

# End-to-End Driving with Online Trajectory Evaluation via BEV World Model

Yingyan Li\* Yuqi Wang\* Yang Liu Jiawei He Lue Fan✉ Zhaoxiang Zhang✉  
 NLPR, Institute of Automation, Chinese Academy of Sciences (CASIA)  
 State Key Laboratory of Multimodal Artificial Intelligence Systems (MAIS)  
 University of Chinese Academy of Sciences (UCAS)

## Abstract

*End-to-end autonomous driving has achieved remarkable progress by integrating perception, prediction, and planning into a fully differentiable framework. Yet, to fully realize its potential, an effective online trajectory evaluation is indispensable to ensure safety. By forecasting the future outcomes of a given trajectory, trajectory evaluation becomes much more effective. This goal can be achieved by employing a world model to capture environmental dynamics and predict future states. Therefore, we propose an end-to-end driving framework **WoTE**, which leverages a BEV World model to predict future BEV states for Trajectory Evaluation. The proposed BEV world model is latency-efficient compared to image-level world models and can be seamlessly supervised using off-the-shelf BEV-space traffic simulators. We validate our framework on both the NAVSIM benchmark and the closed-loop Bench2Drive benchmark based on the CARLA simulator, achieving state-of-the-art performance. Code is released at <https://github.com/liyingyanUCAS/WoTE>.*

## 1. Introduction

End-to-end autonomous driving [6, 30, 35] has achieved remarkable success in recent years and garnered widespread attention. While most approaches [21, 23, 25] primarily focus on predicting a high-quality trajectory, recent works [7, 27] suggest that predicting multiple trajectories simultaneously better reflects the multi-modal nature of driving and leads to improved performance. Given these multiple possible trajectories, it becomes crucial to effectively *evaluate* them in order to ensure the safety and reliability of end-to-end autonomous driving systems.

Traditional trajectory evaluation methods are rule-based [10, 14, 20, 37] and rely on perception results, such as bounding boxes and maps, to assess trajectories. These methods are sensitive to perception inaccuracies and may

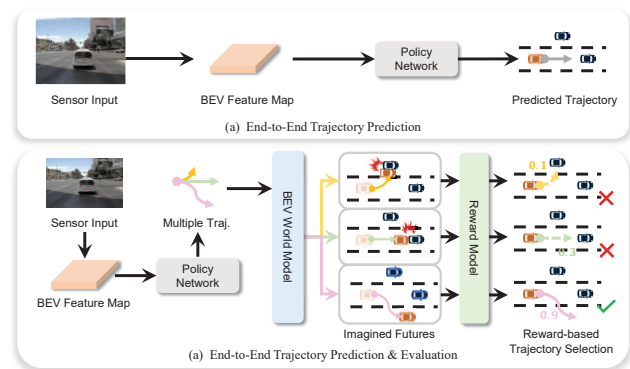


Figure 1. **The illustration of end-to-end trajectory evaluation with BEV world model.** (a) Previous end-to-end driving methods primarily focus on learning a high-quality trajectory. (b) Our WoTE consists of end-to-end trajectory prediction and evaluation. First, we predict multi-modal trajectories. Next, given these trajectories, we leverage the BEV world model to predict the corresponding future BEV states. The reward model evaluates these trajectories based on the predicted states and assigns rewards, selecting the highest-reward trajectory.

not be directly optimized in an end-to-end manner. Recently, some end-to-end driving approaches [7, 27] have incorporated trajectory evaluation. Their evaluations rely on the current state. However, evaluating a trajectory candidate is particularly challenging without knowledge of the *future* led by this trajectory. Similar to human drivers, who anticipate futures before making decisions, a trajectory evaluation network would benefit from predicting the corresponding future states of trajectory candidates.

To effectively predict future states, it is intuitive to adopt methodologies from reinforcement learning, where world models have successfully facilitated future state predictions. In reinforcement learning, model-based methods [5, 18, 32] have consistently shown superior performance compared to model-free approaches [17, 29], largely due to their ability of modeling the environmental dynamics and reasoning about future scenarios through world models. Inspired by this success, we propose adopting a world model [15, 19, 38–40]

\*: Co-first author, ✉: Co-advisor. Email: liyingyan2021@ia.ac.cn

within end-to-end autonomous driving frameworks to facilitate accurate prediction of future states, thus significantly improving trajectory evaluation.

However, leveraging the world model for trajectory evaluation presents some technical issues. First, a good representation of future scenarios is needed. Many existing world models for driving [19, 38, 40] predict the images of future scenarios with diffusion models [33], which can be time-consuming and not suitable for real-time driving. Second, we lack supervision for future states. Supervising the predicted future states of the world model is difficult because world models need to imagine multiple future states based on the multiple trajectory candidates, while only one future state is available in the real-world datasets.

To address these technical issues, we propose WoTE as shown in Fig.1. We propose to predict future states in the BEV space. The information in BEV space is much more compact than in raw images, which enables a more efficient single-step feed-forward future prediction than the time-consuming multi-step denoising in previous driving world models [19, 38, 40, 45]. This characteristic of BEV space makes WoTE highly suitable for real-time driving applications. Additionally, the predicted future BEV states can be easily supervised using off-the-shelf BEV-space traffic simulators such as nuPlan [3, 16]. These simulators model the scenario in BEV space by representing the map and objects using BEV semantic maps, creating a dynamic simulation of the driving environment. Consequently, we can use the BEV semantic maps from these simulated scenarios as supervision for the prediction of the BEV world model. Moreover, these simulators are equipped with well-established evaluation rules to assess the simulated future BEV scenarios, which provide target rewards.

We validate our framework on both NAVSIM [11] benchmark and closed-loop Bench2Drive [24] benchmark based on the CARLA [13] simulator and achieve state-of-the-art performance. Our experiments show that incorporating future states into trajectory evaluation is essential for improved performance. Additionally, the trajectory evaluation module, built on the world model shows great generalization ability across diverse trajectories. Our contributions are as follows:

- We highlight the importance of utilizing imagined future states for trajectory evaluation in end-to-end autonomous driving.
- We introduce WoTE, which consists of an end-to-end trajectory evaluation module based on a BEV world model. The world model in BEV space enables efficient real-time future prediction and overcomes the scarcity of multi-future supervisions.
- WoTE is validated on both the NAVSIM [11] benchmark and the closed-loop CARLA-based Bench2Drive [24] benchmark, achieving state-of-the-art performance.

