RiskDreamer: Autonomous Driving via Entropy-Risk Balancing Action Expansion in Batch Planning with Trusted Traffic Simulations

Qingchao Liu, Chengzhi Gao, Xiangkun He, *Member, IEEE*, Hai Wang, *Senior Member, IEEE*, Chen Lv, *Senior Member, IEEE*, Yingfeng Cai, *Senior Member, IEEE*, and Long Chen

***Abstract*—Ensuring safety and achieving human-level driving performance remain significant challenges for autonomous vehicles. While model-based reinforcement learning with planners enhances sample efficiency and facilitates policy exploration, many such methods rely on planners employing fixed parameters to balance expected rewards and risks, which limits their adaptability in dynamic traffic scenarios. Furthermore, studies use simplified traffic simulations for training and evaluation often results in algorithms that overfit to homogeneous traffic agents, resulting in overly optimistic performance. To address these limitations, we introduce RiskDreamer, a novel framework that employs batch planning within the latent space of the world model. facilitating efficiently exploration. Notably, we extend the action space to incorporate balancing factors as direct action outputs, optimized at each step. This enables the dynamic weighting of entropy, risk, and expected reward, achieving adaptable behavior planning in diverse traffic conditions. Furthermore, RiskDreamer is trained within a trustworthy traffic scenario generation framework based on optimization algorithms, capable of producing heterogeneous traffic agents from real trajectory datasets. The microscopic behavioral characteristics and macroscopic aggregate metrics of the generated background agents align with real-world statistical distributions. Through experimental results, we demonstrate that our method achieve competitive performance compared to strong baseline methods. The code for our research is available at https://github.com/Gaochengzhi/RiskDreamer.**

***Index Terms*—Reinforcement learning, World model, Autonomous vehicle, Traffic simulation.**

I. Introduction

THE development of safe and efficient autonomous driving holds the promise of revolutionizing transportation, with the potential to enhance road safety [1], [2], improve traffic efficiency [3], [4], and increase driving convenience [5]. Realizing this vision requires the ability of autonomous vehicles (AVs) to operate safely and efficiently in dynamic traffic environments.

In recent years, artificial intelligence has greatly advanced autonomous driving, with reinforcement learning (RL) [6], [7], [8] playing a critical role. Nevertheless, achieving a proper balance between safety and efficiency is crucial for the practical application of RL-based autonomous driving systems. While Safe RL methods aim to guarantee safety by modifying optimization criteria [9] or employing policy projection techniques [10], they often struggle to satisfy constraints as cost thresholds approach zero. Moreover, these methods may satisfy safety constraints at the expense of task completion [11]. Alternatively, RL agents using internal world models can efficiently plan action trajectories within a latent space. This approach enables cost minimization while striving to maintain high returns [12].

Previous works have explored combining learned latent space dynamic models with trajectory planning to optimize policies [13], [14], [15], [16]. These methods generally employ a model-predicted trajectory planning approach, their distinctions primarily lie in how they generate trajectories and select the final action. Typically, they first learn a latent space representation and dynamics model of the environment, and then generate a set of candidate trajectories based on these models. Ultimately, the optimal action is chosen according to each method's defined criteria to maximize external rewards. However, the explicit rollout of multiple candidate trajectories still imposes a significant computational burden. Also, these approaches often require task-specific hyper-parameter tuning to achieve a balance between exploration and risk aversion, potentially compromising their adaptability in dynamic traffic scenarios.

Another notable limitation lies in the environment used for training and evaluating autonomous driving agents. One common approach uses large-scale real-world datasets [17], [18] but adopts an open-loop paradigm where background vehicles simply replay pre-recorded trajectories. This prevents the training agent from receiving feedback on how its actions influence the environment, hindering the agent's ability to generalize to highly interactive scenarios. Alternatively, some

Thanks to our editors and all the reviewers who reviewed the manuscript anonymously. This work was supported by the National Key R&D Program of China (2023YFB2504400); National Natural Science Foundation of China (52372413, 52225212); Overseas training plan for outstanding young and middle-aged teachers and principals in colleges and universities in Jiangsu Province and the Young Talent Cultivation Project of Jiangsu University. (*Corresponding author: Qingchao Liu.*)

Qingchao Liu is with the Automotive Engineering Research Institute, Jiangsu University, Zhenjiang 212013, China, and also with the School of Mechanical and Aerospace Engineering, Nanyang Technological University, Singapore 639798 (e-mail: lqc@ujs.edu.cn).

Chengzhi Gao Yingfeng Cai and Long Chen are with the Automotive Engineering Research Institute, Jiangsu University, Zhenjiang 212013, China (e-mail: gaochengzhi1999@gmail.com; caicaixiao0304@126.com; chenlong@ujs.edu.cn).

Xiangkun He is with the Shenzhen Institute for Advanced Study, University of Electronic Science and Technology of China, Shenzhen 518110, China (e-mail: xiangkun.he@uestc.edu.cn).

Hai Wang is with the School of Automotive and Traffic Engineering, Jiangsu University, Zhenjiang 212013, China (e-mail: wanghai1019@163.com).

Chen Lv is with the School of Mechanical and Aerospace Engineering, Nanyang Technological University, Singapore 639798 (e-mail: lyuchen@ntu.edu.sg).

studies employ homogeneous traffic agents from untuned closed-loop simulations with simplified driving tasks [19]. Such homogeneity fails to capture the inherent variability and unpredictability of real-world traffic participants. Consequently, agents trained in these environments may exhibit overly optimistic performance metrics, as they have not been adequately exposed to the challenges of navigating genuinely reactive and diverse traffic situations.

To address these limitations, we introduce RiskDreamer, a novel model-based reinforcement learning framework designed to improve safety and efficiency of autonomous driving agents. RiskDreamer builds upon the DreamerV3 [20] world model architecture to learn efficient dynamic representations. Building upon this foundation, RiskDreamer incorporates a novel batch planning methodology. This approach generates multiple imagined trajectories concurrently within the latent space during a single model rollout, significantly improving planning efficiency. Furthermore, RiskDreamer expands the agent's action space to directly output risk and entropy weights. These weights dynamically balance expected rewards, exploration, and risk aversion. Instead of relying on manually tuned hyperparameters, RiskDreamer learns to adjust these balancing factors at each decision step, directly outputting them from the policy network. This allows the agent to adapt its risk tolerance and exploration strategy based on the current traffic context and conditions. Finally, RiskDreamer is trained within a trustworthy traffic scenario generation framework based on optimization algorithms. This framework can generate heterogeneous traffic agents from real-world trajectory data, ensuring that both microscopic behaviors and macroscopic traffic flow characteristics align with statistical distributions observed in real-world traffic. This enables the agent to learn and generalize in a realistic and challenging simulation environment.

The key contributions of this work are summarized as follows:

1) We introduce a novel batch planning method within a world model's latent space, where the expanded action space directly controls risk and entropy weights, enabling dynamic balancing of reward, exploration, and risk in diverse traffic scenarios.
2) We develop a trustworthy traffic scenario generator using optimization on real-world data. It produces heterogeneous traffic agents with realistic behavior and statistics for training and evaluating autonomous driving agents.
3) Through extensive experiments and ablation studies, we demonstrate that our method effectively balances risk and efficiency, leading to improved performance in highly adversarial traffic scenarios.

The rest of the paper is organized as follows: Section II reviews related works. Section III details the RiskDreamer framework. Section IV presents experiments and results. Finally, Section V concludes the paper and discusses future work.

II. Related Works

A. Safe Reinforcement Learning

Safe RL is commonly formulated as a Constrained Markov Decision Process (CMDP) [21], which seeks to optimize agent rewards while adhering to safety constraints. Solutions to CMDPs cam be categorized into two primary approaches: Primal-Dual Optimization (PDO) [22] and Constrained Policy Optimization (CPO). PDO predominantly relies on Lagrangian relaxation, a technique that iteratively updates primal and dual variables to manage constraints. Within this category, the Lagrangian method [23] is a useful technique that transforms constrained optimization problems into their unconstrained counterparts. It offers straightforward implementation and avoids the computational burden of complex Hessian matrix computations. In contrast, Achiam *et al.* introduced the Constrained Policy Optimization (CPO) algorithm [24], which employs quadratic constraint optimization to handle constraints within RL. While CPO provides an approximation to constrained optimization, it generally entails higher computational costs compared to Lagrangian-based methods. Moreover, the performance of CPO can be affected by approximation errors and sampling inaccuracies.

However, traditional Lagrangian methods are susceptible to oscillations, particularly when operating near constraint boundaries [25]. Furthermore, these methods are sensitive to the initial setting of the Lagrange multiplier λ. Our method addresses these limitations by directly learning risk and entropy weights at each step, thereby dynamically balancing exploration and risk aversion in response to varying traffic scenarios.

B. Planning in Latent Space

World model [26] has succeed in learning potential dynamic models capable of inferring future latent representations of the environment. The Dreamer algorithm series [27], [28], [20], utilized model-based reinforcement learning through its recurrent state-space model (RSSM) [29]. By constructing internal world models, the algorithm enables agents to learn in virtual environments, reducing required physical interactions and enhancing both efficiency and safety in real-world applications. However, the greedy MDP behavior model in DreamerV3 struggles with long sequence decision-making, leading to suboptimal performance in continuous action sequence tasks. Huang *et al.* extended DreamerV3 [20] by incorporating the Lagrangian method and Curiosity Cross-Entropy Method [14] to develop SafeDreamer [30]. However, similar to most Lagrangian-based approaches, its performance is highly sensitive to the cost threshold [31]. Wen *et al.* [15] proposed an algorithm based on the constrained cross-entropy method, which optimizes constraints through iterative updates of elite samples. Despite its effectiveness, this approach incurs high computational costs. In contrast, our method improves planning efficiency by concurrently generating multiple imagined trajectories in latent space.

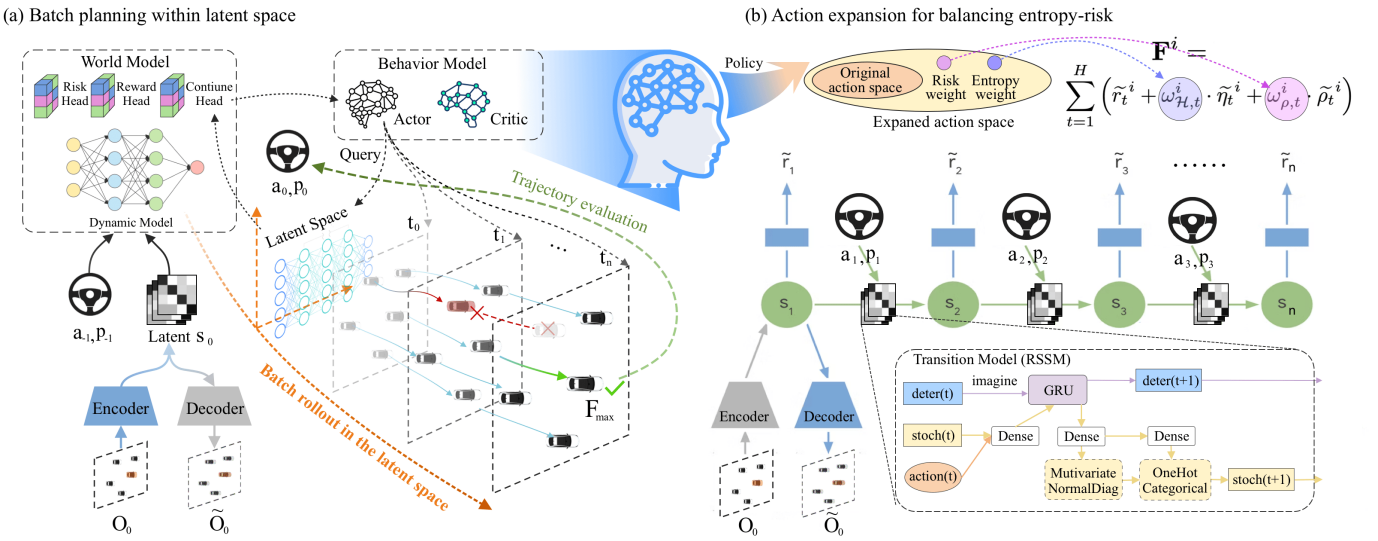


Fig. 1: Framework of the RiskDreamer algorithm. (a) Batch planning within latent space. (b) Action expansion for balancing entropy-risk.

