jain, Sergio Casas, Renjie Liao, Yuwen Xiong, Song Feng, Sean Segal, and Raquel Urtasun. Discrete residual flow for probabilistic pedestrian behavior prediction. *arXiv preprint arXiv:1910.08041*, 2019. 2
- [24] Alex Kendall, Jeffrey Hawke, David Janz, Przemyslaw Mazur, Daniele Reda, John-Mark Allen, Vinh-Dieu Lam, Alex Bewley, and Amar Shah. Learning to drive in a day. *arXiv preprint arXiv:1807.00412*, 2018. 3
- [25] Alex H Lang, Sourabh Vora, Holger Caesar, Lubing Zhou, Jiong Yang, and Oscar Beijbom. Pointpillars: Fast encoders for object detection from point clouds. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 12697–12705, 2019. 2, 4
- [26] Lingyun Luke Li, Bin Yang, Ming Liang, Wenyuan Zeng, Mengye Ren, Sean Segal, and Raquel Urtasun. End-to-end contextual perception and prediction with interaction transformer. *arXiv preprint arXiv:2008.05927*, 2020. 2
- [27] Junwei Liang, Lu Jiang, Kevin Murphy, Ting Yu, and Alexander Hauptmann. The garden of forking paths: Towards multi-future trajectory prediction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10508–10518, 2020. 2
- [28] Ming Liang, Bin Yang, Wenyuan Zeng, Yun Chen, Rui Hu, Sergio Casas, and Raquel Urtasun. Pnpnet: End-to-end perception and prediction with tracking in the loop. In *Proceed-*

- ings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020. 2
- [29] 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. 6, 11, 15
- [30] Wenjie Luo, Bin Yang, and Raquel Urtasun. Fast and furious: Real time end-to-end 3d detection, tracking and motion forecasting with a single convolutional net. In *CVPR*, 2018. 2, 3, 4
- [31] Huifang Ma, Yue Wang, Li Tang, Sarath Kodagoda, and Rong Xiong. Towards navigation without precise localization: Weakly supervised learning of goal-directed navigation cost map. *CoRR*, abs/1906.02468, 2019. 3
- [32] Sivabalan Manivasagam, Shenlong Wang, Kelvin Wong, Wenyuan Zeng, Mikita Sazanovich, Shuhan Tan, Bin Yang, Wei-Chiu Ma, and Raquel Urtasun. Lidarsim: Realistic lidar simulation by leveraging the real world. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11167–11176, 2020. 7
- [33] Gellért Mátyus, Wenjie Luo, and Raquel Urtasun. Deep-roadmapper: Extracting road topology from aerial images. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 3438–3446, 2017. 2
- [34] Gregory P Meyer, Ankit Laddha, Eric Kee, Carlos Valdespi-Gonzalez, and Carl K Wellington. Lasernet: An efficient probabilistic 3d object detector for autonomous driving. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 12677–12686. IEEE, 2019. 2, 4
- [35] Matthias Müller, Alexey Dosovitskiy, Bernard Ghanem, and Vladlen Koltun. Driving policy transfer via modularity and abstraction. *arXiv preprint arXiv:1804.09364*, 2018. 3
- [36] Tung Phan-Minh, Elena Corina Grigore, Freddy A Boulton, Oscar Beijbom, and Eric M Wolff. Covernet: Multimodal behavior prediction using trajectory sets. *arXiv preprint arXiv:1911.10298*, 2019. 2, 5
- [37] Jonah Philion and Sanja Fidler. Lift, splat, shoot: Encoding images from arbitrary camera rigs by implicitly unprojecting to 3d. In *Proceedings of the European Conference on Computer Vision*, 2020. 2, 5, 7, 8
- [38] Philip Polack, Florent Altché, Brigitte d’Andréa Novel, and Arnaud de La Fortelle. The kinematic bicycle model: A consistent model for planning feasible trajectories for autonomous vehicles? In *2017 IEEE Intelligent Vehicles Symposium (IV)*, pages 812–818. IEEE, 2017. 5
- [39] Dean A Pomerleau. Alvinn: An autonomous land vehicle in a neural network. In *NIPS*, 1989. 2
- [40] Nathan D Ratliff, J Andrew Bagnell, and Martin A Zinkevich. Maximum margin planning. In *ICML*, 2006. 7, 14
- [41] Nicholas Rhinehart, Kris M Kitani, and Paul Vernaza. R2p2: A reparameterized pushforward policy for diverse, precise generative path forecasting. In *ECCV*, 2018. 2, 4
- [42] Nicholas Rhinehart, Rowan McAllister, and Sergey Levine. Deep imitative models for flexible inference, planning, and control. *arXiv preprint arXiv:1810.06544*, 2018. 4
- [43] Daniela Ridel, Nachiket Deo, Denis Wolf, and Mohan Trivedi. Scene compliant trajectory forecast with agent-centric spatio-temporal grids. *IEEE RA-L*, 5(2):2816–2823, 2020. 2
- [44] Stéphane Ross, Geoffrey Gordon, and Drew Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pages 627–635, 2011. 2
- [45] 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. 1, 2, 3, 4, 5, 7, 14
- [46] Abbas Sadat, Mengye Ren, Andrei Pokrovsky, Yen-Chen Lin, Ersin Yumer, and Raquel Urtasun. Jointly learnable behavior and trajectory planning for self-driving vehicles. In *IROS*, pages 3949–3956. IEEE, 2019. 1, 3, 5
- [47] Charlie Tang and Russ R Salakhutdinov. Multiple futures prediction. In *Advances in Neural Information Processing Systems*, pages 15398–15408, 2019. 2, 4
- [48] Sebastian Thrun. Learning occupancy grid maps with forward sensor models. *Autonomous robots*, 15(2):111–127, 2003. 2
- [49] Martin Treiber, Ansgar Hennecke, and Dirk Helbing. Congested traffic states in empirical observations and microscopic simulations. *Physical Review E*, 62(2), Aug 2000. 8
- [50] Moritz Werling, Julius Ziegler, Sören Kammel, and Sebastian Thrun. Optimal trajectory generation for dynamic street scenarios in a frenet frame. In *ICRA*, 2010. 1, 5
- [51] 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. 2, 14
- [52] Yuxin Wu and Kaiming He. Group normalization. In *Proceedings of the European conference on computer vision (ECCV)*, pages 3–19, 2018. 11
- [53] Bin Yang, Wenjie Luo, and Raquel Urtasun. Pixor: Real-time 3d object detection from point clouds. In *CVPR*, 2018. 2, 3, 4, 11
- [54] Fisher Yu and Vladlen Koltun. Multi-scale context aggregation by dilated convolutions. *arXiv preprint arXiv:1511.07122*, 2015. 11
- [55] Wenyuan Zeng, Wenjie Luo, Simon Suo, Abbas Sadat, Bin Yang, Sergio Casas, and Raquel Urtasun. End-to-end interpretable neural motion planner. In *CVPR*, 2019. 3, 5, 7, 8
- [56] Tianyang Zhao, Yifei Xu, Mathew Monfort, Wongun Choi, Chris Baker, Yibiao Zhao, Yizhou Wang, and Ying Nian Wu. Multi-agent tensor fusion for contextual trajectory prediction. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 12126–12134, 2019. 2
- [57] Yin Zhou, Pei Sun, Yu Zhang, Dragomir Anguelov, Jiyang Gao, Tom Ouyang, James Guo, Jiquan Ngiam, and Vijay Vasudevan. End-to-end multi-view fusion for 3d object detection in lidar point clouds. *arXiv preprint arXiv:1910.06528*, 2019. 2, 4