## 2. Related Works

### 2.1. End-to-end Autonomous Driving

End-to-end autonomous driving [7, 22, 25, 43] is to train a model that directly maps sensor inputs to trajectories or control signals instead of decomposing the driving task into traditional sub-tasks. These works can be divided into two categories according to the training paradigm: imitation-learning-based and reinforcement-learning-based. Imitation learning is the most common practice in end-to-end autonomous driving. Their predicted trajectories are supervised by the expert trajectories. For instance, P3 [34], takes a differentiable semantic occupancy as the intermediate representation and learns actions from human trajectories. Following this, ST-P3 [21] learns spatial-temporal features for perception, prediction and planning tasks simultaneously. UniAD [22] proposed goal-oriented planning, combining and leveraging advantages of perception and motion prediction modules. VAD [25] introduces vectorized representations for end-to-end planning. There are also some reinforcement-learning-based methods. For example, MaRLn [36] utilizes implicit affordances to develop a reinforcement learning algorithm. LBC [4] trained a vision-based sensorimotor agent from the privileged agent. Combining imitation learning and reinforcement learning, Hydra-MDP [27] utilizes knowledge distillation to obtain supervisions from both rule-based planners and human drivers. It is a promising research direction as more supervisions are utilized.

### 2.2. Trajectory Evaluation

Trajectory evaluation is essential for ensuring the reliability and safety of autonomous driving systems. We divide the main trajectory evaluation methods into two categories: model-free and model-based. Model-free methods directly assess the learned policy using observed data without explicitly modeling environmental dynamics. For instance, Monte Carlo policy evaluation [1] uses the empirical mean return as the metric. The value function approximates the expectation based on sampled returns. Temporal-difference learning [9] uses bootstrapping, where the current estimate of the value function is used to generate target values for updating the value function. Subsequently, model-based trajectory evaluation methods [12] have gained increasing popularity in recent years. By leveraging learned world models that capture environmental dynamics, these methods enable more accurate and predictive policy evaluation. For example, PPUU [18] measures policy performance by quantifying uncertainty in predicted future states. Rails [5] evaluates trajectories through a tabular dynamic programming mechanism, and MARL-CCE [32] employs learned environmental dynamics to assess multi-agent policies in driving scenarios. However, these methods often rely on explicit trajectory representations and non-differentiable metrics, which limits

their end-to-end optimization capabilities. In contrast, our approach evaluates trajectories using BEV features and encoded trajectory embeddings, enabling fully differentiable and thus end-to-end trajectory evaluation.

### 3. Method

In this section, we introduce four key components of WoTE: Trajectory Predictor (Sec. 3.1), BEV World Model (Sec. 3.2), Reward Model (Sec. 3.3) and BEV Space Supervision (Sec. 3.4). As illustrated in Figure 2, our method integrates future state prediction via the BEV World Model and reward prediction via the Reward Model within a unified framework. Specifically, the *Trajectory Predictor* encodes the multi-modal observations into BEV state, represented as a BEV feature map, and provides multiple trajectory proposals. The *BEV World Model* predicts corresponding future BEV states based on these trajectory proposals. With the help of future states, the *Reward Model* predicts the reward of each trajectory. Finally, we choose the trajectory with the highest reward as the final trajectory.

#### 3.1. Trajectory Predictor

As shown in Figure 2 (a), the Trajectory Predictor aims to predict multiple trajectory candidates. First, the BEV encoder encodes the multi-modal sensor inputs into a unified BEV feature map. Then, given trajectory anchors, the trajectory refinement module refines trajectories based on the BEV feature map. We will now describe each part in detail.

**Multi-modal BEV Encoder.** Following TransFuser [31], our method leverages multi-modal sensor inputs, including LiDAR and multi-view RGB images. We use a BEV encoder [31] to transform the current multi-modal inputs into a unified BEV feature map  $\mathbf{B} \in \mathbb{R}^{h \times w \times c}$ . For convenience in the following text, we refer to the BEV feature map as the BEV state.  $h$  and  $w$  represent the height and width of the BEV state, while  $c$  denotes the number of channels.

**Trajectory Anchors.** Following methods in [7, 27], we generate trajectory anchors  $\tau$  by K-Means clustering. This clustering is performed on all the expert trajectories collected from the training dataset. The number of trajectory anchors is denoted as  $N$ .

**Trajectory Refinement.** As illustrated in Figure 2 (a), the trajectory refinement module takes the trajectory anchors and the current BEV states as inputs to predict the refined trajectories. Specifically, we first use a trajectory encoder  $\mathbf{TE}$ , which is implemented by an MLP, to encode the trajectory anchors  $\tau$  into feature vectors. Then, these feature vectors are used as queries to perform cross attention with the current BEV state  $\mathbf{B}$ , which serves as both the key and

value. The output of the cross-attention is then fed into an MLP head to predict the offsets. The offset is added to the trajectory anchors to produce the refined trajectories. The entire process can be expressed as:

$$\hat{\tau} = \tau + \text{MLP}(\text{CrossAttention}(\mathbf{TE}(\tau), \mathbf{B}, \mathbf{B})), \quad (1)$$

where  $\hat{\tau}$  is the refined trajectories.

#### 3.2. BEV World Model

In this section, we use the BEV World Model to predict the future BEV states, as illustrated in Figure 2 (b). The process is divided into two steps. First, the current BEV state and refined trajectories are combined to form the inputs for the world model. Next, the world model recurrently predicts future states over multiple time steps.

**Input of World Model.** Given  $N$  refined trajectories  $\hat{\tau}$  and the current BEV state  $\mathbf{B}_t$  at time  $t$ , we construct  $N$  state-action pairs as the inputs of world model. We use the trajectory encoder  $\mathbf{TE}$  in Eq. 1, with shared parameters, to encode  $\hat{\tau}$  into action embeddings  $\mathbf{A}_t$ . Given  $\mathbf{A}_t = \{\mathbf{a}_t^1, \mathbf{a}_t^2, \dots, \mathbf{a}_t^N\}$ , each state-action pair is represented as  $(\mathbf{B}_t, \mathbf{a}_t^i)$ , where  $i \in \{1, \dots, N\}$ .

**Recurrent Future Prediction.** Given the state-action pairs, we use a world model to predict their corresponding future states. In contrast to prior work [40], which predict the future in image space, we build a world model in BEV space as it is much more efficient. In detail, given a state-action pair  $(\mathbf{B}_t, \mathbf{a}_t^i)$ , we flatten  $\mathbf{B}_t$  into  $hw$  feature vectors. The dimension of each vector is  $c$ . Then we concatenate these vectors with the action embedding  $\mathbf{a}_t^i$ , resulting in a total of  $hw+1$  feature vectors, namely  $\mathbf{F}_i \in \mathbb{R}^{(hw+1) \times c}$ .  $\mathbf{F}_i$  are then fed into the world model, whose network architecture is a transformer encoder, as  $\mathbf{F}_{i+1} = \text{TransformerEncoder}(\mathbf{F}_i)$ . The output of the world model consists of  $h \times w + 1$  feature vectors, corresponding to the predicted future states  $\mathbf{B}_{t+1}^i$  and future actions  $\mathbf{a}_{t+1}^i$ . In summary, we can formulate this process in the following equation:

$$(\mathbf{B}_{t+1}^i, \mathbf{a}_{t+1}^i) = \text{WorldModel}(\mathbf{B}_t, \mathbf{a}_t^i), \quad (2)$$

Furthermore, this world model is able to predict the states  $\mathbf{B}_{t+K}^i$  of the future  $K$  steps in a recurrent manner, as shown by the following equation:

$$(\mathbf{B}_{t+K}^i, \mathbf{a}_{t+K}^i) = \text{WorldModel}(\mathbf{B}_{t+K-1}^i, \mathbf{a}_{t+K-1}^i). \quad (3)$$

The process above is conducted for every state-action pair in a parallel manner. It is important to note that the size of BEV states is relatively small (e.g.,  $h = w = 8$ ), so the world model does not introduce significant computational overhead. Next, we introduce the Reward Model, which predicts rewards for a trajectory based on its future states.