C. Naturalistic Driving Simulation

The training environment is crucial for reinforcement learning, especially in the context of autonomous driving. Current traffic simulations typically rely on either replayed data or homogenous heuristic models to emulate background traffic agents. In a review of RL literature for autonomous vehicles, Noaeen *et al.* [32] identified that few papers incorporated simulations with diverse vehicle types. Additionally, They also noted the absence of explicit discussions about car-following parameters in autonomous vehicles RL research. Schrader *et al.* [33] calibrates the parameters for each pair of car-following vehicles. However, this calibration is based on pairs of following vehicles, lacking calibration for multi-vehicle interactive movements. Yan *et al.* [34] employed neural networks to simulate the behavior of traffic agents, thereby achieving macro-level statistical realism. However, the safety mapping network mechanism they utilized does not guarantee the plausibility of microscopic traffic trajectories. By contrast, our method generates heterogeneous traffic agents with realistic behavior and statistics, enhancing the realism and challenge of the training environment.

III. Methodology

In this section, we present the two core concepts of the RiskDreamer framework: batch planning in the latent space and action expansion for balancing entropy and risk, as illustrated in fig. 1. Additionally, we introduce the trustworthy traffic scenario generation framework used for training and evaluating autonomous driving agents, as shown in Fig. 2.

A. Framework Overview

The proposed RiskDreamer framework comprises three interconnected neural networks: a world model for predicting the outcomes of potential actions, a critic network for evaluating the future value of these outcomes, and a policy network for selecting actions that lead to the most valuable results. As the agent interacts with the environment, these components are trained concurrently based on replayed experiences. A central element of our approach is a batch online planner that integrates these three networks. At each decision-making step, this planner initiates a batch rollout using the world model's dynamics model. This process simultaneously explores multiple potential future trajectories within the latent space in a single rollout, offering enhanced planning efficiency. The policy network features an expanded action space, additionally outputting two balancing weights. These weights dynamically adjust the influence of expected rewards, entropy, and risk aversion at each step, derived from the world model's predictions. This transforms what are typically manually tuned hyperparameters into learnable, context-dependent strategies, enabling the agent to adapt its risk tolerance and exploration strategy according to the prevailing traffic conditions. This contrasts with other methods that rely on fixed hyperparameters, allowing for greater adaptability and potentially improved performance in complex and uncertain environments. We will now delve into the specifics of these components and their interactions.

B. Preliminaries from DreamerV3

RiskDreamer builds upon the foundation of the DreamerV3 world model. As depicted in Fig. 1 (a), at each time step t, the world model receives an observation $\mathbf{o}_t$ and an action $\mathbf{a}_t$ as input. First, an observation encoder ϕ_{enc} maps the observation $\mathbf{o}_t$ to a latent representation $\mathbf{s}_t$:

$$\mathbf{s}_t = \phi_{\text{enc}}(\mathbf{o}_t), \tag{1}$$

Given the current latent state $\mathbf{s}_t$ and action $\mathbf{a}_t$, the dynamics model ϕ_{dyn} predicts the next latent state $\widetilde{\mathbf{s}}_{t+1}$. The dynamics model is typically composed of a deterministic component and a stochastic component. Let $\boldsymbol{h}_t$ denote the deterministic state at time t. The dynamics model can be described as follows:

$$\begin{aligned} \boldsymbol{h}_{t+1} &= f_\phi(\boldsymbol{h}_t, \mathbf{s}_t, \mathbf{a}_t), \\ \widetilde{\mathbf{s}}_{t+1} &\sim p_\theta(\cdot|\boldsymbol{h}_{t+1}), \end{aligned} \tag{2}$$

where f_ϕ is a deterministic function learned from ϕ_{dyn}, and p_θ represents a probabilistic distribution parameterized by θ. The posterior distribution of the latent state $q_\phi(\mathbf{s}_t|\boldsymbol{h}_t, \mathbf{o}_t)$ is inferred by combining the prior $p_\theta(\cdot|\boldsymbol{h}_t)$ from the dynamics model with the current observation $\mathbf{o}_t$.

The world model comprises several prediction heads that decode the latent state into different modalities. In addition to the standard decoder for reconstructing the observation, and reward and continue heads from the original DreamerV3, RiskDreamer incorporates a risk prediction head. The outputs of these heads can be expressed as conditional probabilities:

$$\begin{aligned} \text{Decoder} &: \widetilde{\mathbf{o}}_t \sim \phi_{\text{dec}}(\widetilde{\mathbf{o}}_t \mid \mathbf{s}_t), \\ \text{RewardHead} &: \widetilde{\mathbf{r}}_t \sim \phi_{\text{reward}}(\widetilde{\mathbf{r}}_t \mid \mathbf{s}_t), \\ \text{RiskHead} &: \widetilde{\rho}_t \sim \phi_{\text{risk}}(\widetilde{\rho}_t \mid \mathbf{s}_t), \\ \text{ContinueHead} &: \widetilde{\mathbf{c}}_t \sim \phi_{\text{cont}}(\widetilde{\mathbf{c}}_t \mid \mathbf{s}_t), \end{aligned} \tag{3}$$

where r_t is the reward at time t, $\widetilde{\rho}_t$ is the predicted risk, and c_t indicates whether the episode continues at the next step. The training of the world model involves minimizing a loss function that includes the reconstruction loss for each head and the KL divergence between the prior and posterior distributions of the latent states.

Algorithm 1 Batch Planning in Latent Space

Input: Posterior distribution $\mathbf{s}_t$, embedded observation $\mathbf{e}_t$, training flag τ
Parameters: Planning horizon H, Batch number N
Initialize cumulative value $\mathbb{V}$, entropy $\mathbb{E}$, and risk $\mathbb{R} \leftarrow \mathbf{0}$
Expand $\mathbf{s}_t$ to N simulations: $\mathbf{s}_t \leftarrow \text{expand}(\mathbf{s}_t, N)$
for $t = 1$ to H **do**
  if τ and $t = 1$ **then** ▷ Exploration policy
    Sample actions: $\mathbf{a}_t \sim \pi_{\theta_e}(\cdot|\mathbf{s}_t)$
    Compute log probabilities: $\mathbf{L}_t \leftarrow \log \pi_{\theta_e}(\mathbf{a}_t|\mathbf{s}_t)$
  else ▷ Evaluation policy
    Deterministic action: $\mathbf{a}_t \sim \pi_{\theta_t}(\cdot|\mathbf{s}_t)$
    Select actions: $\mathbf{a}_t \leftarrow \text{mean}(\pi_{\theta_t}(\cdot|\mathbf{s}_t))$ ▷ Use the mean of the distribution
    Compute log probabilities: $\mathbf{L}_t \leftarrow \log \pi_{\theta_t}(\mathbf{a}_t|\mathbf{s}_t)$
  end if
  Predict next state: $\widetilde{\mathbf{s}}_{t+1} \sim \phi_{dyn}(\mathbf{s}_t, \mathbf{a}_t)$
  Predict risk: $\widetilde{\rho}_t \sim \phi_{risk}(\mathbf{s}_t)$
  Predict reward: $\widetilde{\mathbf{r}}_t \sim \phi_{rew}(\mathbf{s}_t)$
  Estimate policy entropy: $\widetilde{\eta}_t \leftarrow \text{Entropy}(\pi_{\theta_e}(\cdot|\mathbf{s}_t))$
  Retrieve risk weight $\omega_{\rho,t}$ and entropy weight $\omega_{\mathcal{H},t}$ from the expanded action space.
  $\omega_{\mathcal{H},t} \leftarrow \text{action}[-1]$ ▷ Entropy weight
  $\omega_{\rho,t} \leftarrow \text{action}[-2]$ ▷ Risk weight
  Update cumulative metrics:
  $\mathbb{V} \leftarrow \mathbb{V} + \widetilde{\mathbf{r}}_t$, $\mathbb{E} \leftarrow \mathbb{E} + \omega_{\mathcal{H},t} \cdot \widetilde{\eta}_t$, $\mathbb{R} \leftarrow \mathbb{R} + \omega_{\rho,t} \cdot \widetilde{\rho}_t$
  Store actions: $\mathbf{A}_t \leftarrow \mathbf{a}_t$
  Update state: $\mathbf{s}_t \leftarrow \mathbf{s}_{t+1}$
end for
Compute objective: $\mathbf{F} \leftarrow \mathbb{V} + \mathbb{E} + \mathbb{R}$
Select best simulation: $i^* \leftarrow \arg\max(\mathbf{F})$
Output: Best action $\mathbf{a}_{i^*}$, log probability $\mathbf{L}_{i^*}$

$$\begin{aligned} \mathcal{L}_{WM} = \mathbb{E}_{(\mathbf{o},\mathbf{a},r,\rho,c)\sim\mathcal{D}}\Big[&- \log p_{\phi_{\text{dec}}}(\mathbf{o}_t|\mathbf{s}_t) - \log p_{\phi_{\text{reward}}}(r_t|\mathbf{s}_t) \\ &- \log p_{\phi_{\text{risk}}}(\rho_t|\mathbf{s}_t) - \log p_{\phi_{\text{cont}}}(c_t|\mathbf{s}_t) \\ &+ \beta \cdot \max\Big(D_{KL}\big(q_\phi(\mathbf{s}_t|\mathbf{h}_t, \mathbf{o}_t) \\ &\big\|p_\theta(\mathbf{s}_t|\mathbf{h}_t)\big), \lambda\Big)\Big], \end{aligned} \tag{4}$$

The expectation $\mathbb{E}$ is over trajectories from the replay buffer $\mathcal{D}$, which includes observations $\mathbf{o}$, actions $\mathbf{a}$, rewards r, risks ρ, and continue flags c. The loss function promotes accurate reconstruction of observations and prediction of rewards, risks, and continue flags, conditioned on the latent state $\mathbf{s}_t$.

A KL divergence term, scaled by β, regularizes the latent dynamics. This term minimizes the difference between the posterior distribution $q_\phi(\mathbf{s}_t|\mathbf{h}_t, \mathbf{o}_t)$ (representation model) and the prior distribution $p_\theta(\mathbf{s}_t|\mathbf{h}_t)$ (transition model). The posterior infers $\mathbf{s}_t$ from the deterministic state $\mathbf{h}_t$ and observation $\mathbf{o}_t$, while the prior predicts $\mathbf{s}_t$ solely from $\mathbf{h}_t$. This regularization ensures consistency between the transition model's predictions and the observations. The $\max$ function, with threshold λ, implements the "free bits" method to prevent vanishing KL gradients.

The deterministic state $\mathbf{h}_t$ is updated recurrently via the dynamics model ϕ_{dyn}, typically using a GRU:

$$\mathbf{h}_t = f_\phi(\mathbf{h}_{t-1}, \mathbf{s}_{t-1}, \mathbf{a}_{t-1}). \tag{5}$$

Both q_ϕ and p_θ are modeled as diagonal Gaussian distributions, with parameters determined by neural networks.

The actor-critic component in RiskDreamer, similar to classic SAC [35], consists of an actor network $\pi_\theta(\mathbf{a}_t|\mathbf{s}_t)$ and a value network $V_\psi(\mathbf{s}_t)$. The actor network predicts a distribution over actions, while the value network estimates the expected future reward.

C. Batch Planning within Latent Space

The batch planning method of RiskDreamer leverages the world model to efficiently explore potential future scenarios. Instead of serially simulating one trajectory at a time, it generates and evaluates multiple trajectories in parallel within the latent space. The process begins with the current latent state $\mathbf{s}_t$ and the embedded observation $\mathbf{e}_t = \phi_{\text{enc}}(\mathbf{o}_t)$.

The batch planning process can be summarized in Algorithm 1. The algorithm takes as input the current latent state $\mathbf{s}_t$, the embedded observation $\mathbf{e}_t$, and a training flag τ that determines whether the exploration policy or the evaluation policy is used. The algorithm also requires the planning horizon H and the number of simulations N to be specified. The latent state $\mathbf{s}_t$ is expanded to N simulations, each of which is used to simulate a potential trajectory.