Appendix

In this appendix 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.

A. Implementation details

A.1. Architecture

Backbone network: Fig. 6 shows an architecture diagram of the backbone network. We combine ideas from [9, 53] 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. 6. All convolutional layers use Group Normalization [52] 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. 7. It takes as input multiple feature maps $\mathcal{C}_{1x}, \mathcal{C}_{2x}, \mathcal{C}$ from the backbone network (see Fig. 6), 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 [54] 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 A.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 9. 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 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) [29] in order to be able to reason about the distance to a particular grid cell from the SDV.

A.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:

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

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

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

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

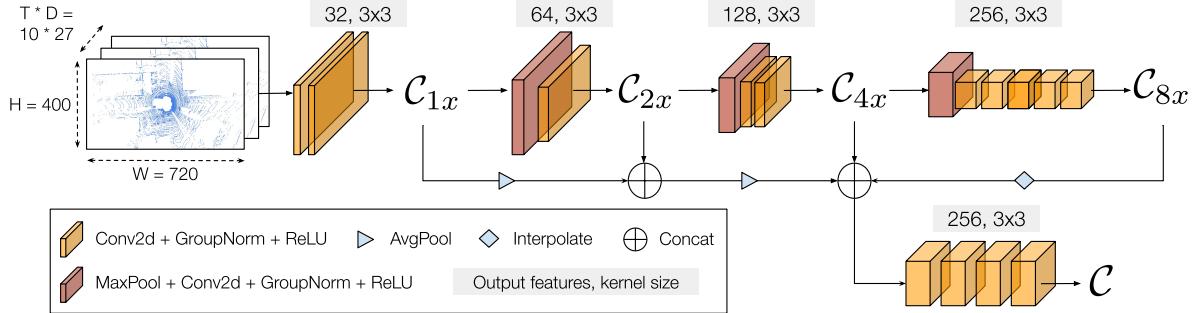


Figure 6: **Backbone Network.** The output features within one CNN block are fixed. All kernel strides and dilation are 1.

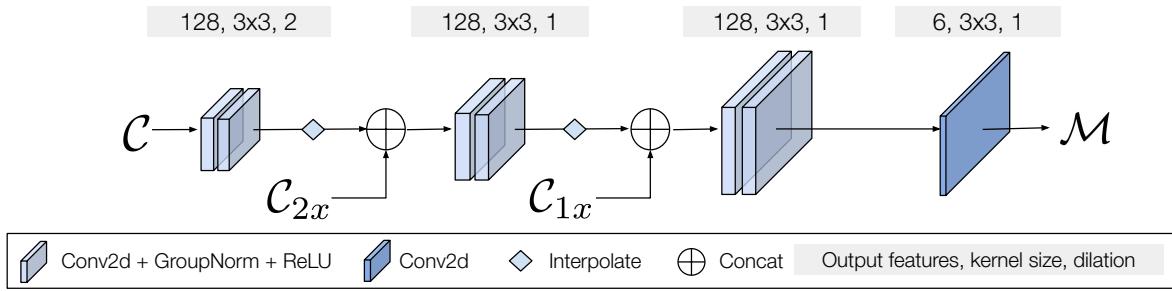


Figure 7: **Mapping Network.** The output features within one CNN block are fixed. All kernel strides are 1.

probability that no occupancy flow event occurs, and take its complement

$$\begin{aligned} p(\mathcal{O}_{t+1,i}^c) &= p\left(\bigcup_j \mathcal{F}_{(t,j) \rightarrow (t+1,i)} : \forall k, l\right) \\ &= 1 - p\left(\bigcap_j \mathcal{F}'_{(t,j) \rightarrow (t+1,i)} : \forall k, l\right) \\ &= 1 - \prod_j (1 - p(\mathcal{F}_{(t,j) \rightarrow (t+1,i)}^c)) \end{aligned}$$

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

A.3. Planning

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

A.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 10 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

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

A.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}^θ indexed at all BEV grid-cells that have overlap with SDV polygon.