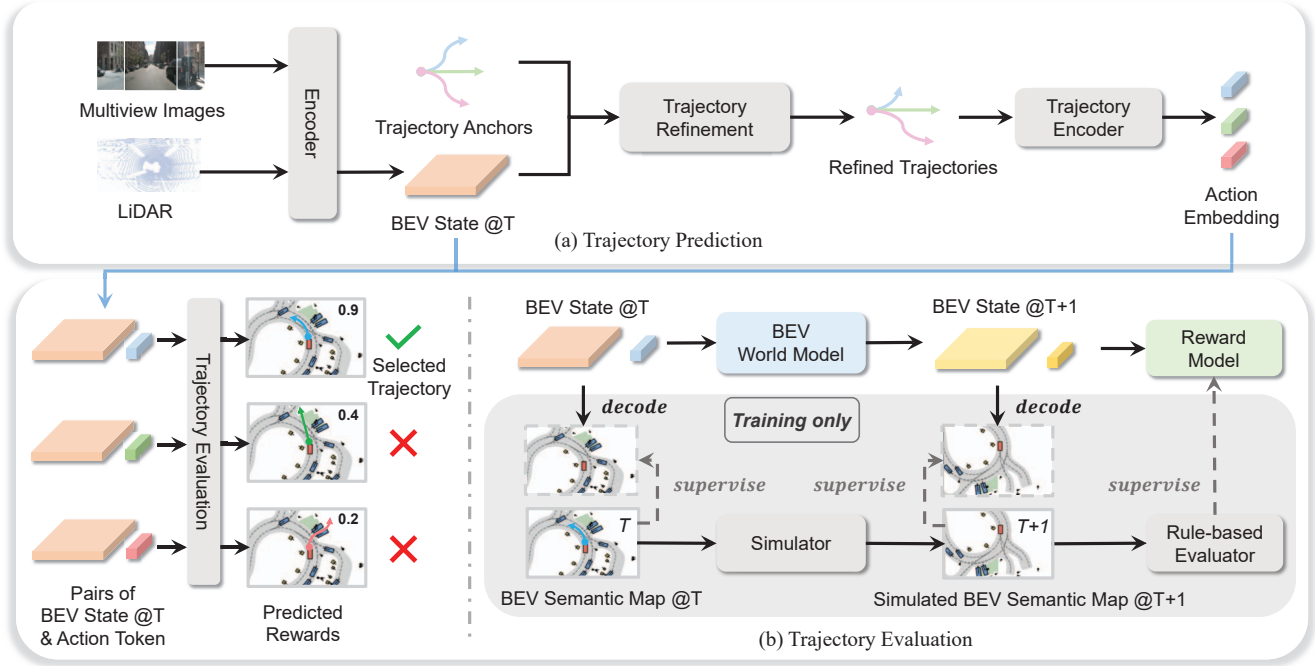


Figure 2. **Overall structure of WoTE.** Our model predicts and evaluates trajectory in an end-to-end manner. It is mainly divided into two parts. (a) Trajectory Prediction: the BEV encoder encodes the multi-modal observations into a BEV feature map (state) and proposes multiple trajectory candidates. (b) Trajectory Evaluation: Given the current BEV state and trajectory candidates, we leverage the BEV World Model to predict the corresponding future BEV states of these trajectories. The Reward Model then predicts rewards with the help of future BEV states. The trajectory with the highest reward is selected as the final trajectory.

### 3.3. Reward Model

In this section, we first present the reward types and then describe how they are learned.

**Reward Types.** Following [27], we have two types of rewards: the imitation reward and the simulation reward. The imitation reward  $r_{im}$  measures how well the predicted trajectory mimics an expert trajectory. The detailed implementation will be introduced in Sec. 3.4. The simulation rewards  $r_{sim}$  measures trajectory quality based on simulator-defined criteria. Following [11], we assess trajectories using the following five criteria: no collisions (NC), drivable area compliance (DAC), time-to-collision (TTC), comfort (Comf), and ego progress (EP). As a result, we have  $r_{sim} = \{r_{sim}^{NC}, r_{sim}^{DAC}, r_{sim}^{TTC}, r_{sim}^{Comf}, r_{sim}^{EP}\}$ . Given  $r_{im}$  and  $r_{sim}$ , the final reward  $r_{final}$  is defined as

$$r_{final} = -\left(w_1 \log r_{im} + w_2 \log r_{sim}^{NC} + w_3 \log r_{sim}^{DAC} + w_4 \log (5r_{sim}^{TTC} + 2r_{sim}^{Comf} + 5r_{sim}^{EP})\right), \quad (4)$$

where  $w_i$  are hyper-parameter.

**Reward Prediction.** Our method needs these rewards for trajectory evaluation in the inference time, so we propose

reward prediction. Take the  $i$ -th refined trajectory  $\hat{\tau}_i$  for illustration, the previous approaches [27] typically predict a reward based solely on the current state. With the help of the world model, our reward model predicts the reward  $\mathbf{r}_i$  of  $i$ -th trajectory based on both the current BEV state and future  $K$  steps BEV states. This process is illustrated in the following equation:

$$\mathbf{r}_i = \text{RewardModel}(\mathbf{B}_{all}^i, \mathbf{a}_{all}^i), \quad (5)$$

where  $\mathbf{B}_{all}^i = \text{cat}([\mathbf{B}_t, \mathbf{B}_{t+1}, \dots, \mathbf{B}_{t+K}])$  and  $\mathbf{a}_{all}^i = \text{MLP}(\text{cat}([\mathbf{a}_t^i, \mathbf{a}_{t+1}^i, \dots, \mathbf{a}_{t+K}^i]))$ .  $\text{cat}([\cdot])$  represents the concatenate operation along the channel dimension. In detail, Reward Model consists of three parts. It first applies 2D convolutions to  $\mathbf{B}_{all}^i$  to aggregate information from different regions and time steps of the BEV states. Next, a global average pooling is applied to form  $\mathbf{B}_{pool} \in \mathbb{R}^c$ ,  $c$  is the channel dimension. We then fed  $\mathbf{S}^i = \text{cat}([\mathbf{B}_{pool}, \mathbf{a}_{all}^i])$ , where  $\mathbf{S}^i \in \mathbb{R}^{2c}$ , into an MLP head to predict the reward  $\mathbf{r}_i \in \mathbb{R}^m$ , where  $m$  denotes the number of reward. Next, we introduce the definition of these rewards and select a final trajectory based on these rewards.

