Constraint-Aware Diffusion Guidance for Robotics: Real-Time Obstacle Avoidance for Autonomous Racing

Hao Ma[1,2], Sabrina Bodmer[1], Andrea Carron[1], Melanie Zeilinger[1] [*], Michael Muehlebach[2*]
[1] Department of Mechanical and Process Engineering, ETH Zurich, Switzerland
[2] Max Planck Institute for Intelligent Systems, Tübingen, Germany
{haomah,sabodmer,carrona,mzeilinger}@ethz.ch, michaelm@tuebingen.mpg.de

Abstract: Diffusion models hold great potential in robotics due to their ability to capture complex, high-dimensional data distributions. However, their lack of constraint-awareness limits their deployment in safety-critical applications. We propose Constraint-Aware Diffusion Guidance (CoDiG), a data-efficient and general-purpose framework that integrates barrier functions into the denoising process, guiding diffusion sampling toward constraint-satisfying outputs. CoDiG enables constraint satisfaction even with limited training data and generalizes across tasks. We evaluate our framework in the challenging setting of miniature autonomous racing, where real-time obstacle avoidance is essential. Real-world experiments show that CoDiG generates safe outputs efficiently under dynamic conditions, highlighting its potential for broader robotic applications. Videos are available at: https://www.youtube.com/watch?v=KNYsTdtdxOU

Keywords: Diffusion Guidance, Constraint-Aware Sampling, Real-Time Obstacle Avoidance, Autonomous Racing, Safe Control

1 Introduction

Since their inception [1, 2], diffusion models have achieved groundbreaking success in image [3, 4], audio [5], and video generation [6]. Due to their exceptional capability in modeling multimodal data and capturing complex high-dimensional distributions, they have recently also garnered significant attention in robotics [7, 8, 9]. Collectively, these works highlight how diffusion models address limitations of traditional policy approaches, such as unimodal assumptions or training instability, thereby offering a more versatile framework for robotic behavior learning.

While diffusion models hold significant promise for robotics, standard formulations still face important challenges related to safety and physical feasibility. Many approaches are trained purely on data without explicitly enforcing constraints, which can lead to collisions or dynamic infeasibility, particularly when encountering out-of-distribution scenarios [10, 11]. Additionally, existing methods often rely heavily on large-scale offline datasets to promote generalization, which can limit their adaptation to unseen environments [12, 13]. Addressing these challenges is crucial for enabling safe and reliable deployment of diffusion models in safety-critical robotic applications.

To overcome these limitations, we propose Constraint-Aware Diffusion Guidance (CoDiG), a general-purpose, data-efficient diffusion-based framework for real-time, safe trajectory generation in robotic tasks such as autonomous racing. CoDiG integrates a barrier function directly into the reverse diffusion process, steering the sampling away from unsafe or dynamically infeasible regions without relying on external classifiers or auxiliary models. To further accelerate sampling and enhance its stability, CoDiG employs a warm-start strategy by initializing the diffusion process near

[*] Shared last author.

9th Conference on Robot Learning (CoRL 2025), Seoul, Korea.

feasible solutions. By augmenting the score updates with barrier gradients during inference, CoDiG enforces safety constraints, enabling efficient and reliable deployment in safety-critical environments. Our main contributions are summarized as follows:

- We introduce Constraint-Aware Diffusion Guidance (CoDiG), a general-purpose and data-efficient trajectory generation framework that enforces constraints during inference, allowing safe and physically feasible generalization from a small set of expert demonstrations to novel scenarios.
- We propose a warm-start strategy that significantly accelerates the inference process, achieving real-time performance suitable for high-frequency, safety-critical control, while ensuring smooth transitions between trajectories generated at successive iterations.
- We deploy CoDiG on a real-world autonomous racing car, tracking the trajectories generated by CoDiG and demonstrating safe obstacle avoidance in dynamic scenarios.

2 Related Work

Many recent works have explored incorporating constraints into diffusion models for robotic tasks. Some approaches enforce constraints during training: Bastek et al. [14] integrate physical laws into the training objective to ensure physically consistent outputs; Giannone et al. [15] align sampling trajectories with constrained optimization paths; and Power et al. [16] separately train on different constraints and combine them at inference. Others address constraints during inference: Carvalho et al. [17] condition the sampling process on goal-reaching and obstacle avoidance; Christopher et al. [18], Xiao et al. [19] enforce feasibility through projection steps, albeit with significant computational overhead; Römer et al. [20] incorporate model-based projections directly into the backward diffusion process to enforce constraint satisfaction during trajectory generation, avoiding the need for external post-sampling corrections; and Yu et al. [21] generate local collision-free motions through conditional sampling. Several methods handle constraints in both training and inference phases, such as Ajay et al. [22] for decision-making, Gong et al. [23] with trajectory-level diffusion, and Botteghi et al. [24], which train safe priors and apply runtime filtering. Among these, Yu et al. [21] primarily handle inference-time constraints, while Botteghi et al. [24] combine both stages. Overall, incorporation during training time promotes inherent feasibility, while inference-time methods offer flexibility at the cost of higher computational complexity during inference.