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

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}^θ , 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}^θ . This promotes slow

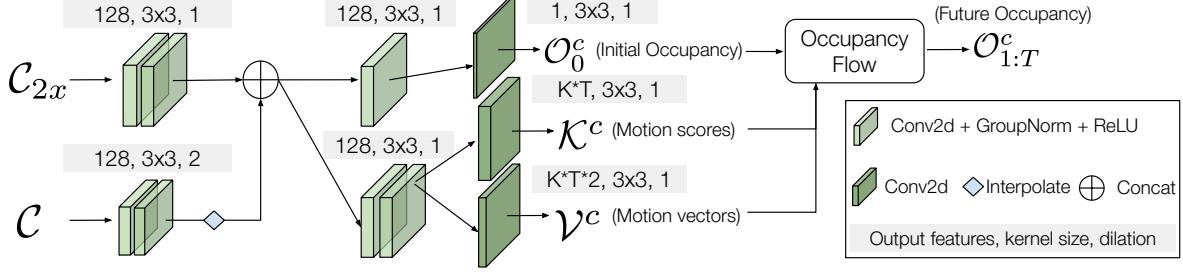


Figure 8: **Perception and Prediction Network.** It outputs the initial occupancy and current and future motion. The output features within one CNN block are fixed. All kernel strides are 1.

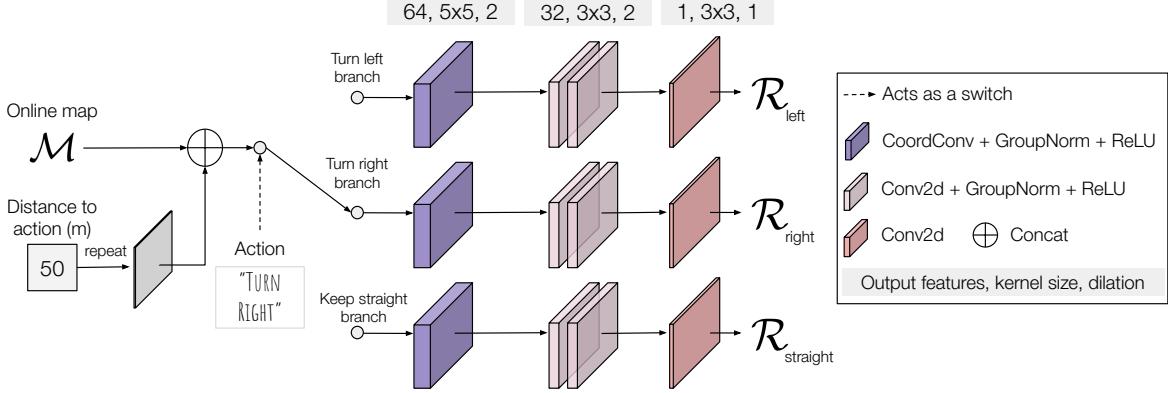


Figure 9: **Routing Network.** The output features within one CNN block are fixed. All kernel strides are 1. The architecture is the same for the 3 branches, but with different learnable parameters.

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 SDV at time t . The headway cost is then computed over this

set as follows:

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

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.

A.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).

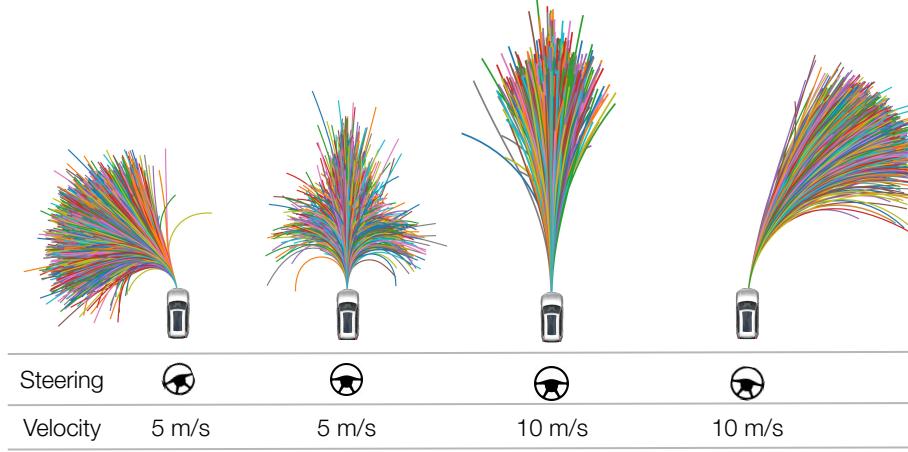


Figure 10: Sets of trajectories retrieved from the expert demonstrations. The initial state of the SDV is used to query trajectories that start with similar state (the initial acceleration is $0.0 \frac{m}{s^2}$ in all the shown cases).

Loss	Success (%)↑	OffRoute (%)↓	L2 (m)↓	any event	Progress per event (m) ↑				Comfort	
	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 3: **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.

Trajectory Scoring (2nd stage): Since selecting the minimum-cost trajectory within a discrete set is non-differentiable, we use the max-margin loss [40, 45] 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.

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

where f_o^t is the occupancy cost function at time step t , f_r are the rest of the planning subcosts, and $[\cdot]_+$ 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.