The refined trajectory with the highest final reward is selected as the final trajectory. Next, we introduce the supervision of these predicted rewards and BEV states.

### 3.4. Supervision in BEV Space

The world model predicts multiple future feature states corresponding to different trajectories. However, supervising these predictions in the image space is challenging, as driving logs provide only a single future trajectory. To address this, we propose supervising in the BEV space. Supervising in BEV space alleviates the difficulty of accurately modeling future states in image space while also improving computational efficiency. Many real-world traffic simulators support traffic simulations in the BEV space. They generate realistic and reliable future BEV scenarios along with rewards. We adopt nuPlan [3] as our traffic simulator for its broad adoption and realistic scenario simulations.

**Supervision of BEV States.** For BEV state supervision, we decode a semantic BEV map based on the BEV state to enable explicit supervision as Figure 2 (b) shows. We utilize upsampling and transposed convolution layers to decode a BEV semantic map from the BEV state. The dimension of the BEV semantic map is  $H \times W \times L$ , where  $L$  denotes the number of semantic classes. This BEV semantic map provides a comprehensive representation of a BEV scenario, covering classes including background, road, walkways, centerline, static objects, vehicles and pedestrians. Each element is represented as a one-hot encoded channel in  $L$ . For each time step  $t + k$ , We use Focal Loss [28] to supervise the predicted BEV semantic map  $\mathbb{B}_{t+k}$  as

$$\mathcal{L}_{\text{BEV}} = \text{FocalLoss}(\mathbb{B}_{t+k}, \mathbb{B}_{t+k}^*), \quad (6)$$

where  $\mathbb{B}_{t+k}^*$  is the ground truth simulated BEV semantic map at time  $t + k$  provided by the simulator.

**Supervision of Simulation Rewards.** Regarding the simulation rewards, we leverage the simulator to produce five rewards for evaluating a trajectory. Specifically, given the  $i$ -th trajectory  $\hat{\tau}^i$ , the simulator simulates the future location of the other agents and ego vehicle and produces a simulated future BEV semantic map with agents. Next, it uses a rule-based evaluator to evaluate this simulated future BEV scenarios for producing the corresponding target simulation rewards of  $\hat{\tau}^i$ . Given the rewards provided by the simulator, we use the Binary Cross Entropy (BCE) loss to supervise the predicted simulation reward  $r_{\text{sim}}$ . As a result, the loss is formulated as

$$\mathcal{L}_{\text{reward}}^{\text{sim}} = \text{BCE}(r_{\text{sim}}, r_{\text{sim}}^*). \quad (7)$$

**Supervision of Imitation Reward.** In addition to the supervision from the simulator, our framework is also guided by the expert driver.

We compute the target of imitation reward by first calculating the L2 distances  $d_i$  between the  $i$ -th trajectory anchor

and an expert trajectory. Then, we apply the negative of these distances from  $N$  trajectory anchors to the softmax function, yielding:  $r_{\text{im},i}^* = \frac{\exp(-d_i)}{\sum_{j=1}^N \exp(-d_j)}$ . Intuitively, a trajectory anchor closer to the expert trajectory than others will receive a higher reward. As a result, the imitation reward  $r_{\text{im}}^* \in \mathbb{R}^N$  quantifies the similarity of each trajectory anchor to the human trajectory. The predicted imitation reward  $r_{\text{im}}$ , which is produced by Eq. 5, are supervised using the Cross Entropy loss as

$$\mathcal{L}_{\text{reward}}^{\text{im}} = \text{CrossEntropy}(r_{\text{im}}, r_{\text{im}}^*). \quad (8)$$

**Supervision of Trajectory.** For supervising the refined trajectories, we adopt the commonly used winner-take-all strategy [46]. In this strategy, we select the trajectory anchor that is closest to the expert trajectory. Only the corresponding refined trajectory of this anchor will be supervised. Let this trajectory anchor be denoted as  $\tau_i$ . We use the L1 loss to measure the difference between the expert trajectory  $\tau^*$  and the refined trajectory  $\hat{\tau}_i$  as

$$\mathcal{L}_{\text{traj}} = |\hat{\tau}_i - \tau^*|. \quad (9)$$

Instead of supervising all refined trajectories, we focus solely on the most optimal one. This strategy helps the model capture a diverse range of trajectory patterns, with each anchor specializing in a particular trajectory modality.

The overall training loss  $\mathcal{L}_{\text{total}}$  is as follow:

$$\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{BEV}} + \mathcal{L}_{\text{reward}}^{\text{sim}} + \mathcal{L}_{\text{reward}}^{\text{im}} + \mathcal{L}_{\text{traj}}. \quad (10)$$

## 4. Experiments

### 4.1. Benchmark

**NAVSIM Dataset** The NAVSIM dataset [11] is constructed based on nuPlan [3]. Specifically, OpenScene [8] first down-sampled the nuPlan data from 10Hz to 2Hz to condense it into 120 hours of driving logs. NAVSIM resampled the data from OpenScene to emphasize challenging scenarios, reducing simple situations like straight-line driving. The dataset is divided into two parts: Navtrain and Navtest, comprising 1192 scenarios for training and validation and 136 scenarios for testing. Unlike nuScenes [2], which collects data in simpler, slower driving scenarios, NAVSIM ensures that challenging scenarios are prioritized, making simply fitting ego status insufficient for planning.

**NAVSIM Metrics** The original end-to-end driving metrics were primarily designed to evaluate the deviation between the predicted trajectories and those driven by human experts. However, this evaluation approach has proven inadequate, resulting in poor performance in real-world driving contexts. NAVSIM introduces more practical and reliable metrics, focusing on aligning open-loop and closed-loop metrics.