Compared to prior inference-time approaches, our CoDiG framework handles constraints by augmenting score updates with lightweight barrier gradients during sampling, without relying on projections, auxiliary models, or simulators, unlike classifier-guided [25] and energy-guided diffusion [26] that learn auxiliary networks at each step. This provides efficient, continuous guidance toward feasible trajectories while preserving the generative flexibility of diffusion models with a time-dependent weight γ_t that ramps up during denoising. Warm-start initialization further accelerates convergence and enhances sampling stability, enabling real-time deployment (2.5 Hz on hardware). Unlike previous works mainly evaluated in simulation or in quasi-static environments, we demonstrate CoDiG on a real-world autonomous racing platform, where strict dynamic feasibility and rapid obstacle avoidance are critical. While Sheebaelhamd et al. [27] have also suggested autoregressive architectures as an alternative to diffusion-based generation, it is unclear whether constrained-aware generation via barrier functions is also effective with these architectures. These aspects highlight the unique contributions of CoDiG in enabling efficient, reliable, and real-time constraint handling within generative robotic planning. See Appendix A for a compact side-by-side summary of training- vs. inference-time strategies (Table 1).

3 Methodology

Recent advances in score-based generative modeling introduced diffusion processes and stochastic differential equations (SDEs), offering a continuous-time view of forward noise injection and reverse denoising [28]. Since our work builds on this foundation, we briefly review score-based generative modeling and introduce the notation used throughout the paper.

3.1 Preliminaries

Let $x_0 \in \mathbb{R}^d$ denote a noise-free data sample drawn from the data distribution $p_0(x)$. A score-based generative model defines a continuous-time diffusion process $\{x_t\}_{t\in[0,T]}$, where t denotes the diffusion time, such that x_T becomes approximately Gaussian. It is important to note that throughout this paper, we encounter two notions of "time": here, t refers to the artificial diffusion time governing the processes, while later, τ will denote the physical time in real-world dynamical systems.

Diffusion Process. The forward diffusion process gradually perturbs the data by solving the following SDE:

$$\mathrm{d}x_t = f(x_t, t)\,\mathrm{d}t + g(t)\,\mathrm{d}w_t, \quad t \in [0, T], \quad x_0 \sim p_0,$$

where $x_t \in \mathbb{R}^d$ is the perturbed data at time t, $f : \mathbb{R}^d \times [0, T] \to \mathbb{R}^d$ is the drift term, $g : [0, T] \to \mathbb{R}$ is the scalar-valued diffusion term, and $w_t \in \mathbb{R}^d$ denotes a standard Wiener process.

A common instantiation of the diffusion process is the *Ornstein–Uhlenbeck (OU) process* [29], in which the drift pulls x_t toward a mean $\mu \in \mathbb{R}^d$:

$$\mathrm{d}x_t = \beta(t)(\mu - x_t)\,\mathrm{d}t + g(t)\,\mathrm{d}w_t, \quad t \in [0, T], \tag{1}$$

where $\beta(t)$ is a positive scalar-valued function controlling the drift strength. In this case, the OU process admits a closed-form solution for the mean and variance of the marginal distribution of x_t. Specifically, by introducing

$$\bar{\beta}_t := \exp\left(-\int_0^t \beta(\nu)\ \mathrm{d}\nu\right),$$

then the marginal distribution of x_t is Gaussian and expressed as

$$p_t(x_t \mid x_0) = \mathcal{N}\left(x_t;\ \mu - (\mu - x_0)\bar{\beta}_t,\ \frac{g(t)^2}{2\beta(t)}\left(\mathbb{I} - \bar{\beta}_t^2\mathbb{I}\right)\right), \tag{2}$$

where $\mathbb{I} \in \mathbb{R}^{d\times d}$ is the identity matrix.

Sampling Process. To generate new data, one samples from the reverse-time SDE corresponding to the forward process. Under suitable regularity conditions, this reverse SDE takes the form [30]:

$$\mathrm{d}x_t = \left[f(x_t, t) - g(t)^2\,\nabla_x \log p_t(x_t)\right]\ \mathrm{d}t + g(t)\ \mathrm{d}\widetilde{w}_t, \quad t \in [0, T], \quad x_T \sim p_{x_T}, \tag{3}$$

where $\widetilde{w}_t$ is a standard Wiener process running backward in time, and $\nabla_x \log p_t(x_t)$ is the score function of the marginal distribution.

In practice, the score function is unknown and approximated by a neural network $s_\theta(x_t, t)$ trained using denoising score matching. The training objective minimizes the expected squared error between the predicted score and the true score:

$$\mathbb{E}_{t\sim \mathrm{U}[0,T]}\mathbb{E}_{x_0\sim p_0(x)}\mathbb{E}_{x_t\sim p_t(x_t|x_0)}\left[\left|s_\theta(x_t, t) - \nabla_x \log p_t(x_t \mid x_0)\right|^2\right],$$

where $|\cdot|$ denotes the ℓ_2-norm, and $\mathrm{U}[0, T]$ the uniform distribution with support $[0, T]$.