At each planning step within the horizon, the policy network proposes actions for each of the N simulated trajectories. For training (τ is true) and in the first step ($t = 1$), an exploration policy π_{θ_e} is used to sample actions, encouraging policy diversity. Subsequently, or during evaluation (τ is false or $t > 1$), the evaluation policy π_{θ_t} is employed, and the action

with the highest probability is selected. The world model then predicts the subsequent latent state for each simulation based on the chosen action. At each step t of the planning horizon, the algorithm also predicts the risk $\widetilde{\rho}_t$ and the value $\widetilde{v}_t$ of the current state, and estimates the entropy $\widetilde{\eta}_t$ of the policy.

The value of each imagined trajectory is implicitly evaluated through the cumulative sum of the predicted values, estimated policy entropies, and predicted risks. Let the N simulations be indexed. The cumulative value $\mathbb{V}$, entropy $\mathbb{E}$, and risk $\mathbb{R}$ are updated at each step t for each simulation. Specifically, for each simulation $i \in \{1, \ldots, N\}$:

$$\begin{aligned} \mathbb{V}^i &\leftarrow \mathbb{V}^i + \widetilde{v}_t{}^i, \\ \mathbb{E}^i &\leftarrow \mathbb{E}^i + \omega^i_{\mathcal{H},t} \cdot \widetilde{\eta}_t{}^i, \\ \mathbb{R}^i &\leftarrow \mathbb{R}^i + \omega^i_{\rho,t} \cdot \widetilde{\rho}_t{}^i, \end{aligned} \tag{6}$$

The objective $\mathbf{F}$ for each simulation is calculated as the sum of the cumulative value, entropy, and risk at the end of the planning horizon H:

$$\mathbf{F}^i = \mathbb{V}^i + \mathbb{E}^i + \mathbb{R}^i = \sum_{t=1}^{H} \left(\widetilde{r}_t{}^i + \omega^i_{\mathcal{H},t} \cdot \widetilde{\eta}_t{}^i + \omega^i_{\rho,t} \cdot \widetilde{\rho}_t{}^i \right), \tag{7}$$

Finally, the simulation with the highest objective is selected, and the first action of that simulation is returned as the best action. This batch planning approach allows the agent to consider a set of potential future scenarios in parallel and make decisions based on the aggregated value, entropy, and risk.

D. Action Expansion for Balancing Entropy and Risk

A key innovation of RiskDreamer is the expansion of the policy network's action space. Instead of directly outputting only the action to be taken in the environment, the policy network in RiskDreamer outputs additional parameters that control the balance between exploration, exploitation, and risk aversion. Specifically, the action output by the policy network π_φ is augmented to include weights for entropy and risk:

$$\mathbf{a}_t = [\mathbf{a}'_t, \omega_{\rho,t}, \omega_{\mathcal{H},t}], \tag{8}$$

where $\mathbf{a}'_t$ represents the action to be executed in the environment, $\omega_{\rho,t} \in [-1, 1]$ is the weight assigned to the predicted risk, and $\omega_{\mathcal{H},t} \in [-1, 1]$ is the weight assigned to the entropy of the state and policy distributions.

These weights are outputs of the policy network and are conditioned on the current latent state $\mathbf{s}_t$:

$$\begin{aligned} \omega_{\rho,t} &= \pi^{\rho}_{\varphi}(\mathbf{s}_t), \\ \omega_{\mathcal{H},t} &= \pi^{\mathcal{H}}_{\varphi}(\mathbf{s}_t). \end{aligned} \tag{9}$$

During the batch planning process, these weights are used to modulate the contribution of risk and entropy to the evaluation of each imagined trajectory. By making these balancing factors outputs of the policy network, RiskDreamer enables the agent to dynamically adapt its risk tolerance and exploration strategy based on the specific context and perceived risk of the situation. For instance, in high-risk scenarios, the policy network might learn to increase ω_ρ to prioritize risk aversion, while in less critical situations, it might increase $\omega_{\mathcal{H}}$ to encourage exploration. This mechanism provides a more flexible and adaptive approach compared to using fixed hyperparameters for balancing these factors.

E. Trustworthy Traffic Scenario Generation

To make simulation an effective tool for developing autonomous driving algorithms, the simulator needs to generate realistic traffic scenarios with precise distributions that accurately reflect the behavior of background traffic agents.

We achieve this by calibrating the parameters of EIDM [36] car-following models using aggregated feature distributions derived from real-world vehicle trajectory datasets as optimization targets. Through iterative refinement of these parameter distributions, we minimize the divergence between simulated and real trajectory features, thereby generating realistic and naturally interactive driving behaviors for background traffic agents.

Fig. 2 illustrates our generation process in detail. In step (a), we run the simulator as a black-box, high-cost function in the optimization process. Specifically, the simulator can be represented as a function $f(\mathbf{x})$, where $\mathbf{x} = [x_1, x_2, \ldots, x_N]^\top$ is the vector of input parameters for the car-following model that drives the background traffic agents. These parameters are subject to the inequality constraints specified by the boundaries shown in Table I:

TABLE I
BOUNDARIES OF THE CAR-FOLLOWING MODEL PARAMETERS

Parameters	Descriptions (Unit)	Car	Bus
τ_{mean}	Average time headway (s)	(0.5, 4)	(0.5, 4)
τ_{std}	Std. deviation of time headway $(-)$	(0, 10)	(0, 10)
v_{mean}	Average max speed (m/s)	(8, 26)	(8, 26)
v_{std}	Std. deviation of max speed $(-)$	(0, 20)	(0, 20)
a	Max acceleration (m/s^2)	(0.2, 4)	(0.2, 4)
d	Max deceleration (m/s^2)	(0.2, 4)	(0.2, 4)
L_{sublane}	Eagerness to align laterally in-lane $(-)$	(0, 1)	(0, 1)
L_{pushy}	Willingness to encroach laterally $(-)$	(0, 1)	(0, 1)
$L_{\text{speed\_gain}}$	Lane-changing for speed gain $(-)$	(0, 5)	(0, 5)
$L_{\text{assertive}}$	Acceptance of smaller gaps $(-)$	(1, 100)	(1, 100)
$L_{\text{cooperative}}$	Cooperative lane-changing $(-)$	(0, 1)	(0, 1)
$L_{\text{lookahead}}$	Adjustment for lookahead distance (m)	(2, 100)	(2, 100)

Meanwhile, the scenario configurations, including road topology, traffic flow, and vehicle type distribution, are pre-established based on the statistical properties of the dataset.

The trajectory data generated by the simulator will undergo (b) trajectory data collection and variable extraction. This process yields aggregated distributions of variables such as speed, acceleration, and headway for different types of agents (e.g., trucks and passenger cars). Subsequently, we employ the Kullback-Leibler (KL) divergence to compare the aggregated feature distributions between the simulated and real-world data. The KL divergence for the m-th feature is calculated as:

$$D^{(m)}_{\text{KL}} = \sum_i p^{(m)}_i \log \left(\frac{p^{(m)}_i}{q^{(m)}_i} \right), \tag{10}$$

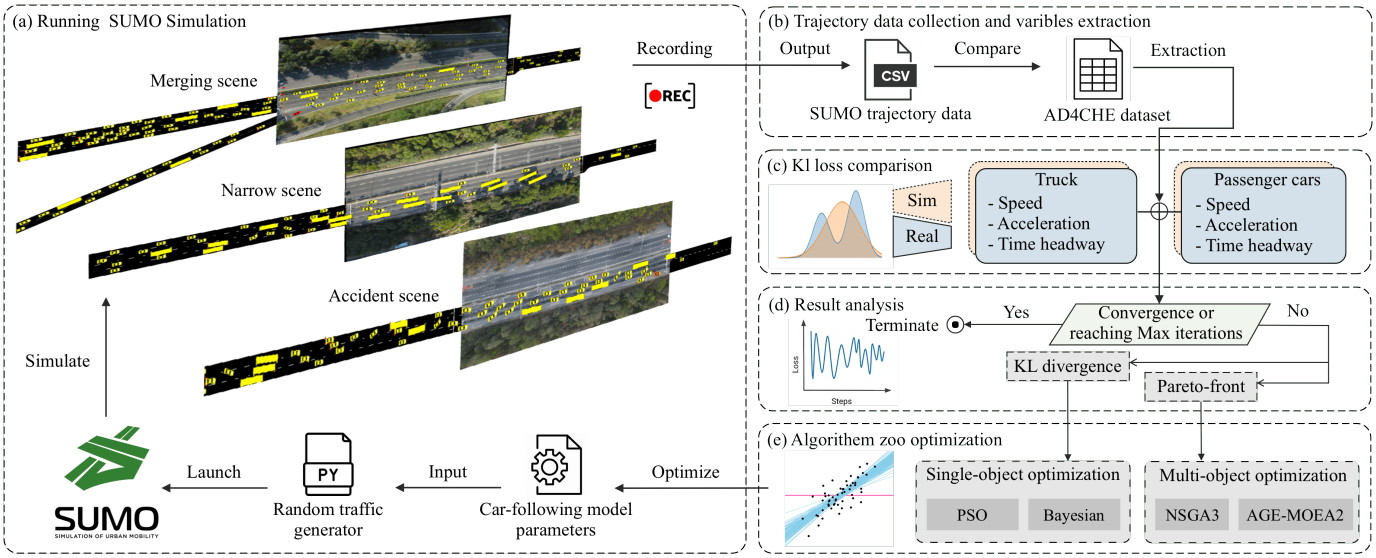


Fig. 2: Calibration framework of the trustworthy traffic simulation

where $p_i^{(m)}$ and $q_i^{(m)}$ are the probability distributions of the m-th feature from the simulated data and the real data, respectively. The set of KL divergences $\{D_{\mathrm{KL}}^{(1)}, D_{\mathrm{KL}}^{(2)}, \ldots, D_{\mathrm{KL}}^{(M)}\}$ serve as the objectives in the optimization task.

Our goal is to solve the following optimization problem:

$$\begin{aligned} \min_{\mathbf{x}} \quad & \mathbf{F}(\mathbf{x}) = [D_{\mathrm{KL}}^{(1)}, D_{\mathrm{KL}}^{(2)}, \ldots, D_{\mathrm{KL}}^{(M)}], \\ \text{s.t.} \quad & x_i^L \leq x_i \leq x_i^U, \quad i = 1, \ldots, N. \end{aligned} \tag{11}$$

where x_i^L and x_i^U are the lower and upper bounds for the i-th parameter. In step (d), result analysis, we compare the optimization outcomes with our predefined convergence criteria to determine whether to continue optimization iterations. When using multi-objective optimization, we can manually adjust the weights of each optimization objective or examine the Pareto front to understand the trade-offs between different objectives.

We utilize (e) state-of-the-art algorithms from [37] [38] to optimize a new set of parameters for running the simulations.

IV. Experiments and Results

This section details the experimental evaluation conducted to validate the performance of our proposed algorithm within realistic simulated traffic scenarios. We trained and evaluated our algorithm against seven baselines, followed by a comprehensive analysis of the experimental results, including an ablation study. The subsequent subsections elaborate on the experimental setup, traffic scenario generation, training procedures, and results analysis.

A. Simulation Environment

Adversarial congestion scenarios, including mandatory lane changes during merges, accidents, and lane narrowing, pose critical challenges to highway automation. These scenarios are major contributors to autonomous driving system failures [39], [40]. To evaluate our method's performance under such conditions, we utilized the AD4CHE dataset [41], which focuses on congestion on Chinese highways and urban highways.

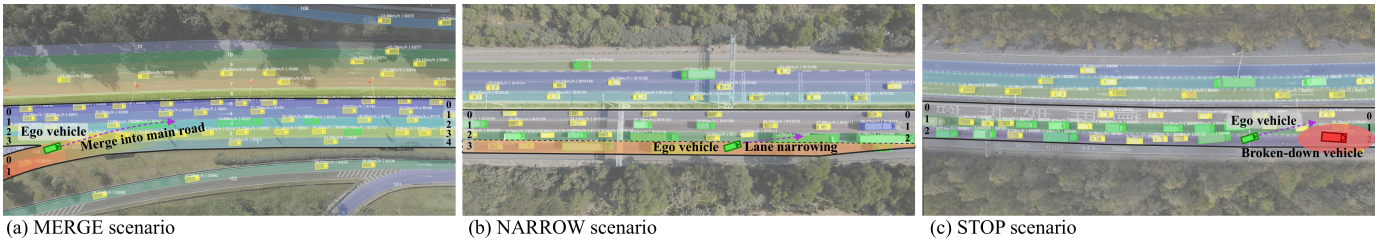
We selected three representative scenarios from AD4CHE's aerial data (5.12 hours total): (a) Merge scenario (files 1-8), where the ego vehicle merges from a two-lane road onto a four-lane highway, requiring conflict resolution in the merging zone; (b) Narrow scenario (files 18-23), where the ego vehicle must change lanes leftward before a road narrowing; and (c) Stop scenario (files 15-16), where the ego vehicle navigates around a broken-down vehicle in slow-moving traffic. These scenarios are illustrated in Fig. 3.