B. Additional Experiments

B.1. 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 su-

pervising 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 [51]. 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 8 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.

B.2. 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

Routing input	Success (%)↑	OffRoute (%)↓	L2 (m)↓	any event	Progress per event (m)↑			Comfort		
					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 4: **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)↓	any event	Progress per event (m)↑			Comfort		
					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 5: **Upper bound performance** in closed-loop simulation when using an HD map to feed the motion planner the true online map and route.

distance to action as explained in Fig. 9. Our results in Table 4 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 [29] 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.

B.3. Upper bound with access to HD map

Table 5 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 (i.e. the second stage in the training described in Sec. 3.4). 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 con-

firms 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.

C. Qualitative results

C.1. MP3 outputs in closed-loop simulation

Figures 11, 12 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. 13, 14, 15, 16 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.

C.2. Plan comparison against baselines in closed-loop simulation

Figures 17, 18, 19 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).

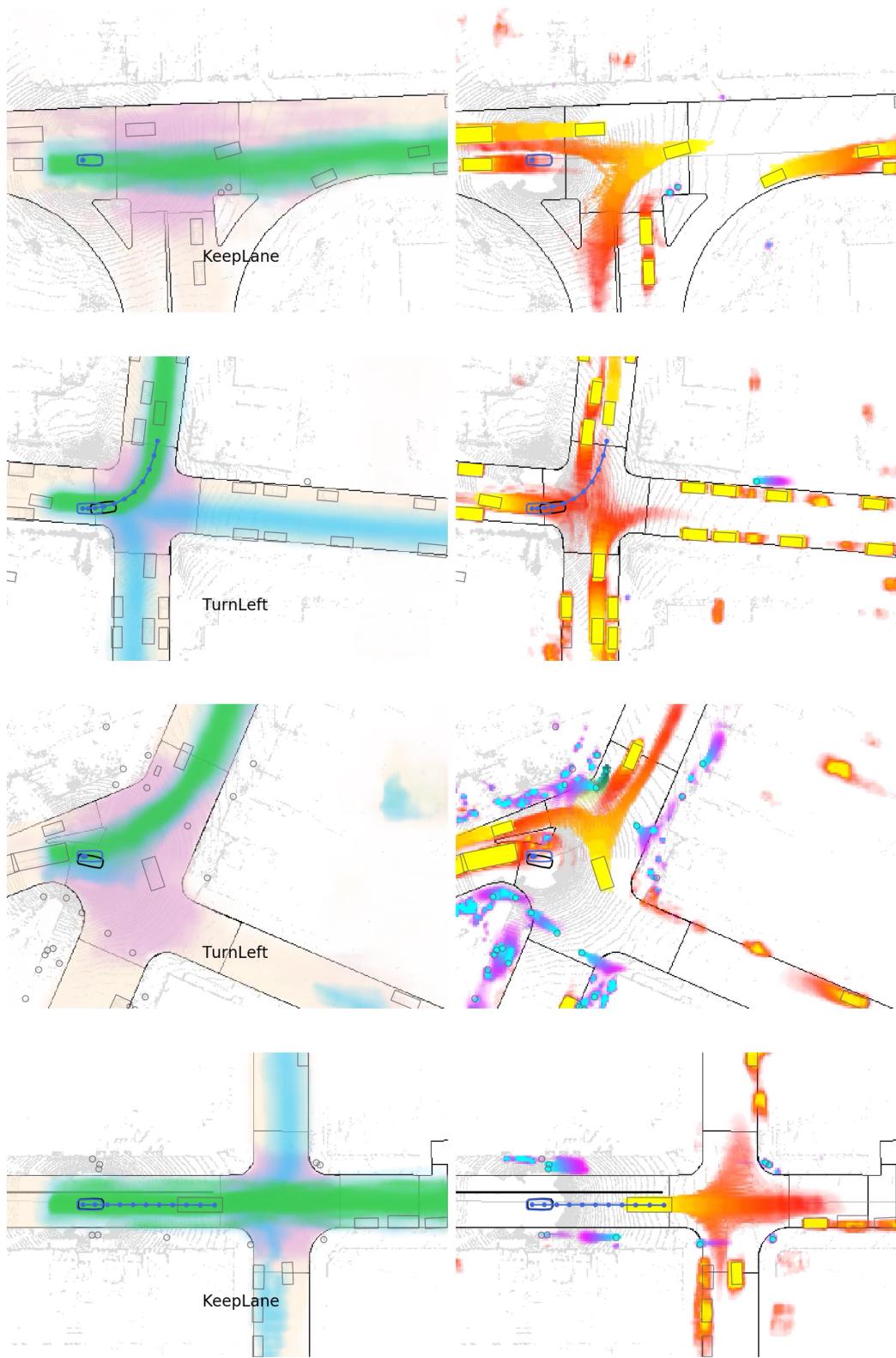


Figure 11: Qualitative results of MP3 in closed-loop. We show our predicted scene representations and motion plan in different interesting scenarios. The black bounding box represents the state of the expert driver at that point, and we can see it's very close to the planner state (in blue)