Method	Input	#Traj.	Traj. Eval.	NC $\uparrow$	DAC $\uparrow$	EP $\uparrow$	TTC $\uparrow$	Comf. $\uparrow$	PDMS $\uparrow$
Human	-	-	-	100	100	87.5	100	99.9	94.8
Constant Velocity	-	1	$\times$	69.9	58.8	49.3	49.3	100.0	21.6
Ego Status MLP	-	1	$\times$	93.0	77.3	62.8	83.6	100.0	65.6
VADv2 [41]	C	8192	Rule-based	97.9	91.7	77.6	92.9	100.0	83.0
UniAD [22]	C	1	Rule-based	97.8	91.9	78.8	92.9	100.0	83.4
LTF [31]	C	1	$\times$	97.4	92.8	79.0	92.4	100.0	83.8
PARA-Drive [41]	C	1	Rule-based	97.9	92.4	79.3	93.0	99.8	84.0
TransFuser [31]	C & L	1	$\times$	97.7	92.8	79.2	92.8	100.0	84.0
LAW [26]	C	1	$\times$	96.4	95.4	81.7	88.7	99.9	84.6
DRAMA [43]	C & L	1	$\times$	98.0	93.1	80.1	94.8	100.0	85.5
Hydra-MDP [27]	C & L	8192	Model-free	98.3	96.0	78.7	94.6	100.0	86.5
WoTE	C & L	256	Model-based	98.5	96.8	81.9	94.9	99.9	88.3

Table 1. **Comparison with the SOTA approaches on NAVSIM test set.** Rule-based: UniAD and PARA-Drive use occupancy maps, while VADv2 relies on vectorized maps for trajectory evaluation following specific rules. Model-free: Hydra-MDP evaluates trajectory without world modeling. Model-based: our WoTE evaluates trajectory with the assistance of the BEV world model. NC: no at-fault collision. DAC: drivable area compliance. EP: ego progress. TTC: time-to-collision. Comf.: comfort. PDMS: the predictive driver model score. Traj.: Trajectory. Eval.: Evaluation. C: Camera. L: LiDAR.

Method	Traj. Eval.	Success Rate $\uparrow$	Driving Score $\uparrow$
AD-MLP [44]	-	0.00	18.05
UniAD [22]	Rule-based	16.36	45.81
VAD [25]	Rule-based	15.00	42.35
TCP [42]	-	30.00	59.90
WoTE	Model-based	31.36	61.71

Table 2. **Comparison with SOTA approaches on closed-loop Bench2Drive [24] benchmark on CARLA simulator.** Traj.: Trajectory. Eval.: Evaluation.

Specifically, NAVSIM evaluates model performance using the Predictive Driver Model Score (PDMS), which is calculated based on five factors: No At-Fault Collision (NC), Drivable Area Compliance (DAC), Time-to-Collision (TTC), Comfort (Comf.), and Ego Progress (EP). The PDMS is calculated as  $PDMS = NC \times DAC \times \frac{5 \times EP + 5 \times TTC + 2 \times C}{12}$ .

**Bench2Drive Dataset and Metrics** To further evaluate our framework, we conduct closed-loop experiments in the CARLA simulator [13]. Specifically, we adopt the recently proposed Bench2Drive [24] benchmark as our closed-loop evaluation benchmark. Bench2Drive provides a standardized training dataset on CARLA, ensuring fair comparisons across different methods. The benchmark consists of 220 short evaluation routes (around 150m each) distributed across all CARLA towns, with each route containing one safety-critical scenario. This design effectively reduces the evaluation variance. For metric, it utilizes the standard CARLA metric, Driving Score (DS), as the primary metric. Additionally, Bench2Drive reports the success rate, which represents the proportion of successfully completed routes.

A route is considered successful if and only if DS reaches 100% on this route.

## 4.2. Implementation Details

**NAVSIM** For input data, it is aligned with TransFuser [31]. We concatenate the front-view image with center-cropped front-left and front-right images, resulting in a combined resolution of  $256 \times 1024$  pixels. For LiDAR, the  $64m \times 64m$  point cloud surrounding the ego vehicles is used. For network architecture, we employ ResNet34 as the backbone for BEV feature extraction following TransFuser [31]. The world model consists of two transformer decoder layers. We use  $N = 256$  trajectory anchors. The predicted future states are of time  $t + 2s, t + 4s$  when the current is of time  $t$ . The reward weights are set as follows:  $w_1 = 0.1, w_2 = w_3 = 0.5, w_4 = 1.0$ . The training is conducted on the Navtrain split using 8 NVIDIA L20 GPUs with a total batch size of 256, distributed across 50 epochs. In the ablation study section, we train our model for 20 epochs instead of 50 to accelerate the training process. For fast training, we pre-compute the simulation results (BEV semantic maps and scores) based on the trajectory anchors. We then feed trajectory anchors into the trajectory evaluation module during training. During testing, the trajectory evaluation module evaluates the refined trajectory instead of the anchor trajectories. We utilize the Adam optimizer with a learning rate of  $1e-4$ .

**Bench2Drive** For Bench2Drive, we adopt TCP [42] as our baseline and integrate the trajectory evaluation module. We choose TCP as it is the best open-source framework officially

Traj. Eval.	Future States	NC $\uparrow$	DAC $\uparrow$	EP $\uparrow$	TTC $\uparrow$	Conf. $\uparrow$	PDMS $\uparrow$
×	×	96.4	91.5	76.2	90.3	99.9	81.0
✓	×	97.1	92.9	78.2	91.9	100.0	83.2
✓	✓	98.0	94.7	79.9	93.4	100.0	85.6

Table 3. **Ablation study on trajectory evaluation and future state prediction.** Traj: Trajectory. Eval.: Evaluation.

provided by Bench2Drive. Similar to NAVSIM, we utilize imitation rewards along with simulation-based rewards, including No Collision (NC) and Drivable Area Compliance (DAC). The number of trajectory anchors is set to 256. Notably, rather than employing the winner-take-all strategy (Eq. 9) to supervise only the best trajectory, we find that supervising all trajectories leads to improved performance. The model is trained for 27 epochs with a batch size of 300. We use the Adam optimizer with a learning rate of  $1e-4$ .

### 4.3. Comparison with SOTA

**NASIM** Table 1 compares our method with state-of-the-art approaches on the NAVSIM test set. By leveraging the world model to simulate future scenarios and guide trajectory evaluation, our method achieves a PDMS of 87.1. WoTE outperforms the previous model-free approach, Hydra-MDP, highlighting the advantages of model-based trajectory evaluation.

**Bench2Drive** To further validate our approach, we conduct a closed-loop evaluation on the Bench2Drive benchmark within the CARLA simulator. As shown in Table 2, our method improves the Driving Score by 1.81 points, demonstrating its effectiveness in various evaluation settings.

### 4.4. Ablation Studies

**Trajectory evaluation and future state prediction matter.** In this experiment, we investigate the importance of the trajectory evaluation and future state prediction as Table 3 shows. The first row in Table 3 represents our baseline TransFuser [31], which performs trajectory prediction only. In the second row, we change to our framework, which consists of both trajectory prediction and trajectory evaluation. However, we do not use the predicted future states as the input of Reward Model. That is,  $\mathbf{B}_{\text{all}} = \mathbf{B}_t$  and  $\mathbf{a}_{\text{all}} = \mathbf{a}_t$  in Eq. 5. In the third row, we integrate the BEV world model to predict future states, leading to notable performance improvements across multiple metrics.