B. Trustworthy Traffic Scenario Generation

We utilize SUMO [42] (Simulation of Urban MObility) to create the simulation environment. Four optimization algorithms—PSO [43], Bayesian [44], NSGA3 [45], and AGE2 [46]—are employed to minimize the KL divergence between the aggregated real-world and simulated data for acceleration, velocity, and time headway. Each algorithm iterates 3000 times (population-based and evolutionary algorithms use 100 populations$\times$30 iterations to ensure fair results.)

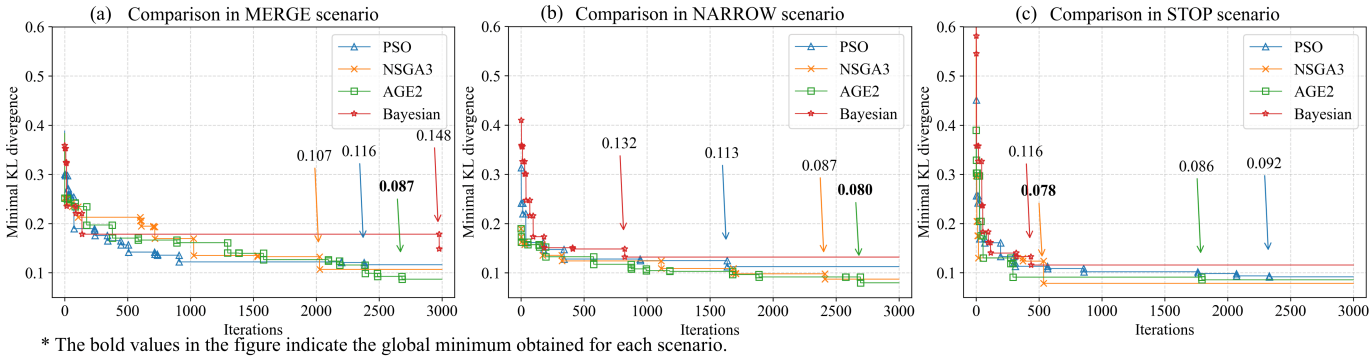
The calibrated aggregated traffic flow metrics include acceleration, velocity, and time headway distributions for both car and bus vehicle types. These calibration results are compared against both the real-world dataset and the original SUMO-generated metrics. The calibration process was done in a workstation equipped with 2$\times$AMD 5995WX CPU (128 cores total) and 256GB of RAM, required approximately 24 hours to complete 3000 iterations across three scenarios for four algorithms.

Fig. 4 shows the convergence comparison of the optimization objective in the three scenarios. The results show that all four optimization algorithms can reduce the error to a satisfactory level (mean KL divergence around 0.1). Single-objective algorithms (PSO, Bayesian) typically converge to the minimum value more quickly, while multi-objective algorithms (NSGA3, AGE2) ultimately achieve smaller values within a limited number of iterations.



(a) MERGE scenario (b) NARROW scenario (c) STOP scenario

Fig. 3: Three scenarios in the simulation environment: (a) MERGE: vehicle merging into the main lane from the ramp, (b) NARROW: vehicle navigating a narrowing right lane, (c) STOP: vehicle bypassing a broken-down vehicle.



* The bold values in the figure indicate the global minimum obtained for each scenario.

Fig. 4: Convergence comparison of optimization objective in (a) MERGE scenario, (b) NARROW scenario, (c) STOP scenario.

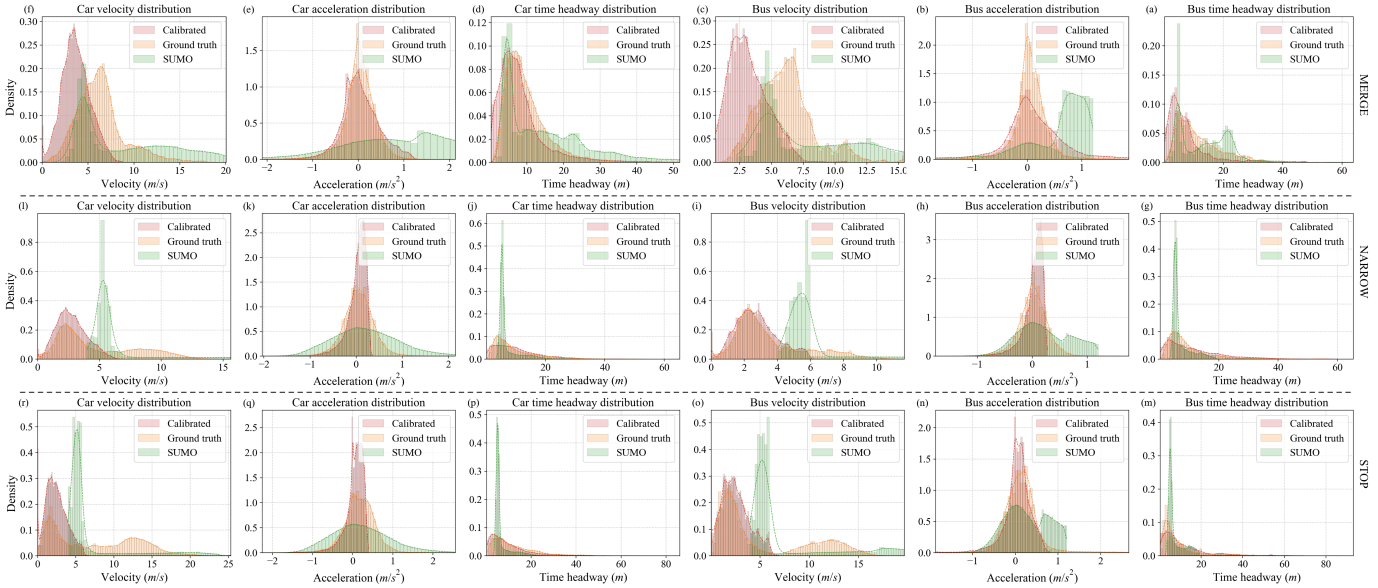


Fig. 5: Comparison of calibration results for velocity, acceleration, and time headway distributions across three scenarios for cars and buses

Fig. 5 presents the distributions of multiple calibrated traffic flow aggregation metrics, compared with both the real dataset and uncalibrated original SUMO-generated metrics. The results demonstrate that, except for a few scenarios shown in Fig. 5 (c) and (f), the calibrated metric distributions generally align with the true dataset in terms of central tendency, while the original SUMO-generated distributions typically exhibit wider dispersion. The time headway distributions in both real and calibrated data reveal a characteristic skewed pattern with peaks at smaller time intervals, indicating the presence of aggressive driving behaviors in the dataset. The original SUMO distributions shows significant deviations at larger time headways ($\geq$10 m), particularly underestimating the frequency of large time headways in complex scenarios such as MERGE and NARROW. Regarding acceleration, the calibrated distributions show a more pronounced peak near zero acceleration, closely matching the ground truth distribution, while the SUMO distribution exhibits a shifted peak. In the STOP scenario, SUMO severely underestimates the frequency of negative accelerations, whereas the calibrated model more accurately reflects actual braking behavior. Finally, the calibrated speed distributions effectively capture the bimodal characteristics (a high-frequency region and a more dispersed long-tail distribution), significantly outper-

forming SUMO. This improvement is particularly evident in the MERGE and STOP scenarios for cars, where SUMO notably overestimates speeds in the mid-range (e.g., $5\,\mathrm{m\,s^{-1}}$ to $10\,\mathrm{m\,s^{-1}}$) and underestimates the long-tail low-speed region. The calibrated model demonstrably corrects these deficiencies.

C. Agent Training and Evaluation Setup

To ensure experimental reproducibility, all agents were trained across five random seeds, each for 1000 episodes. Each episode was limited to 3000 time steps. Both training and evaluation were performed in trustworthy traffic scenarios, as described earlier in this paper. The experiments were conducted on a workstation equipped with dual NVIDIA RTX 3090 GPUs, an AMD Ryzen 9950X CPU, and 64 GB of RAM. The training process took approximately 12 hours. The following sections provide detailed information on the agent configuration.

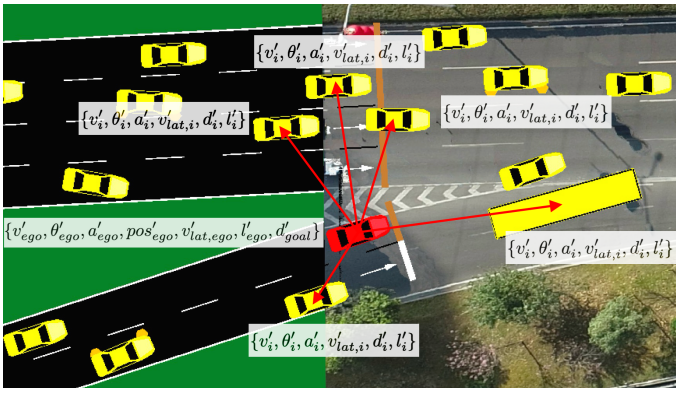


Fig. 6: Observation space for ego agent

Observation Space: The observation space $\mathbf{o}_t$ is updated at $10\,\mathrm{Hz}$ and provides the agent's perception of its surroundings. It is formed by concatenating the observation vectors of the five nearest vehicles within a $100\,\mathrm{m}$ detection range, together with the ego vehicle's own state as shown in Fig. 6. The observation space is defined as:

$$\mathbf{o}_t = [\mathbf{o}_{\mathrm{veh},i}]_{i=1,\ldots,5} \oplus \mathbf{o}_{\mathrm{ego}}. \tag{12}$$

For each of the five nearest vehicles ($i = 1, 2, \ldots, 5$), the observation vector is a 6-dimensional set of normalized parameters:

$$\mathbf{o}_{\mathrm{veh},i} = \{v'_{\mathrm{i}}, \theta'_{\mathrm{i}}, a'_{\mathrm{i}}, v'_{\mathrm{lat},i}, d'_{\mathrm{i}}, l'_{\mathrm{i}}\}, \tag{13}$$

where v'_{i}, θ'_{i}, a'_{i}, $v'_{\mathrm{lat},i}$, d'_{i}, and l'_{i} respectively represent the i-th vehicle's normalized speed, angle, acceleration, lateral speed, distance, and length.

The ego vehicle's observation vector consists of seven dimensions:

$$\mathbf{o}_{\mathrm{ego}} = \{v'_{\mathrm{ego}}, \theta'_{\mathrm{ego}}, a'_{\mathrm{ego}}, pos'_{\mathrm{ego}}, v'_{\mathrm{lat,ego}}, l'_{\mathrm{ego}}, d'_{\mathrm{goal}}\}, \tag{14}$$

incorporating the ego vehicle's normalized speed, angle, acceleration, lane position, lateral speed, length, and distance to goal.

Table II summarizes each variable in the observation space:

TABLE II
OBSERVATION VARIABLES AND THEIR NORMALIZATION.