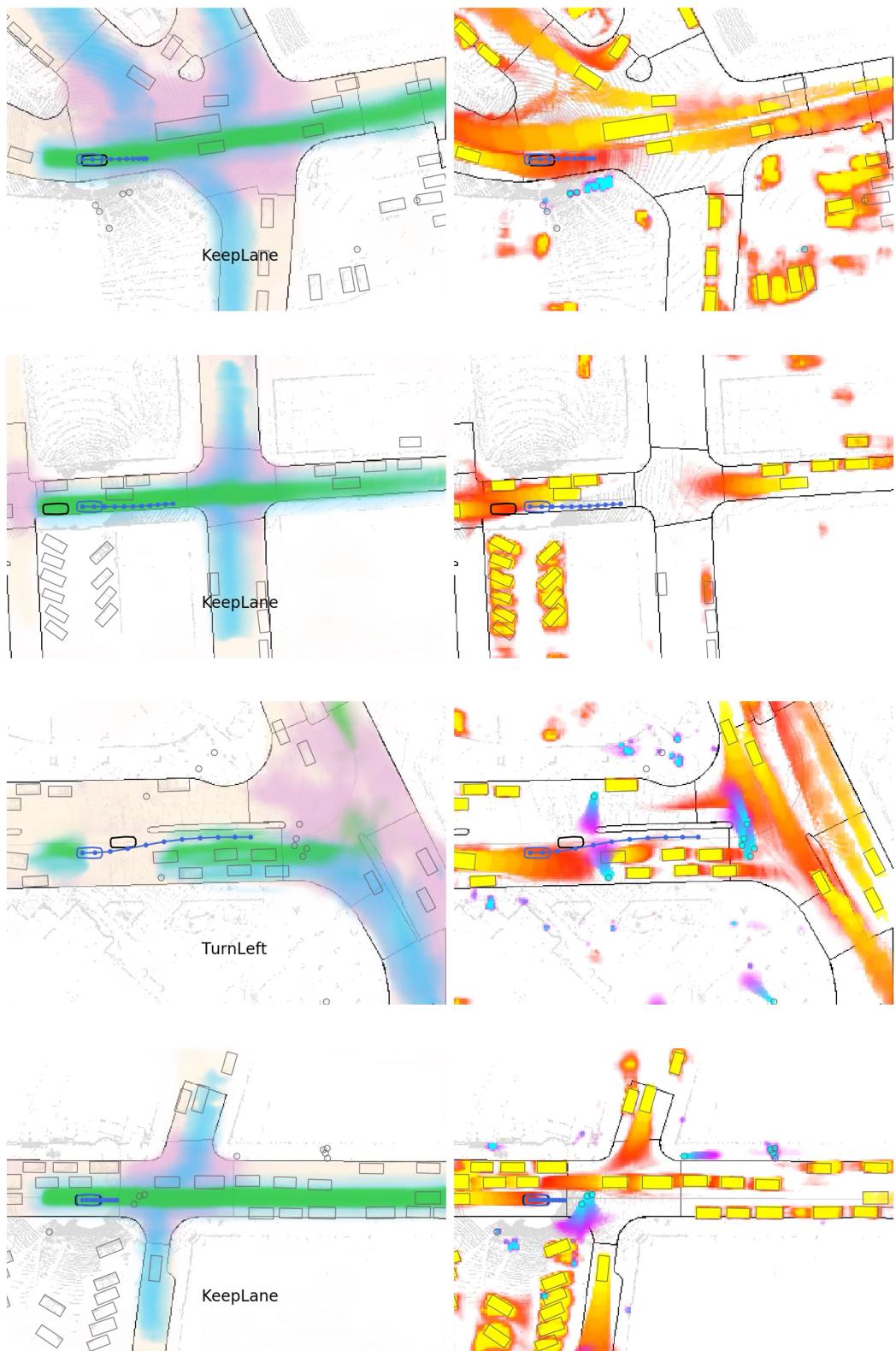


Figure 12: More qualitative results of MP3 in closed-loop.

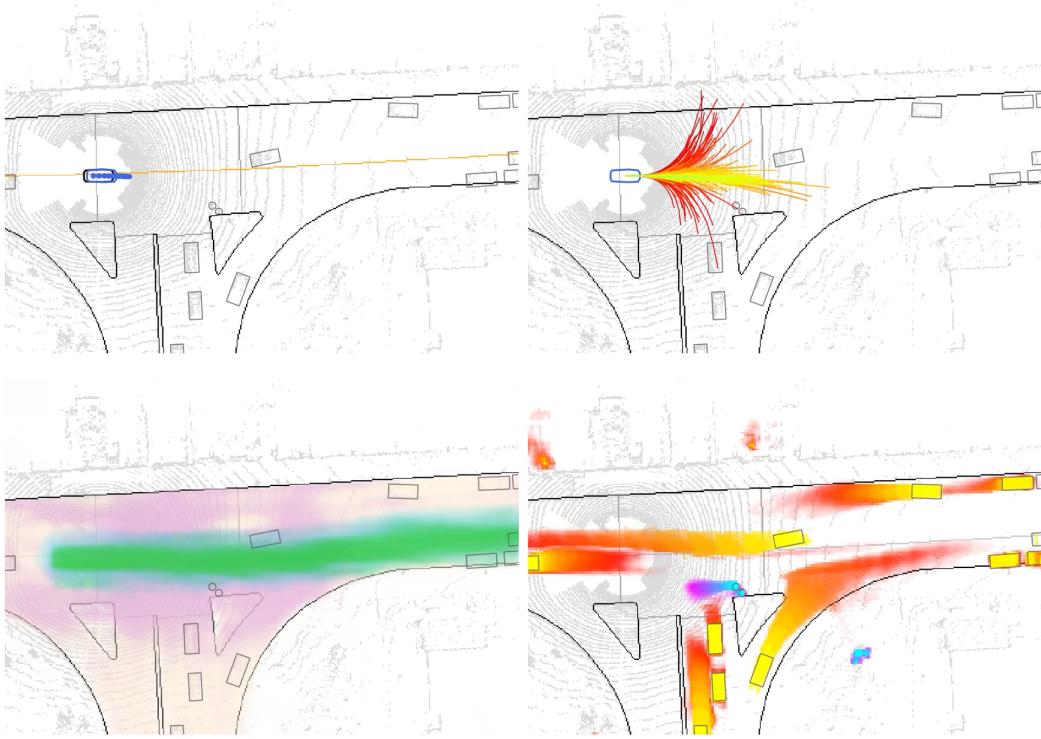


Figure 13: **Yielding scenario at an intersection.** Top right showcases a visualization of the cost of the retrieved trajectory samples. The warmer the color the higher the cost.