**Imitation and simulation rewards are complementary.** To investigate how each type of reward affects driving performance, we conduct an ablation study, as shown in Table 4. During inference, we adjust the reward weights in Eq. 4 to isolate their effects: setting  $w_1 = 0$  enables evaluation with only simulation rewards, while setting  $w_2 = w_3 = w_4 = 0$  tests performance with only an imitation reward. The results indicate that imitation and simulation rewards emphasize

Imi. Reward	Sim. Rewards	NC $\uparrow$	DAC $\uparrow$	EP $\uparrow$	TTC $\uparrow$	Conf. $\uparrow$	PDMS $\uparrow$
✓	×	97.9	92.5	77.7	93.4	99.9	83.5
×	✓	96.5	94.7	79.2	90.1	99.4	83.6
✓	✓	98.0	94.7	79.9	93.4	100.0	85.6

Table 4. **Ablation study on the imitation and simulation rewards.** The experiment shows that imitation and simulation rewards are complementary. Imi: Imitation. Sim.: Simulation.

Future Prediction Steps	NC $\uparrow$	DAC $\uparrow$	EP $\uparrow$	TTC $\uparrow$	Conf. $\uparrow$	PDMS $\uparrow$
0s $\rightarrow$ 4s	97.4	93.6	79.1	91.9	100.0	84.0
0s $\rightarrow$ 2s $\rightarrow$ 4s	98.0	94.7	79.9	93.4	100.0	85.6

Table 5. **Ablation study on the different number of time steps in the future prediction.** Our BEV world model predicts future states in a recurrent manner. A  $\rightarrow$  B: the predicted future state at time B is based on the state at time A.

#Trajs.	NC $\uparrow$	DAC $\uparrow$	EP $\uparrow$	TTC $\uparrow$	Conf. $\uparrow$	PDMS $\uparrow$	Latency $\downarrow$
64	97.4	94.1	79.6	91.9	100.0	84.5	17.2 ms
128	97.9	94.6	80.0	93.0	100.0	85.4	17.9 ms
256	98.0	94.7	79.9	93.4	100.0	85.6	18.7 ms

Table 6. **Ablation study on the different number of Trajectories.** The number of trajectories is the same during training and testing. The latency is evaluated on an NVIDIA L20 GPU. Trajs.: Trajectories.

different aspects of planning. Trajectory evaluation using imitation rewards performs better in NC and TTC, while using simulation rewards excels in DAC and EP. By integrating both, the model leverages their respective strengths, achieving improved overall performance across all key metrics.

**Recurrent future state prediction helps.** Our framework supports predicting future states recurrently, as detailed in Eq. 3. In Table 5, we present the results obtained by varying the number of BEV states predicted by the world model. Our findings indicate that using a finer time step for future state prediction significantly improves performance. This is because a finer time step provides richer temporal information, which aids trajectory evaluation.

**More trajectories help.** This ablation study explores the effect of varying the number of trajectories on overall performance. Specifically, the model in each row is trained and tested under the same number of trajectories, and the world model is fed with refined trajectories. As shown in Table 6, using a small number of trajectories, such as 64, significantly degrades performance. As the number of trajectories increases, performance improves notably, with a marked improvement observed when increasing from 64 to 128 trajectories. However, as the number of trajectories increases from 128 to 256, the performance gains diminish,

#Train Traj.	#Test Traj.	Traj. Refined	NC $\uparrow$	DAC $\uparrow$	EP $\uparrow$
256	256	×	96.6	94.6	79.0
256	1024	×	97.2	95.6	79.3
256	256	✓	98.0	94.7	79.9

Table 7. **Generalization of the trajectory across different trajectory configurations.** the input trajectories of BEV world model a jectory.

indicating that the performance of the m optimal level. Based on these results, w default number of trajectories for our ex **Latency analysis.** A common concern is of our framework. To address this, we n on an NVIDIA L20 GPU, as shown in GPU processes state-action pairs in para remains efficient even as the number of t The total latency is only 18.7 ms, well v requirements for end-to-end autonomou

#### Generalization ability across differer

shown in Table 7, we investigate the generalization ability of our trajectory evaluation module. To be specific, we sample  $N = 1024$  instead of  $N = 256$  trajectory anchors from the dataset. Since our model is trained under the setting of  $N = 256$  trajectory anchors, the model has never seen these  $N = 1024$  trajectories. Surprisingly, we find a 1.3 PDMS increase when using these unseen trajectory anchors. Besides, the model performs even better when using refined trajectories as inputs. These refined trajectories continuously change based on varying sensor inputs. This shows that the proposed world model has great generalization ability when dealing with unseen trajectories.

#### 4.5. Visual Analysis

**End-to-end planning trajectories.** To highlight the improvements in end-to-end driving achieved through trajectory evaluation, we present visualizations of end-to-end planning trajectories in Figure 3.

**Trajectory rewards.** Figure 4 illustrates the predicted rewards for different trajectories.

#### 5. Conclusion

In this paper, we introduced a novel framework that leverages a BEV world model for end-to-end trajectory evaluation in autonomous driving. By integrating a BEV world model, we enable a more informed evaluation process that considers the dynamic evolution of driving scenarios, leading to more effective trajectory evaluation. our approach benefits from dense supervision provided by BEV-space traffic simulators, which supply both semantic future states and rule-based

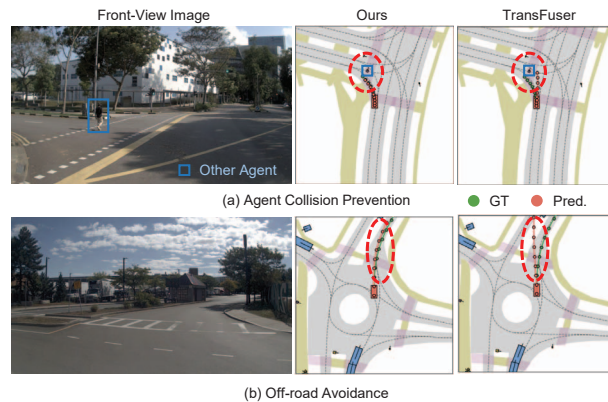


Figure 3. **Visualization of end-to-end planning trajectories.** We compare our method with TransFuser [31], which serves as our trajectory prediction baseline without a trajectory evaluation module. Our trajectory evaluation module effectively filters out low-quality trajectories and helps avoid collisions.