Symbol	Description and Normalization
$v'_{\mathrm{i}}, v'_{\mathrm{ego}}$	Vehicle speed, normalized by $v_{\max}$
$\theta'_{\mathrm{i}}, \theta'_{\mathrm{ego}}$	Vehicle angle, normalized by $\theta_{\max}$
$a'_{\mathrm{i}}, a'_{\mathrm{ego}}$	Vehicle acceleration, normalized by $a_{\max}$
$v'_{\mathrm{lat},i}, v'_{\mathrm{lat,ego}}$	Lateral speed, normalized by $v_{\mathrm{lat,max}}$
d'_{i}	Distance to the i-th vehicle, normalized by d_{detect}
$l'_{\mathrm{i}}, l'_{\mathrm{ego}}$	Vehicle length, normalized by l_{norm}
pos'_{ego}	Ego lane position, normalized by d_{lane}
d'_{goal}	Distance to goal, normalized by $d_{goal,init}$

Action Space: The action space $\mathbf{a}_t$ is a two-dimensional vector:

$$\mathbf{a}_t = \{a_t, \delta_t\}, \tag{15}$$

where a_t represents the acceleration and δ_t the lateral movement. Both components are continuous values within the range $[-1, 1]$. When interacting with the simulator, the acceleration a_t is scaled by $a_{\max}$, while δ_t remains unchanged as defined by the SUMO interface.

Reward Function: The reward R is defined as Eq. 16:

$$\mathrm{R} = \alpha \cdot \frac{v_{\mathrm{ego}}}{v_{\max}} - \beta \cdot \left(\mathrm{C}_{\mathrm{collision}} + \mathrm{C}_{\mathrm{TTC}} + \mathrm{C}_{\mathrm{overtime}}\right) + \gamma \cdot \mathrm{P}_{\mathrm{nav}}, \tag{16}$$

where $\mathrm{C}_{\mathrm{collision}} = \left(\mathrm{C}_{\mathrm{base}} + v_{\mathrm{col}}\right)$ if a collision occurs and 0 otherwise. $\mathrm{C}_{\mathrm{TTC}}$ is defined as $\max\left(0, \mathrm{TTC}_{\min} - \mathrm{TTC}_{\mathrm{ego}}\right)/\mathrm{TTC}_{\min}$. Meanwhile, $\mathrm{C}_{\mathrm{overtime}} = \mathrm{C}_{\mathrm{base}}\,(1 - \mathrm{P}_{\mathrm{nav}})$ if the episode exceeds the time limit, and 0 otherwise. We measure progress as $\mathrm{P}_{\mathrm{nav}} = 1$ if $d_{\mathrm{goal}} < 30\,\mathrm{m}$, or $1 - d_{\mathrm{goal}}/d_{\mathrm{goal,init}}$ otherwise.

The weighting parameters α, β, and γ are set to 1, 1, and 0.1, respectively, based on a large-scale hyperparameter search. The base collision cost $\mathrm{C}_{\mathrm{base}}$ is 10, and the minimum Time-To-Collision threshold $\mathrm{TTC}_{\min}$ is $3\,\mathrm{s}$. Here, v_{ego} is the ego vehicle's mean speed, $v_{\max}$ its maximum speed, and v_{col} the ego vehicle's speed upon collision. The term $\mathrm{TTC}_{\mathrm{ego}}$ is the minimum Time-To-Collision with other vehicles during the episode, d_{goal} is the distance to the goal when terminated, and $d_{\mathrm{goal,init}}$ is the initial distance to the goal.

D. Benchmark and Metrics

To evaluate RiskDreamer, we benchmark its performance against advanced RL methods, categorized into following groups:

1) Established RL Methods: **Proximal Policy Optimization (PPO)** [47]: A widely used RL algorithm employing policy clipping for stable training. PPO is efficient, requires minimal hyperparameters, and generalizes well across continuous and discrete action spaces.

Soft Actor-Critic (SAC) [35]: An off-policy algorithm using maximum entropy policy gradients for enhanced exploration and sample efficiency. SAC exhibits stable convergence and robust performance, especially in continuous action spaces.

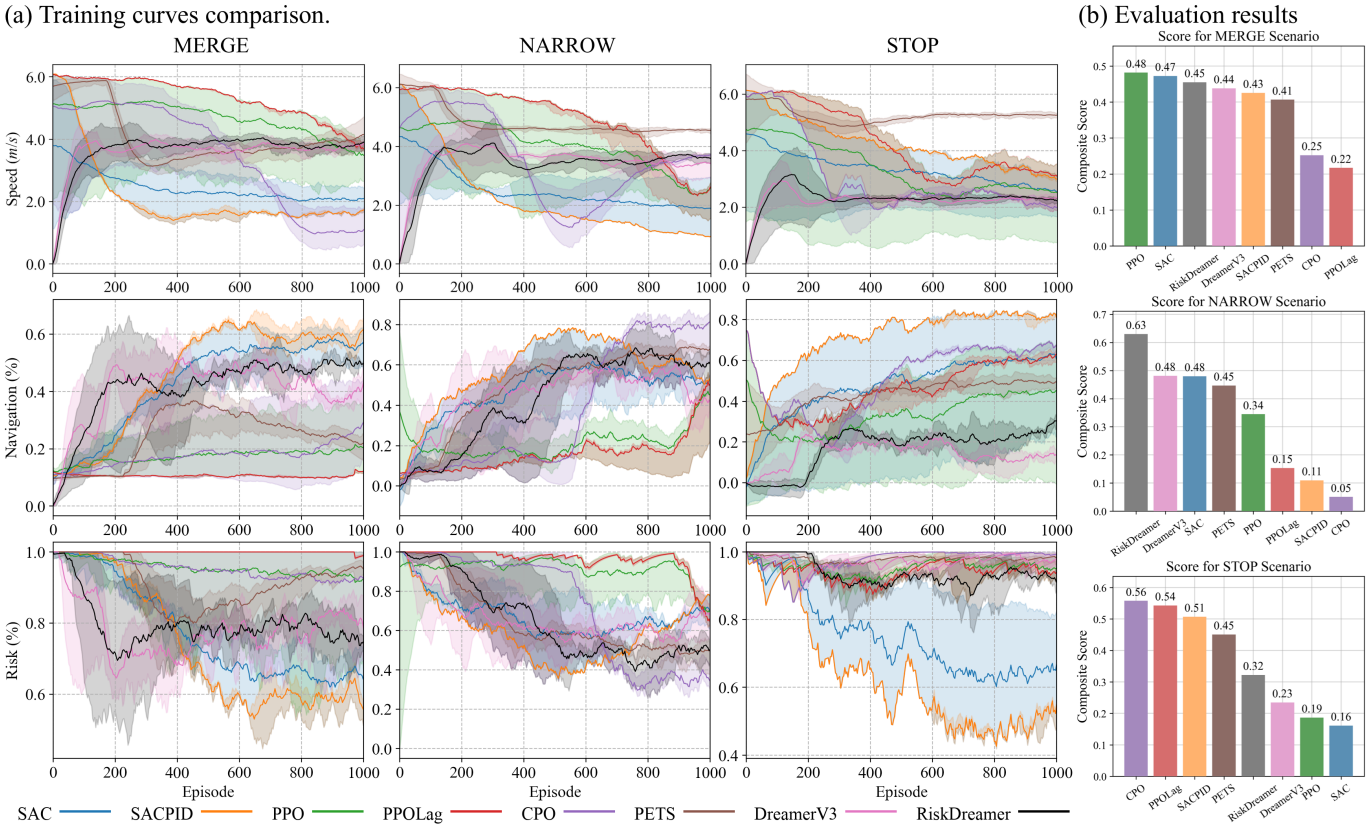


Fig. 7: Results of different algorithm in the three scenarios. (a) Training curves comparison. (b) Evaluation results

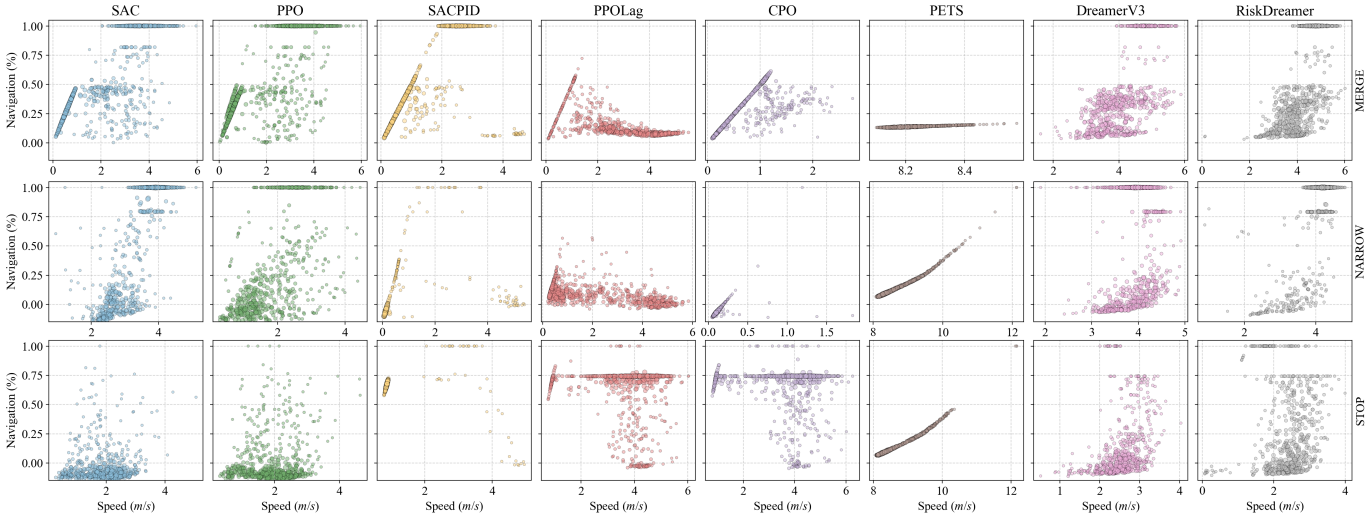


Fig. 8: Evaluation pattern of agent performance: Speed vs. Navigation. Bigger points indicate denser data point distributions.

2) Vanilla Safe RL Methods: **Conservative Policy Optimization (CPO)** [24]: A foundational trust region safe RL method. CPO enforces safety by limiting policy updates within a trust region, providing a conservative approach for risky environments.

PPO-Lagrangian (PPO-Lag) [47]: Extends PPO with a Lagrangian term to balance reward maximization and constraint satisfaction. Effective for constrained optimization.

Soft Actor-Critic with PID Controller (SACPID) [48]: Enhances SAC with a PID controller to mitigate oscillations in Lagrangian methods, leading to more stable constrained optimization.

3) Model-Based Safe RL Method: **Constrained Cross-Entropy Method for Policy Evolution on Trajectory Search (PETS)** [15]: A gradient-free, population-based method using cross-entropy optimization. PETS refines policies based on performance and constraint satisfaction via trajectory sampling.

4) SOTA World Model Method: **DreamerV3** [20]: State-of-the-art multi-task world model. DreamerV3 learns environ-

TABLE III
PERFORMANCE COMPARISON OF DIFFERENT METHODS IN MULTIPLE SCENARIOS.