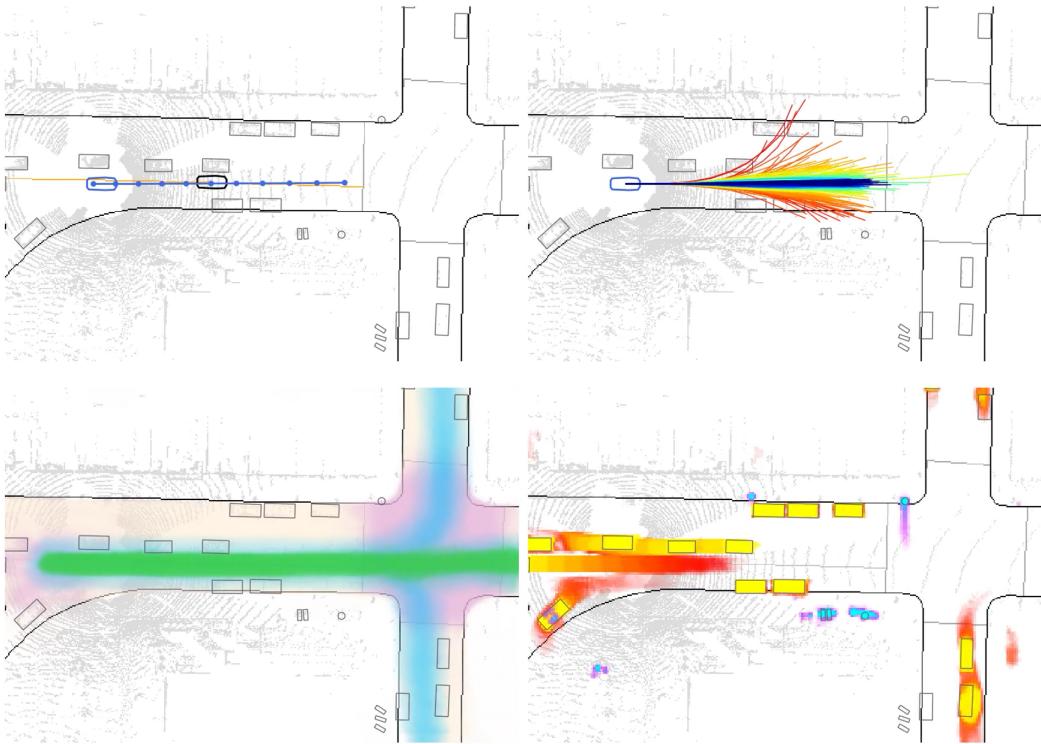


Figure 14: **Cruising scenario.** As can be seen in the top right, the fast moving straight trajectories achieve the lowest score as there is no one in front of the SDV.

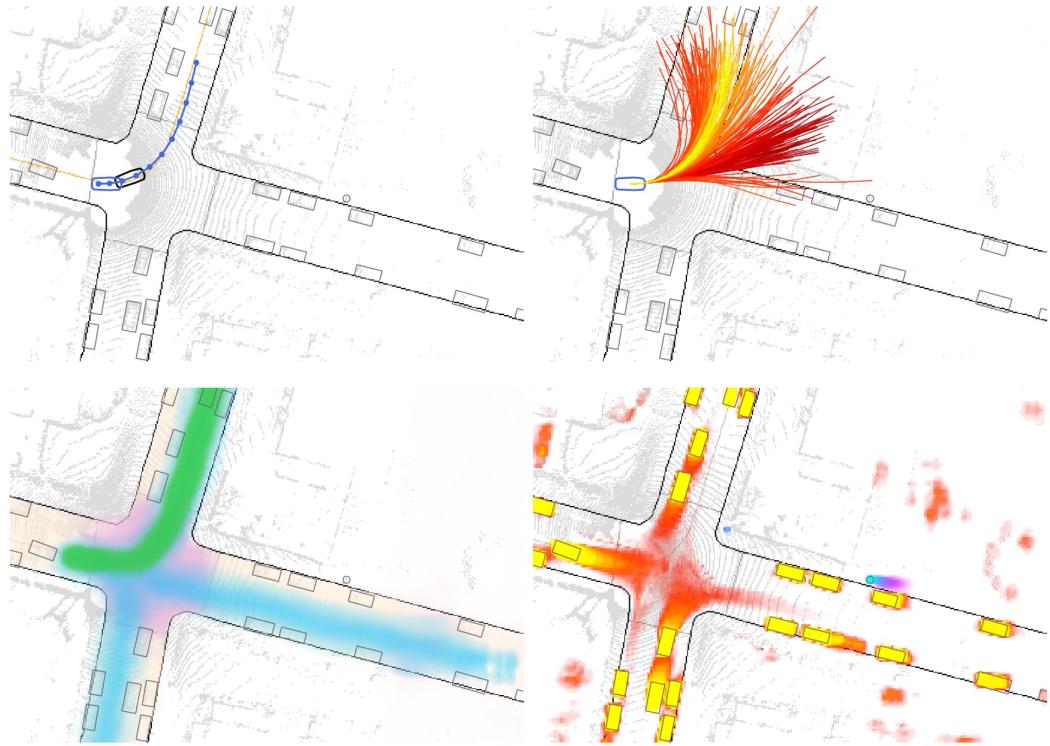


Figure 15: **Left turn scenario** in which the SDV progresses fast.