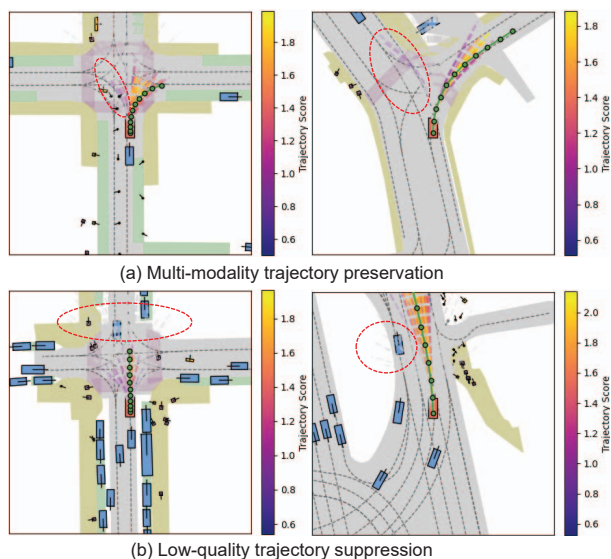


Figure 4. **Visualization of rewards of all trajectories.** Brighter colors indicate higher trajectory rewards, while gray trajectories correspond to those with very low rewards. (a): WoTE effectively retains diverse multi-modal trajectories, as long as they comply with traffic rules and do not result in unsafe outcomes. (b): it demonstrates that trajectories considered unreasonable receive significantly lower rewards, highlighting the effectiveness of our trajectory evaluation module.

reward targets. Our method achieves state-of-the-art performance on the NAVSIM and Bench2Drive benchmarks while maintaining real-time efficiency. Overall, our work establishes trajectory evaluation as an important research direction for end-to-end autonomous driving. We hope our framework serves as a strong baseline and inspires further research in end-to-end online trajectory evaluation.

## 6. Acknowledgements

This work was supported in part by the National Science and Technology Major Project (No. 2022ZD0116500), the National Natural Science Foundation of China (No. U21B2042, No. 62320106010).

## References

- [1] Sébastien MR Arnold, Pierre L’Ecuyer, Liyu Chen, Yi-fan Chen, and Fei Sha. Policy learning and evaluation with randomized quasi-monte carlo. *arXiv preprint arXiv:2202.07808*, 2022. 2
- [2] Holger Caesar, Varun Bankiti, Alex H. Lang, Sourabh Vora, Venice Erin Liong, Qiang Xu, Anush Krishnan, Yu Pan, Giancarlo Baldan, and Oscar Beijbom. nuScenes: A multimodal dataset for autonomous driving. *CVPR*, 2020. 5
- [3] Holger Caesar, Juraj Kabzan, Kok Seang Tan, Whye Kit Fong, Eric Wolff, Alex Lang, Luke Fletcher, Oscar Beijbom, and Sammy Omari. nuplan: A closed-loop ml-based planning benchmark for autonomous vehicles. *arXiv preprint arXiv:2106.11810*, 2021. 2, 5
- [4] Dian Chen, Brady Zhou, Vladlen Koltun, and Philipp Krähenbühl. Learning by cheating. In *Conference on Robot Learning*, pages 66–75. PMLR, 2020. 2
- [5] Dian Chen, Vladlen Koltun, and Philipp Krähenbühl. Learning to drive from a world on rails. In *CVPR*, pages 15590–15599, 2021. 1, 2
- [6] Li Chen, Penghao Wu, Kashyap Chitta, Bernhard Jaeger, Andreas Geiger, and Hongyang Li. End-to-end autonomous driving: Challenges and frontiers. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2024. 1
- [7] Shaoyu Chen, Bo Jiang, Hao Gao, Bencheng Liao, Qing Xu, Qian Zhang, Chang Huang, Wenyu Liu, and Xinggang Wang. Vadv2: End-to-end vectorized autonomous driving via probabilistic planning. *arXiv preprint arXiv:2402.13243*, 2024. 1, 2, 3
- [8] OpenScene Contributors. Openscene: The largest up-to-date 3d occupancy prediction benchmark in autonomous driving, 2023. 5
- [9] Christoph Dann, Gerhard Neumann, and Jan Peters. Policy evaluation with temporal differences: A survey and comparison. *The Journal of Machine Learning Research*, 15(1): 809–883, 2014. 2
- [10] Daniel Dauner, Marcel Hallgarten, Andreas Geiger, and Kashyap Chitta. Parting with misconceptions about learning-based vehicle motion planning. In *Conference on Robot Learning*, pages 1268–1281. PMLR, 2023. 1
- [11] Daniel Dauner, Marcel Hallgarten, Tianyu Li, Xinshuo Weng, Zhiyu Huang, Zetong Yang, Hongyang Li, Igor Gilitschenski, Boris Ivanovic, Marco Pavone, et al. Navsim: Data-driven non-reactive autonomous vehicle simulation and benchmarking. *arXiv preprint arXiv:2406.15349*, 2024. 2, 4, 5
- [12] Christopher Diehl, Timo Sievernich, Martin Krüger, Frank Hoffmann, and Torsten Bertram. Umbrella: Uncertainty-aware model-based offline reinforcement learning leveraging planning. *arXiv preprint arXiv:2111.11097*, 2021. 2
- [13] Alexey Dosovitskiy, German Ros, Felipe Codevilla, Antonio Lopez, and Vladlen Koltun. Carla: An open urban driving simulator. In *CoRL*, 2017. 2, 6
- [14] Haoyang Fan, Fan Zhu, Changchun Liu, Liangliang Zhang, Li Zhuang, Dong Li, Weicheng Zhu, Jiangtao Hu, Hongye Li, and Qi Kong. Baidu apollo em motion planner. *arXiv preprint arXiv:1807.08048*, 2018. 1
- [15] Shenyuan Gao, Jiazhi Yang, Li Chen, Kashyap Chitta, Yihang Qiu, Andreas Geiger, Jun Zhang, and Hongyang Li. Vista: A generalizable driving world model with high fidelity and versatile controllability. *arXiv preprint arXiv:2405.17398*, 2024. 1
- [16] Cole Gulino, Justin Fu, Wenjie Luo, George Tucker, Eli Bronstein, Yiren Lu, Jean Harb, Xinlei Pan, Yan Wang, Xiangyu Chen, et al. Waymax: An accelerated, data-driven simulator for large-scale autonomous driving research. *Advances in Neural Information Processing Systems*, 36, 2024. 2
- [17] Tuomas Haarnoja, Aurick Zhou, Kristian Hartikainen, George Tucker, Sehoon Ha, Jie Tan, Vikash Kumar, Henry Zhu, Abhishek Gupta, Pieter Abbeel, et al. Soft actor-critic algorithms and applications. *arXiv preprint arXiv:1812.05905*, 2018. 1
- [18] Mikael Henaff, Alfredo Canziani, and Yann LeCun. Model-predictive policy learning with uncertainty regularization for driving in dense traffic. In *International Conference on Learning Representations*, 2019. 1, 2
- [19] Anthony Hu, Lloyd Russell, Hudson Yeo, Zak Murez, George Fedoseev, Alex Kendall, Jamie Shotton, and Gianluca Corrado. Gaia-1: A generative world model for autonomous driving. *arXiv preprint arXiv:2309.17080*, 2023. 1, 2
- [20] Shengchao Hu, Li Chen, Penghao Wu, Hongyang Li, Junchi Yan, and Dacheng Tao. St-p3: End-to-end vision-based autonomous driving via spatial-temporal feature learning. In *European Conference on Computer Vision*, pages 533–549. Springer, 2022. 1
- [21] Shengchao Hu, Li Chen, Penghao Wu, Hongyang Li, Junchi Yan, and Dacheng Tao. St-p3: End-to-end vision-based autonomous driving via spatial-temporal feature learning. In *ECCV*, 2022. 1, 2
- [22] Yihan Hu, Jiazhi Yang, Li Chen, Keyu Li, Chonghao Sima, Xizhou Zhu, Siqi Chai, Senyao Du, Tianwei Lin, Wenhai Wang, et al. Goal-oriented autonomous driving. *arXiv preprint arXiv:2212.10156*, 2022. 2, 6
- [23] Yihan Hu, Jiazhi Yang, Li Chen, Keyu Li, Chonghao Sima, Xizhou Zhu, Siqi Chai, Senyao Du, Tianwei Lin, Wenhai Wang, Lewei Lu, Xiaosong Jia, Qiang Liu, Jifeng Dai, Yu Qiao, and Hongyang Li. Planning-oriented autonomous driving. In *CVPR*, 2023. 1
- [24] Xiaosong Jia, Zhenjie Yang, Qifeng Li, Zhiyuan Zhang, and Junchi Yan. Bench2drive: Towards multi-ability benchmarking of closed-loop end-to-end autonomous driving. *arXiv preprint arXiv:2406.03877*, 2024. 2, 6
- [25] Bo Jiang, Shaoyu Chen, Qing Xu, Bencheng Liao, Jiajie Chen, Helong Zhou, Qian Zhang, Wenyu Liu, Chang Huang, and Xinggang Wang. Vad: Vectorized scene representation for efficient autonomous driving. In *ICCV*, 2023. 1, 2, 6
- [26] Yingyan Li, Lue Fan, Jiawei He, Yuqi Wang, Yuntao Chen, Zhaoxiang Zhang, and Tieniu Tan. Enhancing end-to-end