Method	Metric	MERGE Scenario				NARROW Scenario				STOP Scenario			
		Mean	.Std	ΔValue	Δ%	Mean	.Std	ΔValue	Δ%	Mean	.Std	ΔValue	Δ%
SAC	Navigation	0.542	0.337	+0.157	↑40.9%	0.511	0.423	+0.196	↑62.3%	0.130	0.133	-0.227	↓63.6%
	Mean Speed	2.467	1.419	-0.836	↓25.3%	3.252	0.726	+0.145	↑4.7%	1.872	0.735	-1.350	↓41.9%
	Risk	0.448	0.498	-0.147	↓24.7%	0.644	0.479	+0.084	↑14.9%	0.998	0.045	+0.168	↑20.2%
	Score	0.472	0.487	+0.079	↑20.0%	0.480	0.368	+0.143	↑42.4%	0.161	0.239	-0.209	↓56.5%
PPO	Navigation	**0.560**	0.352	+0.175	↑45.4%	0.386	0.384	+0.071	↑22.5%	0.165	0.177	-0.192	↓53.8%
	Mean Speed	2.389	1.418	-0.915	↓27.7%	1.988	0.905	-1.119	↓36.0%	1.898	0.746	-1.324	↓41.1%
	Risk	0.400	0.490	-0.195	↓32.7%	0.734	0.442	+0.174	↑31.0%	0.992	0.089	+0.162	↑19.5%
	Score	**0.482**	0.492	+0.088	↑22.4%	0.345	0.382	+0.008	↑2.3%	0.187	0.256	-0.184	↓49.6%
SACPID	Navigation	0.521	0.360	+0.136	↑35.4%	0.125	0.197	-0.190	↓60.2%	**0.665**	0.118	+0.309	↑86.6%
	Mean Speed	1.604	1.149	-1.699	↓51.4%	0.584	1.235	-2.523	↓81.2%	1.109	0.787	-2.112	↓65.6%
	Risk	**0.210**	0.408	-0.385	↓64.7%	0.114	0.318	-0.446	↓79.7%	**0.052**	0.222	-0.778	↓93.7%
	Score	0.425	0.427	+0.032	↑8.1%	0.110	0.395	-0.227	↓67.5%	0.507	0.250	+0.137	↑37.0%
PPOLag	Navigation	0.171	0.139	-0.214	↓55.7%	0.095	0.093	-0.220	↓70.0%	0.583	0.253	+0.226	↑63.5%
	Mean Speed	2.602	1.446	-0.701	↓21.2%	2.333	1.716	-0.774	↓24.9%	3.597	1.238	+0.376	↑11.7%
	Risk	0.804	0.397	+0.209	↑35.2%	0.878	0.328	+0.318	↑56.7%	0.888	0.316	+0.058	↑7.0%
	Score	0.217	0.445	-0.176	↓44.8%	0.154	0.519	-0.183	↓54.4%	0.543	0.412	+0.172	↑46.6%
CPO	Navigation	0.313	0.163	-0.072	↓18.6%	0.065	0.049	-0.250	↓79.2%	0.607	0.240	+0.251	↑70.3%
	Mean Speed	0.876	0.512	-2.428	↓73.5%	0.129	0.123	-2.978	↓95.9%	3.538	1.364	+0.316	↑9.8%
	Risk	0.316	0.465	-0.279	↓46.9%	**0.032**	0.176	-0.528	↓94.3%	0.858	0.349	+0.028	↑3.3%
	Score	0.252	0.191	-0.141	↓35.9%	0.051	0.051	-0.286	↓85.0%	**0.558**	0.442	+0.187	↑50.6%
PETS	Navigation	0.139	0.006	-0.246	↓63.8%	0.166	0.116	-0.149	↓47.2%	0.171	0.118	-0.185	↓52.0%
	Mean Speed	**8.236**	0.082	+4.932	↑149.3%	**8.810**	0.642	+5.703	↑183.5%	**8.818**	0.663	+5.597	↑173.7%
	Risk	1.000	0.000	+0.405	↑68.1%	0.996	0.063	+0.436	↑77.8%	0.994	0.077	+0.164	↑19.7%
	Score	0.406	0.025	+0.013	↑3.3%	0.447	0.209	+0.110	↑32.6%	0.451	0.216	+0.080	↑21.7%
DreamerV3	Navigation	0.404	0.323	+0.019	↑4.9%	0.479	0.439	+0.164	↑52.0%	0.199	0.260	-0.158	↓44.2%
	Mean Speed	4.136	0.766	+0.832	↑25.2%	3.893	0.457	+0.786	↑25.3%	2.554	0.501	-0.668	↓20.7%
	Risk	0.810	0.393	+0.215	↑36.2%	0.640	0.480	+0.080	↑14.2%	0.962	0.191	+0.132	↑15.9%
	Score	0.438	0.323	+0.044	↑11.3%	0.481	0.336	+0.144	↑42.7%	0.235	0.236	-0.135	↓36.6%
RiskDreamer **(Ours)**	Navigation	0.429	0.349	+0.044	↑11.5%	**0.692**	0.405	+0.377	↑119.8%	0.332	0.330	-0.024	↓6.8%
	Mean Speed	4.120	0.832	+0.816	↑24.7%	3.867	0.601	+0.760	↑24.5%	2.386	0.638	-0.836	↓25.9%
	Risk	0.770	0.421	+0.175	↑29.5%	0.444	0.497	-0.116	↓20.7%	0.898	0.303	+0.068	↑8.2%
	Score	0.455	0.349	+0.061	↑15.6%	**0.630**	0.336	+0.293	↑86.8%	0.322	0.300	-0.048	↓13.1%

ment dynamics and improves policies through imagination, enabling robust learning across diverse environments.

The performance of each algorithm was evaluated across MERGE, NARROW, and STOP scenarios. The evaluation metrics are defined as follows:

1) **Navigation:** Percentage of navigation distance covered. Higher value indicates better route following.
2) **Mean Speed:** Average speed of the ego vehicle during the episode. Reflects driving efficiency.
3) **Risk:** Average risk across episodes (0: no collision, 1: collision).
4) **Composite Score:** $0.7 \times \text{Navigation} + 0.3 \times \frac{\text{Mean Speed}}{\text{Max Speed}}$.

E. Results Analysis

This section presents the results of the training and evaluation of the RiskDreamer and the benchmarked methods. Fig. 7 (a) presents the training curves for key performance indicators. Fig. 7 (b) summarizes the composite scores achieved by each algorithm in each scenario. Table III provides detailed quantitative results for each metric across all scenarios and algorithms. Fig. 8 shows the distribution of evaluation scores in Speed vs. Navigation coordinates.

Analysis of Training Curves: As shown in Fig. 7 (a), in the MERGE scenario, RiskDreamer demonstrates stable improvements in navigation success rate, with its risk level gradually decreasing and stabilizing. The baseline DreamerV3 initially performs similarly to RiskDreamer, but its performance declines as training continues. On the other hand, SAC and SACPID excel at reducing risk but demonstrate conservative training results in their actions, leading to slow speeds.

In the NARROW scenario, the navigation difficulty is reduced. While maintaining its speed, RiskDreamer shows con-

tinuous improvement in navigation success rate. The baseline DreamerV3 still experiences performance degradation in the later stages of training. PPO and SAC show improvement in navigation but maintain low speeds. However, they often exhibit high risk levels. PETS maintains high speed but encounters difficulties in navigation and maintains high risk. CPO sometimes achieves low risk and high speed, but training results fluctuate significantly.

In the STOP scenario, navigation difficulty is highest. RiskDreamer demonstrates speed reduction and moderate improvement in navigation success rate. Although there is improvement compared to the baseline DreamerV3, it performs mediocrely in this scenario. SAC and SACPID show rapid initial risk reduction, followed by stabilization at a higher level. PETS initially achieves high speed but cannot park reliably, resulting in sustained high risk.

Analysis of Evaluation Results: As shown in Fig. 7 (b), the composite score provides an overall performance measure that balances speed, navigation, and risk. In the MERGE scenario, PPO, SAC, and RiskDreamer achieve comparable composite scores, indicating a balance between different objectives. PETS and PPOLag show lower scores due to the trade-offs between speed and safety.

In the NARROW scenario, RiskDreamer achieves the highest composite score, significantly outperforming other algorithms. Compared to DreamerV3, it shows a $31.25\,\%$ improvement, though DreamerV3 also achieves a competitive score. CPO again shows lower scores, highlighting its difficulty in navigating confined spaces while maintaining safety. CPO and PPOLag show negative composite scores, indicating poor overall performance.

In the STOP scenario, CPO achieves the highest composite score, reflecting its ability to prioritize safety. PPOLag and SACPID also achieve relatively high scores. RiskDreamer demonstrates a moderate composite score, showing a $29.13\,\%$ improvement compared to the baseline DreamerV3, though neither performs at the top of the rankings.

Quantitative Analysis: The quantitative results in table III confirm the observations from the charts. RiskDreamer performs well in the MERGE scenario, achieving an average speed ($4.12\ \mathrm{m/s}$) and a respectable navigation score (0.43). While its risk is moderate (0.77), its final composite score (0.426) is competitive. In the NARROW scenario, RiskDreamer performs the best, achieving a high average speed ($3.87\ \mathrm{m/s}$) and the highest navigation score (0.69), leading to the highest composite score (0.63).

In the STOP scenario, RiskDreamer exhibits a lower average speed ($2.39\ \mathrm{m/s}$) but maintains a reasonable navigation score (0.33) and a relatively high risk (0.90), resulting in a lower composite score (0.30) compared to safety-focused algorithms like CPO and PPOLag. PETS consistently achieves the highest average speeds across all scenarios but exhibits extremely low navigation scores and near-perfect risk, leading to low composite scores, indicating its failure to effectively navigate and prioritize safety. Conversely, CPO and PPOLag tend towards lower speeds but prioritize safety, achieving high risk scores in the STOP scenario but lower composite scores in the MERGE and NARROW scenarios due to their conservative behavior.

Evaluation Pattern Analysis: Fig. 8 provides a visual representation of the evaluation results in Speed vs. Navigation coordinates. Smaller scatter points represent denser sampling regions, reflecting the distribution of algorithm performance at varying speeds and navigation success rates. There are noticeable differences in the point cloud distributions of each algorithm across different scenarios. The distributions of SAC and PPO are similar: there's a higher concentration of points in the regions with lower navigation success rates (representing early collisions or failures), while there's also a linear distribution of points clustered when the navigation success rate approaches 1. This suggests that they either tend to fail early on or can successfully achieve the final goal. DreamerV3 and RiskDreamer also exhibit similar trends, but their point distribution in the medium navigation range is slightly elevated, and as speed increases, navigation completion also increases accordingly. Compared to the baseline algorithm DreamerV3, RiskDreamer achieves a linear distribution at high success rates at a faster speed, indicating it possesses a certain advantage in balancing risk and efficiency.

In the context of safe reinforcement learning, methods such as SACPID, PPOLag and CPO, their distribution areas in the figure are observed to be generally lifted towards the middle. This implies that a relative balance has been achieved between speed and success rate, but the number of full navigations (success rate of 1) has decreased, indicating that these algorithms sacrifice some extreme optimal performance while improving safety. It's worth noting that, although PETS has a noticeable distribution in the high-speed region, it can hardly complete navigation, and its performance is severely limited.

Discussion: RiskDreamer demonstrates the ability to adapt its behavior across different scenarios. In the MERGE and NARROW scenarios, it strikes a good balance between speed and navigation while maintaining reasonable risk levels. While its composite score in the STOP scenario is inferior compared to safety-focused RL algorithms, it still outperforms the baseline DreamerV3. The results suggest that RiskDreamer avoids the overly conservative behavior of some safety RL algorithms and the reckless behavior of purely performance-driven approaches. The ablation study in the next section will further explore the effectiveness of RiskDreamer's dynamic risk-entropy balancing mechanism.

F. Ablation Study

To evaluate the contribution of individual components within the RiskDreamer framework, we conducted a series of ablation studies. These studies specifically assessed the impact of the entropy and risk-balancing mechanisms, as well as the effect of the online planning horizon on the batch planning process.