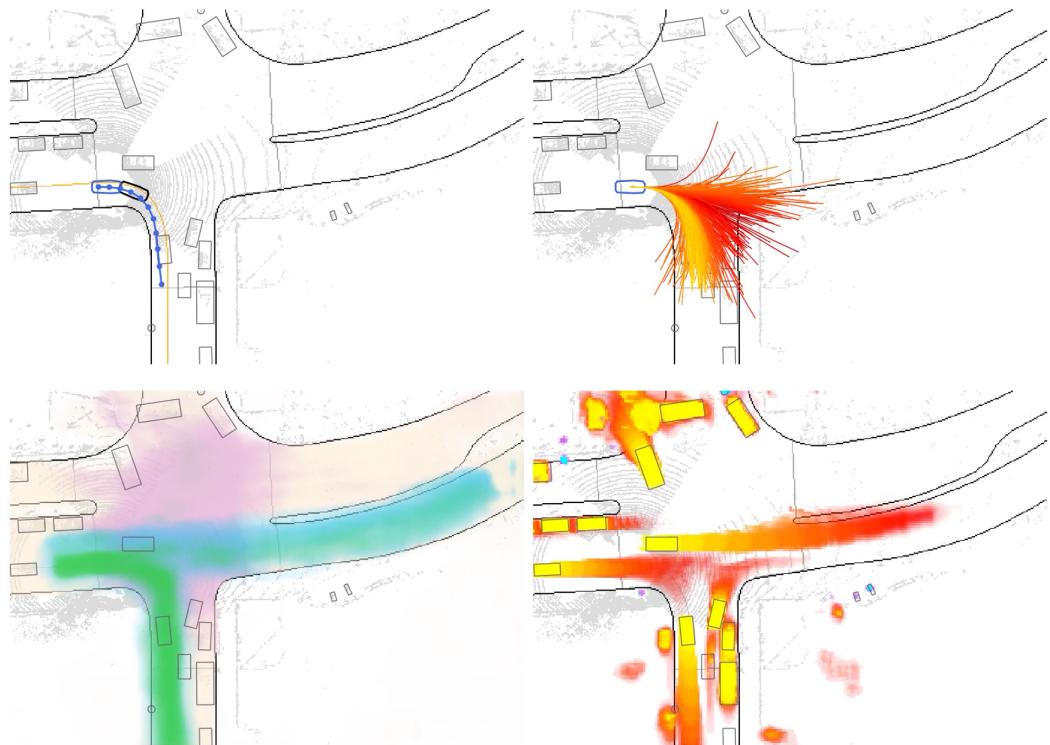


Figure 16: **Right turn scenario.** We can see how the trajectory samples adapt to the current SDV velocity.



Figure 17: **Planning comparison in closed-loop** MP3 is the only method that follows the route and does not collide in this example.