- autonomous driving with latent world model. *arXiv preprint arXiv:2406.08481*, 2024. 6
- [27] Zhenxin Li, Kailin Li, Shihao Wang, Shiyi Lan, Zhiding Yu, Yishen Ji, Zhiqi Li, Ziyue Zhu, Jan Kautz, Zuxuan Wu, et al. Hydra-mdp: End-to-end multimodal planning with multi-target hydra-distillation. *arXiv preprint arXiv:2406.06978*, 2024. 1, 2, 3, 4, 6
- [28] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. In *ICCV*, 2017. 5
- [29] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fiedelnd, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *nature*, 518(7540):529–533, 2015. 1
- [30] Aditya Prakash, Kashyap Chitta, and Andreas Geiger. Multimodal fusion transformer for end-to-end autonomous driving. In *CVPR*, pages 7077–7087, 2021. 1
- [31] Aditya Prakash, Kashyap Chitta, and Andreas Geiger. Multimodal fusion transformer for end-to-end autonomous driving. In *CVPR*, 2021. 3, 6, 7, 8
- [32] Guanren Qiao, Guorui Quan, Rongxiao Qu, and Guiliang Liu. Modelling competitive behaviors in autonomous driving under generative world model. In *European Conference on Computer Vision*, pages 19–36. Springer, 2024. 1, 2
- [33] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10684–10695, 2022. 2
- [34] 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 *ECCV*, 2020. 2
- [35] Ardi Tampuu, Tabet Matiisen, Maksym Semikin, Dmytro Fishman, and Naveed Muhammad. A survey of end-to-end driving: Architectures and training methods. *IEEE Transactions on Neural Networks and Learning Systems*, 33(4): 1364–1384, 2020. 1
- [36] Marin Toromanoff, Emilie Wirbel, and Fabien Moutarde. End-to-end model-free reinforcement learning for urban driving using implicit affordances. In *CVPR*, 2020. 2
- [37] Martin Treiber, Ansgar Hennecke, and Dirk Helbing. Congested traffic states in empirical observations and microscopic simulations. *Physical review E*, 62(2):1805, 2000. 1
- [38] Xiaofeng Wang, Zheng Zhu, Guan Huang, Xinze Chen, Jiagan Zhu, and Jiwen Lu. Drivedreamer: Towards real-world-driven world models for autonomous driving. *arXiv preprint arXiv:2309.09777*, 2023. 1, 2
- [39] Yuqi Wang, Ke Cheng, Jiawei He, Qitai Wang, Hengchen Dai, Yuntao Chen, Fei Xia, and Zhaoxiang Zhang. Drivingdojo dataset: Advancing interactive and knowledge-enriched driving world model. *arXiv preprint arXiv:2410.10738*, 2024.
- [40] Yuqi Wang, Jiawei He, Lue Fan, Hongxin Li, Yuntao Chen, and Zhaoxiang Zhang. Driving into the future: Multiview visual forecasting and planning with world model for autonomous driving. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14749–14759, 2024. 1, 2, 3
- [41] Xinshuo Weng, Boris Ivanovic, Yan Wang, Yue Wang, and Marco Pavone. Para-drive: Parallelized architecture for real-time autonomous driving. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 15449–15458, 2024. 6
- [42] Penghao Wu, Xiaosong Jia, Li Chen, Junchi Yan, Hongyang Li, and Yu Qiao. Trajectory-guided control prediction for end-to-end autonomous driving: a simple yet strong baseline. *NeurIPS*, 2022. 6
- [43] Chengran Yuan, Zhanqi Zhang, Jiawei Sun, Shuo Sun, Zefan Huang, Christina Dao Wen Lee, Dongen Li, Yuhang Han, Anthony Wong, Keng Peng Tee, et al. Drama: An efficient end-to-end motion planner for autonomous driving with mamba. *arXiv preprint arXiv:2408.03601*, 2024. 2, 6
- [44] Jiang-Tian Zhai, Ze Feng, Jinhao Du, Yongqiang Mao, Jiang-Jiang Liu, Zichang Tan, Yifu Zhang, Xiaoqing Ye, and Jingdong Wang. Rethinking the open-loop evaluation of end-to-end autonomous driving in nuscenec. *arXiv preprint arXiv:2305.10430*, 2023. 6
- [45] Yumeng Zhang, Shi Gong, Kaixin Xiong, Xiaoqing Ye, Xiao Tan, Fan Wang, Jizhou Huang, Hua Wu, and Haifeng Wang. Bevworld: A multimodal world model for autonomous driving via unified bev latent space. *arXiv preprint arXiv:2407.05679*, 2024. 2
- [46] Zikang Zhou, Jianping Wang, Yung-Hui Li, and Yu-Kai Huang. Query-centric trajectory prediction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 17863–17873, 2023. 5