Impact of Entropy and Risk Balancing: Table IV presents two ablated versions: one using only the entropy balancing term in the action expansion and the other using only the risk balancing term. The full RiskDreamer model achieves the highest overall score (0.63), outperforming both ablated versions and the baseline DreamerV3. RiskDreamer demonstrates

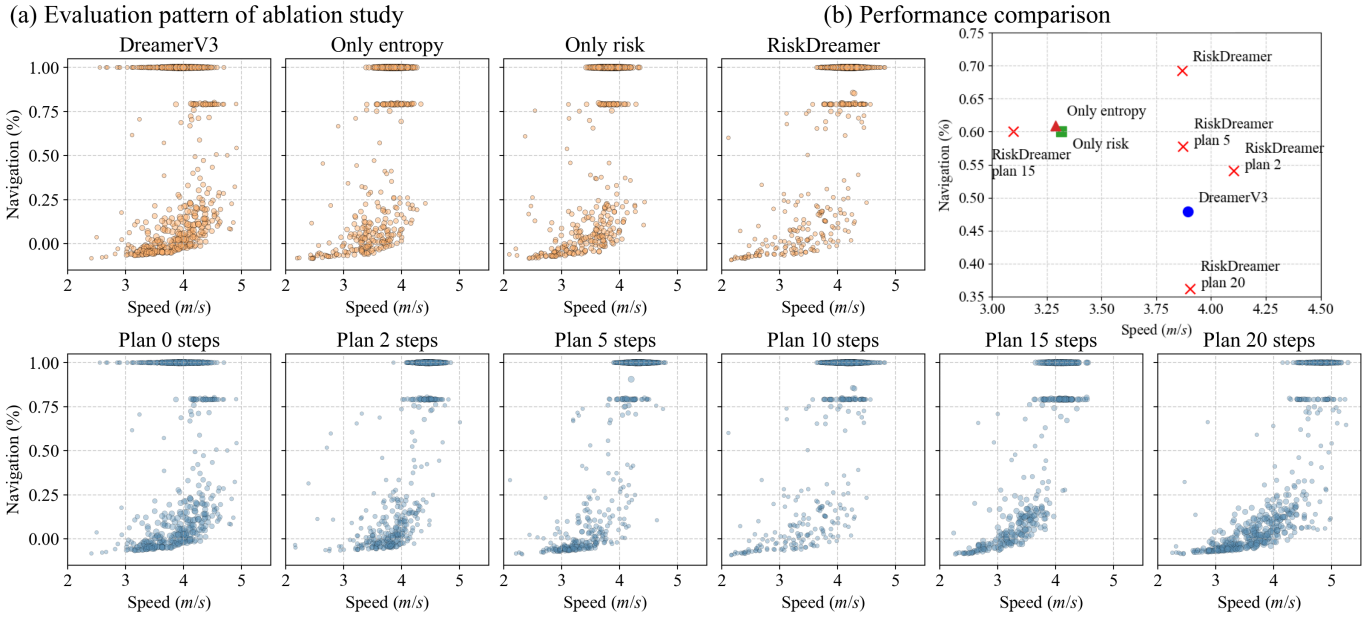


Fig. 9: Results of ablation study (a) Evaluation pattern (b) Performance comparison

TABLE IV
IMPACT OF ENTROPY AND RISK BALANCING

Algorithm	Speed (Std.)	Navigation (Std.)	Risk (Std.)	Score
RiskDreamer	3.87 (0.60)	**0.69** (0.40)	**0.44** (0.50)	**0.63**
Only entropy term	3.29 (0.95)	0.61 (0.42)	0.63 (0.48)	0.55
Only risk term	3.32 (0.90)	0.60 (0.43)	0.61 (0.49)	0.54
DreamerV3	**3.89** (0.46)	0.48 (0.44)	0.64 (0.48)	0.48

TABLE V
IMPACT OF THE NUMBER OF PLANNING BRANCHES

Horizon (H)	Speed (Std.)	Navigation (Std.)	Risk (Std.)	Score
H = **10 (Chosen)**	3.87 (0.60)	**0.69** (0.40)	**0.44** (0.50)	**0.63**
H = 5	3.87 (0.59)	0.58 (0.44)	0.54 (0.50)	0.55
H = 15	3.10 (1.15)	0.60 (0.41)	0.75 (0.43)	0.54
H = 2	**4.10** (0.50)	0.54 (0.44)	0.60 (0.49)	0.53
H = 0	3.89 (0.46)	0.48 (0.44)	0.64 (0.48)	0.48
H = 20	3.90 (0.87)	0.36 (0.40)	0.80 (0.40)	0.40

superior performance in Navigation (0.69) and Risk (0.44), indicating a more effective balance between task completion and risk mitigation. Although the ablated versions with only entropy or only risk balancing show slightly improved navigation score compared to DreamerV3, they still underperform relative to RiskDreamer and exhibit higher risk values (0.63 and 0.61, respectively). DreamerV3, which does not include explicit entropy or risk balancing, achieves the highest Mean Speed (3.89) but performs poorly in Navigation (0.48) and incurs the highest Risk (0.64), resulting in the lowest overall Score (0.48). Fig. 9 (a) shows that using only the entropy or risk term reduces the low-speed portion of completed navigation clusters, whereas the full RiskDreamer implementation increases the high-speed portion of these clusters. Fig. 9 (b) illustrates that both single-term versions produce comparable outcomes by decreasing speed while improving navigation success.

Influence of Planning horizon length: We further investigated the impact of the length of planning horizons employed in our batch planning approach. Table V shows RiskDreamer's performance under different planning horizon lengths, ranging from 0 (equivalent to DreamerV3's single-step action selection) to 20. The results indicate that the length of planning horizon significantly affects performance. RiskDreamer with a planning length of 10, our chosen configuration, achieved the best overall Score (0.63), Navigation (0.69), and Risk (0.44). Reducing the planning length to 5 still maintains a competitive Mean Speed (3.87) but leads to a noticeable decrease in Navigation (0.58) and an increase in Risk (0.54), resulting in a lower Score (0.55). Conversely, increasing the planning length beyond 10, such as to 15 and 20, results in degradation of all metrics except for Mean Speed in the case of plan = 20. Specifically, with planning lengths of 15 and 20, we observe a significant increase in Risk (0.75 and 0.80 respectively) and a substantial drop in Navigation (0.60 and 0.36 respectively), leading to lower Scores (0.54 and 0.40). Interestingly, while a very short planning length of 2 leads to the highest Mean Speed (4.10), it also results in compromised Navigation (0.54) and Risk (0.60), and a lower Score (0.53) compared to plan = 10. The case with plan length 0, which is equivalent to acting without planning (similar to DreamerV3), yields the lowest Score (0.48). As shown in Fig. 9 (a) Evaluation pattern, even plan 2 steps can reduce linear clustering in the low-speed region, with bottom collision clustering being an exception. Furthermore, except for step 15, the model's performance speed tends to increase as steps increase. Fig. 9 (b) Performance comparison shows increased navigation for all except plan 20.

V. Conclusion

In this paper, we introduced RiskDreamer, a novel model-based reinforcement learning framework that leverages batch planning in a latent space to dynamically balance risk, entropy, and expected reward. By extending the action space to output these balancing factors directly, our approach addresses the limitations of fixed-parameter risk allocation and overcomes the shortcomings of overly simplified or homogeneous traffic simulations. The trustworthy traffic scenario generation framework further reinforces the realism of the simulation environment by calibrating diverse traffic agents' behaviors to match real-world statistics.

A. Evaluation Results

The experimental evaluations across three typical autonomous driving scenarios demonstrate the effectiveness of the proposed RiskDreamer framework. In the MERGE scenario, RiskDreamer achieves a competitive composite score by balancing speed, navigation success, and risk, showcasing its ability to navigate complex merging situations without being overly conservative or reckless. The NARROW scenario further highlights RiskDreamer's capability in challenging environments, where it achieves the highest navigation score and a strong overall composite score, indicating effective maneuvering in constrained spaces. While in the STOP scenario, safety-focused algorithms achieved higher composite scores due to their emphasis on minimizing risk, RiskDreamer still demonstrated its improvements against baseline DreamerV3. Compared to baseline methods, RiskDreamer exhibits a more adaptive behavior, avoiding the high-risk tendencies of purely performance-driven approaches like PETS and the overly cautious nature of some SafeRL methods such as CPO and PPOLag in dynamic scenarios. The quantitative analysis further supports these observations, confirming RiskDreamer's ability to dynamically adjust its behavior based on the specific scenario demands.

B. Limitations and Future Work

Despite RiskDreamer's promising performance, there are several limitations and future research directions need to be taken into consideration. Firstly, due to computational constraints, our current implementation chooses a state-based environment representation. Future work could explore richer, more realistic perception by incorporating vision-based or multi-modal sensory inputs, enhancing the agent's understanding of complex traffic scenarios. Secondly, while our trustworthy traffic scenario generator improves realism, SUMO remains a simplified traffic simulation environment with a relatively limited dataset. Future research could integrate larger-scale, real-world trajectory datasets and explore more advanced simulators such as CARLA to further enhance the fidelity of the training and evaluation environment. Finally, to more comprehensively validate RiskDreamer's real-world applicability and human-vehicle interaction, future studies should incorporate human-in-the-loop evaluations using driving simulators, bridging the gap between simulation and real-world driving scenarios.

References

[1] Q. Liu, C. Li, H. Jiang, S. Nie, and L. Chen, "Transfer learning-based highway crash risk evaluation considering manifold characteristics of traffic flow," *Accident Analysis & Prevention*, vol. 168, p. 106598, 2022.

[2] Q. Liu, R. Yu, Y. Cai, and L. Chen, "Studying the predictability of crash risk caused by manual takeover of autonomous vehicles in mixed traffic flow," *Transportation Letters*, vol. 16, no. 10, pp. 1205–1223, 2024.

[3] Q. Liu, J. Liu, Y. Cai, and L. Chen, "Exploring the impact of the takeover time for conditionally automated driving vehicles on traffic flow in highway merging area," *IEEE transactions on intelligent transportation systems*, vol. 23, no. 12, pp. 24 753–24 764, 2022.

[4] Q. Liu, F. Gao, J. Zhao, Y. Cai, L. Chen, and C. Lv, "Real-time lng buses emissions prediction based on a temporal fusion trans-formers model." *Journal of Environmental Informatics*, vol. 44, no. 1, 2024.

[5] Q. Liu, C. Gao, H. Wang, Y. Cai, L. Chen, and C. Lv, "Learning from trajectories: How heterogeneous cacc platoons affect the traffic flow in highway merging area," *IEEE Transactions on Vehicular Technology*, 2024.

[6] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. Van Den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot *et al.*, "Mastering the game of go with deep neural networks and tree search," *nature*, vol. 529, no. 7587, pp. 484–489, 2016.

[7] O. Vinyals, I. Babuschkin, W. M. Czarnecki, M. Mathieu, A. Dudzik, J. Chung, D. H. Choi, R. Powell, T. Ewalds, P. Georgiev *et al.*, "Grandmaster level in starcraft ii using multi-agent reinforcement learning," *nature*, vol. 575, no. 7782, pp. 350–354, 2019.

[8] J. Schrittwieser, I. Antonoglou, T. Hubert, K. Simonyan, L. Sifre, S. Schmitt, A. Guez, E. Lockhart, D. Hassabis, T. Graepel *et al.*, "Mastering atari, go, chess and shogi by planning with a learned model," *Nature*, vol. 588, no. 7839, pp. 604–609, 2020.

[9] Y. Zhang, Q. Vuong, and K. W. Ross, "First order constrained optimization in policy space," in *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin, Eds., 2020. [Online]. Available: https://proceedings.neurips.cc/paper/2020/hash/af5d5ef24881f3c3049a7b9bfe74d58b-Abstract.html

[10] H. J. Wabersich, L. Hewing, A. Carron, and M. N. Zeilinger, "Probabilistic model predictive safety certification for learning-based control," *IEEE Transactions on Automatic Control*, vol. 67, no. 1, p. 176–188, 2022. [Online]. Available: http://dx.doi.org/10.1109/tac.2021.3049335

[11] T. He, W. Zhao, and C. Liu, "Autocost: Evolving intrinsic cost for zero-violation reinforcement learning," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 37, no. 12, 2023, pp. 14 847–14 855.

[12] Z. Liu, H. Zhou, B. Chen, S. Zhong, M. Hebert, and D. Zhao, "Constrained model-based reinforcement learning with robust cross-entropy method," *ArXiv preprint*, vol. abs/2010.07968, 2020. [Online]. Available: https://arxiv.org/abs/2010.07968

[13] G. Williams, P. Drews, B. Goldfain, J. M. Rehg, and E. A. Theodorou, "Aggressive driving with model predictive path integral control," in *2016 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2016, pp. 1433–1440.

[14] M. Kotb, C. Weber, and S. Wermter, "Sample-efficient real-time planning with curiosity cross-entropy method and contrastive learning," in *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2023, pp. 9456–9463.

[15] M. Wen and U. Topcu, "Constrained cross-entropy method for safe reinforcement learning," in *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada*, S. Bengio, H. M. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, Eds., 2018, pp. 7461–7471. [Online]. Available: https://proceedings.neurips.cc/paper/2018/hash/34ffeb359a192eb8174b6854643cc046-Abstract.html

[16] H. Gao, X. Zheng, Q. Liu, L. Zhou, C. Huang, M. Hu, C. Wang, K. Li, D. Wang, and D. Li, "A spatial–temporal predictive transformer network for level-3 autonomous vehicle decision-making," *IEEE Transactions on Neural Networks and Learning Systems*, 2024.

[17] H. Caesar, J. Kabzan, K. S. Tan, W. K. Fong, E. Wolff, A. Lang, L. Fletcher, O. Beijbom, and S. Omari, "nuplan: A closed-loop ml-based planning benchmark for autonomous vehicles," *arXiv preprint arXiv:2106.11810*, 2021.