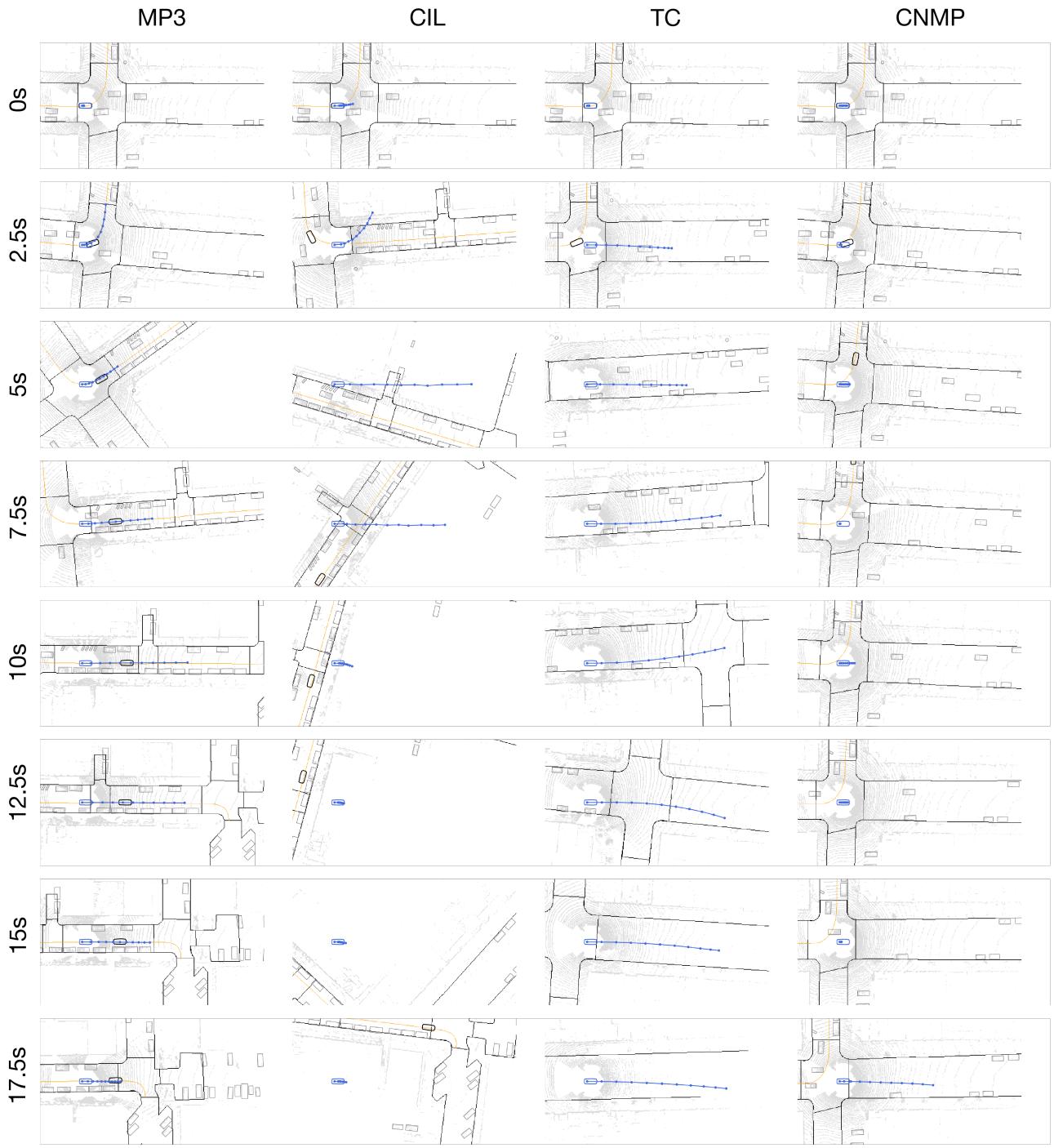


Figure 18: **More planning comparison in closed-loop.** MP3 is the only method that manages to effectuate the unprotected left turn

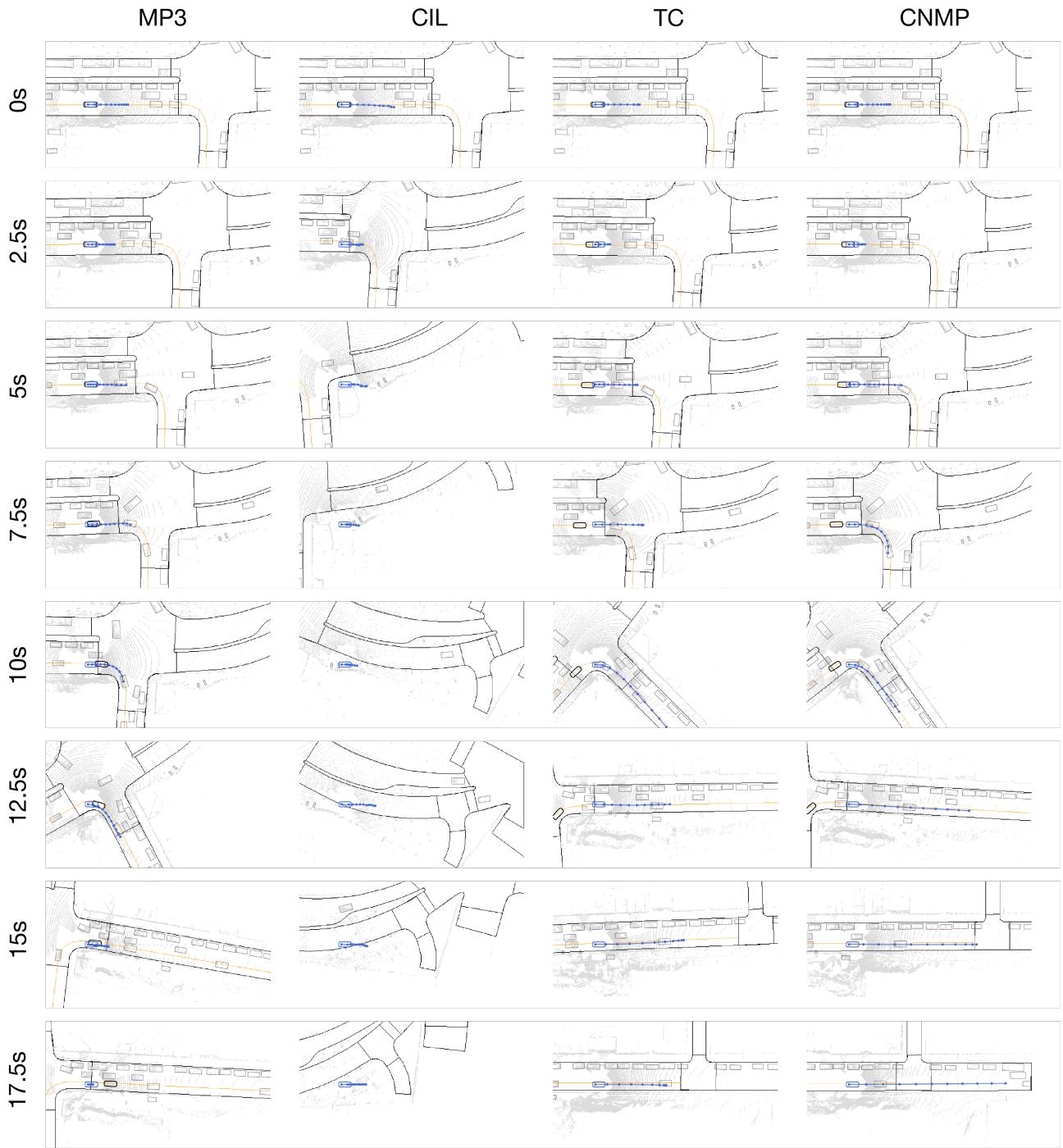


Figure 19: **More planning comparison in closed-loop.** MP3 closely imitates the expert. CIL diverges from the route. TC and CNMP stay on the route but collide with other vehicles.