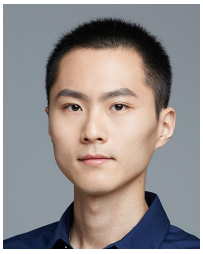
[18] S. Ettinger, S. Cheng, B. Caine, C. Liu, H. Zhao, S. Pradhan, Y. Chai, B. Sapp, C. R. Qi, Y. Zhou, Z. Yang, A. Chouard, P. Sun, J. Ngiam, V. Vasudevan, A. McCauley, J. Shlens, and D. Anguelov, "Large scale interactive motion forecasting for autonomous driving: The waymo open motion dataset," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2021, pp. 9710–9719.
[19] X. He, J. Wu, Z. Huang, Z. Hu, J. Wang, A. Sangiovanni-Vincentelli, and C. Lv, "Fear-neuro-inspired reinforcement learning for safe autonomous driving," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 46, no. 1, pp. 267–279, 2024.
[20] D. Hafner, J. Pasukonis, J. Ba, and T. Lillicrap, "Mastering diverse domains through world models," *ArXiv preprint*, vol. abs/2301.04104, 2023. [Online]. Available: https://arxiv.org/abs/2301.04104
[21] E. Altman, *Constrained Markov decision processes*. Routledge, 2021.
[22] R. T. Rockafellar, *Conjugate duality and optimization*. SIAM, 1974.
[23] E. Altman, "Constrained markov decision processes with total cost criteria: Lagrangian approach and dual linear program," *Mathematical methods of operations research*, vol. 48, pp. 387–417, 1998.
[24] J. Achiam, D. Held, A. Tamar, and P. Abbeel, "Constrained policy optimization," in *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, ser. Proceedings of Machine Learning Research, D. Precup and Y. W. Teh, Eds., vol. 70. PMLR, 2017, pp. 22–31. [Online]. Available: http://proceedings.mlr.press/v70/achiam17a.html
[25] A. Stooke, J. Achiam, and P. Abbeel, "Responsive safety in reinforcement learning by PID lagrangian methods," in *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, ser. Proceedings of Machine Learning Research, vol. 119. PMLR, 2020, pp. 9133–9143. [Online]. Available: http://proceedings.mlr.press/v119/stooke20a.html
[26] D. Ha and J. Schmidhuber, "World models," *ArXiv preprint*, vol. abs/1803.10122, 2018. [Online]. Available: https://arxiv.org/abs/1803.10122
[27] D. Hafner, T. P. Lillicrap, J. Ba, and M. Norouzi, "Dream to control: Learning behaviors by latent imagination," in *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020. [Online]. Available: https://openreview.net/forum?id=S1lOTC4tDS
[28] D. Hafner, T. Lillicrap, M. Norouzi, and J. Ba, "Mastering atari with discrete world models," *ArXiv preprint*, vol. abs/2010.02193, 2020. [Online]. Available: https://arxiv.org/abs/2010.02193
[29] D. Hafner, T. P. Lillicrap, I. Fischer, R. Villegas, D. Ha, H. Lee, and J. Davidson, "Learning latent dynamics for planning from pixels," in *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, ser. Proceedings of Machine Learning Research, K. Chaudhuri and R. Salakhutdinov, Eds., vol. 97. PMLR, 2019, pp. 2555–2565. [Online]. Available: http://proceedings.mlr.press/v97/hafner19a.html
[30] W. Huang, J. Ji, C. Xia, B. Zhang, and Y. Yang, "Safedreamer: Safe reinforcement learning with world models," in *The Twelfth International Conference on Learning Representations*, 2024. [Online]. Available: https://openreview.net/forum?id=tsE5HLYtYg
[31] Z. Gao, Y. Mu, C. Chen, J. Duan, P. Luo, Y. Lu, and S. E. Li, "Enhance sample efficiency and robustness of end-to-end urban autonomous driving via semantic masked world model," *IEEE Transactions on Intelligent Transportation Systems*, 2024.
[32] M. Noaeen, A. Naik, L. Goodman, J. Crebo, T. Abrar, Z. S. H. Abad, A. L. Bazzan, and B. Far, "Reinforcement learning in urban network traffic signal control: A systematic literature review," *Expert Systems with Applications*, vol. 199, p. 116830, 2022.
[33] M. Schrader and J. Bittle, "A global sensitivity analysis of traffic microsimulation input parameters on performance metrics," *IEEE Transactions on Intelligent Transportation Systems*, vol. 25, no. 9, pp. 11 739–11 752, 2024.
[34] X. Yan, Z. Zou, S. Feng, H. Zhu, H. Sun, and H. X. Liu, "Learning naturalistic driving environment with statistical realism," *Nature communications*, vol. 14, no. 1, p. 2037, 2023.
[35] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor," in *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, ser. Proceedings of Machine Learning Research, J. G. Dy and A. Krause, Eds., vol. 80. PMLR, 2018, pp. 1856–1865. [Online]. Available: http://proceedings.mlr.press/v80/haarnoja18b.html
[36] D. Salles, S. Kaufmann, and H.-C. Reuss, "Extending the intelligent driver model in sumo and verifying the drive off trajectories with aerial measurements," *SUMO Conference Proceedings*, vol. 1, p. 1–25, Jul. 2022. [Online]. Available: https://www.tib-op.org/ojs/index.php/scp/article/view/95
[37] N. Stander and K. Craig, "On the robustness of a simple domain reduction scheme for simulation-based optimization," *International Journal for Computer-Aided Engineering and Software (Eng. Comput.)*, vol. 19, 06 2002.
[38] J. Blank and K. Deb, "pymoo: Multi-objective optimization in python," *IEEE Access*, vol. 8, pp. 89 497–89 509, 2020.
[39] A. Sinha, S. Chand, V. Vu, H. Chen, and V. Dixit, "Crash and disengagement data of autonomous vehicles on public roads in california," *Scientific data*, vol. 8, no. 1, p. 298, 2021.
[40] S. Nordhoff, "A conceptual framework for automation disengagements," *Scientific Reports*, vol. 14, no. 1, p. 8654, 2024.
[41] Y. Zhang, C. Wang, R. Yu, L. Wang, W. Quan, Y. Gao, and P. Li, "The ad4che dataset and its application in typical congestion scenarios of traffic jam pilot systems," *IEEE Transactions on Intelligent Vehicles*, vol. 8, no. 5, pp. 3312–3323, 2023.
[42] P. A. Lopez, M. Behrisch, L. Bieker-Walz, J. Erdmann, Y.-P. Flötteröd, R. Hilbrich, L. Lücken, J. Rummel, P. Wagner, and E. Wießner, "Microscopic traffic simulation using sumo," in *The 21st IEEE International Conference on Intelligent Transportation Systems*. IEEE, 2018. [Online]. Available: https://elib.dlr.de/124092/
[43] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Proceedings of ICNN'95-international conference on neural networks*, vol. 4. ieee, 1995, pp. 1942–1948.
[44] J. Snoek, H. Larochelle, and R. P. Adams, "Practical bayesian optimization of machine learning algorithms," *Advances in neural information processing systems*, vol. 25, 2012.
[45] K. Deb and H. Jain, "An evolutionary many-objective optimization algorithm using reference-point-based nondominated sorting approach, part i: solving problems with box constraints," *IEEE transactions on evolutionary computation*, vol. 18, no. 4, pp. 577–601, 2013.
[46] A. Panichella, "An improved pareto front modeling algorithm for large-scale many-objective optimization," in *Proceedings of the Genetic and Evolutionary Computation Conference*, 2022, pp. 565–573.
[47] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *ArXiv preprint*, vol. abs/1707.06347, 2017. [Online]. Available: https://arxiv.org/abs/1707.06347
[48] A. Stooke, J. Achiam, and P. Abbeel, "Responsive safety in reinforcement learning by PID lagrangian methods," in *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, ser. Proceedings of Machine Learning Research, vol. 119. PMLR, 2020, pp. 9133–9143. [Online]. Available: http://proceedings.mlr.press/v119/stooke20a.html

Qingchao Liu received the Ph.D. degree from Southeast University, Nanjing, China, in 2015. He joined the Automotive Engineering Research Institute, Jiangsu University, Zhenjiang, where he is currently working as an Associate Professor. His research interests include Driving behavior analysis, path planning, autonomous vehicle, cooperative adaptive cruise control, traffic flow theory, and intelligent transportation systems. He concurrently serves as a member of the American TRB Artificial Intelligence Subcommittee, a member of the China Society of Automotive Engineers, a member of the China Intelligent Transportation Association, and a senior visiting scholar at Nanyang Technological University in Singapore.

Chengzhi Gao received the bachelor degree in Information and Computing Science from the Anhui Agricultural University in 2022. He is currently pursuing a postgraduate degree in transportation engineering with Jiangsu University, China. His research interests include vehicular platoon control, traffic simulation and intelligent transportation systems. He is currently the deputy leader of the Waliang Lake Ecological Observation Team.

Xiangkun He (Member, IEEE) is currently a UESTC 100 Young Professor at the University of Electronic Science and Technology of China. Previously, he was a Research Fellow at Nanyang Technological University, Singapore, and served as a Senior Research Scientist at Huawei Noah's Ark Lab from 2019 to 2021. He earned his Ph.D. in 2019 from the School of Vehicle and Mobility at Tsinghua University. He has authored over 50 papers in top-tier journals and conferences, such as TPAMI, TNNLS, TITS, TRC, and Engineering, and holds 8 granted patents. His research interests include reinforcement learning, trustworthy AI, autonomous vehicles, and robotics. He has received many awards and honors, selectively including the Tsinghua University Outstanding Doctoral Dissertation Award in 2019, Best Paper Finalist at IEEE ICMA 2020, Huawei Major Technological Breakthrough Award in 2021, Best Paper Runner-Up Award at CVCI 2022, and Runner-Up in the Intelligent Algorithm Final of the 2022 Alibaba Global Future Vehicle Challenge. He serves as a reviewer for over 50 renowned journals and conferences and was also a member of the award committee for the Autonomous Driving Control Benchmark Challenge at IEEE CDC 2023.

Hai Wang (Senior Member, IEEE) received the B.S., M.S., and Ph.D. degrees from the School of Instrument Science and Engineering, Southeast University, Nanjing, China, respectively. In 2012he joined the School of Automotive and Traffic Engineering at Jiangsu University, where, he is currently working as a Professor. He has authored or coauthored more than 50 papers in the field of machine vision-based environment sensing for intelligent vehicles. His research interests include computer vision, intelligent transportation systems and intelligent vehicles.



Chen Lv (Senior Member. IEEE) received the Ph.D. degree from the Department of Automotive Engineering, Tsinghua University China in 2016From 2014 to 2015, he was a joint Ph.D. Researcher with the EECS Department, University of California at Berkeley. He is currently an Assistant Professor with Nanyang Technology University, Singapore. His research interests include cyber-physical systems, hybrid systems, advanced vehicle control, and intelligence, where he has contributed over 90 articles and holds 12 granted Chinese patents. He received the Highly Commended Paper Award of IMechE, U.K., in 2012, the National Fellowship for Doctoral Student in 2013, the NSK Outstanding Mechanical Engineering ward in 2014, the China SAE Outstanding Paper Award. in 2015, the 1st Class Award of China Automotive Industry Scientific and Technological Invention in 2015, the Tsinghua University Outstanding Doctoral Thesis Award in 2016, and the IV2018 Best Workshop/Special Issue Paper Award. He serves as a Guest Editor for IEEE Intelligent Transportation Systems Magazine, IEEE/ASME TRANSACTIONS ON MECHATRONICS, and Applied Energy, and an Associate Editor/Editorial.

Yingfeng Cai (Senior Member, IEEE) received the B.S., M.S., and Ph.D. degrees from the School of Instrument Science and Engineering, Southeast University, Nanjing. China. In 2013, she joined the Automotive Engineering Research Institute, Jiangsu University, as a Professor. Her research interests include computer vision, intelligent transportation systems. And intelligent automobiles.

Long Chen received the B.Sc. and Ph.D. degrees in mechanical engineering from Jiangsu University, Zhenjiang, China, in 1982 and 2006, respectively. He is currently a Professor with the Automotive Engineering Research Institute, Jiangsu University Zhenjiang. His research interests include electric vehicles, electric drives, simulation and control of vehicle dynamic performance, vehicle operation, and transport planning.