3.2 Constraint-Aware Diffusion Guidance

Before introducing the proposed CoDiG framework, we specify the functional forms of the drift and diffusion terms in (1) for concreteness and clarity. It is important to emphasize that the proposed framework does not rely on these specific choices - the following definitions are adopted purely for illustrative purposes and to remain consistent with the experimental setup described later.

We let $\mu = 0$, and define the drift term and the diffusion term as

$$f(x_t, t) = -\beta(t)\,x_t, \quad g(t) = \sqrt{2\beta(t)}, \quad t \in [0, T],$$

which yields the so-called variance preserving SDE [28], where $g(t)^2 = 2\beta(t)$ holds for all $t \in [0, T]$ such that the marginal variance of x_t is preserved over time. This specific choice ensures that the forward process remains stable and tractable for training and sampling, while still allowing for an expressive and well-defined reverse-time generative process. Under this formulation, the forward diffusion process described by (1) converges to a standard Gaussian distribution for large T. As analyzed in Song et al. [28], the term $\sqrt{2\beta(t)}$ should grow with time, requiring $\beta(t)$ to be strictly increasing.

For simplicity and numerical stability, we normalize the diffusion process to $t \in [0, 1]$. To ensure convergence to a standard Gaussian, the diffusion term $\sqrt{2\beta(t)}$ must grow rapidly within this interval. In our implementation, we model $\beta(t)$ as a quadratic function, $\beta(t) = r_1 t^2 + r_0$, with parameters detailed in Appendix D.3. In this case, (3) can be reformulated as:

$$\mathrm{d}x_t = \left[-\beta(t)\, x_t - 2\beta(t)\, \nabla_x \log p_t(x_t)\right] \mathrm{d}t + \sqrt{2\beta(t)}\, \mathrm{d}\widetilde{w}_t, \quad t \in [0, 1], \quad x_1 \sim p_{x_1}. \tag{4}$$

Next, we consider the marginal distribution $p_t(x_t)$, which represents the probability distribution of a sample at an intermediate time step, in the absence of constraints. Before incorporating constraints into this distribution, we first introduce the following definitions. Let $c : \mathbb{R}^d \times [0, 1] \to \mathbb{R}^k$ denote a time-dependent constraint function, encoding the safety or feasibility requirements of the system. We define the feasible region at time $\tau \geq 0$ as

$$\mathcal{C}_\tau := \left\{x \in \mathbb{R}^d \mid c(x, \tau) \leq 0\right\},$$

where the inequality is interpreted element-wise. Naturally, when constraints are introduced, the distribution of interest becomes the conditional distribution:

$$p_t(x_t \mid \mathcal{C}_\tau), \quad t \in [0, 1], \quad \tau \geq 0.$$

These constraints may encode different forms of feasibility or safety requirements, depending on the task setting. For example, in autonomous racing, $\mathcal{C}_\tau$ refers to the obstacle-free, drivable region of a racing track. While in diffusion-based control policies, $\mathcal{C}_\tau$ must account for system dynamics, for respecting the underlying physical constraints of the system. Here, we use the time subscript τ to emphasize that such constraints can be time-varying, which is often the case in dynamic or interactive environments. For simplicity, and without loss of clarity, we will omit this subscript in the following when no confusion arises.

Several existing methods attempt to directly model the marginal distribution $p_t(x_t \mid \mathcal{C})$ by injecting the constraint representation into the diffusion model architecture [31]. While effective in big-data domains such as image synthesis, these approaches face significant limitations in the context of robotics: (i) Learning $p_t(x_t \mid \mathcal{C})$ from scratch requires many expert demonstrations that satisfy $\mathcal{C}$, which are often expensive or impractical to collect in robotics. (ii) Since $\mathcal{C}$ is often time-varying and task-specific, models trained on a fixed distribution may fail to generalize to unseen or dynamic constraints at test time.

To overcome these limitations, we leverage the known structure of the constraint $\mathcal{C}$ during sampling to dynamically guide the generation process. We propose an alternative formulation of the constrained distribution $p_t(x_t \mid \mathcal{C})$, which does not require learning the conditional model directly from data:

$$p_t(x_t \mid \mathcal{C}) = p_t(x_t)\, \frac{e^{-\gamma_t V(x_t;\, \mathcal{C})}}{Z_t},$$

where $Z_t := \int_{\mathbb{R}^d} p_t(x)\, e^{-\gamma_t V(x,\, \mathcal{C})}\, \mathrm{d}x$ is a normalization constant. The barrier function $V : \mathbb{R}^d \to \mathbb{R}_+$ assigns large values to infeasible data points, while remaining close to zero within the feasible region. Intuitively, applying the barrier function pushes the probability of infeasible data points toward zero. Importantly, the barrier function is *derived from explicit task constraints* (e.g., obstacle clearance and near time-optimality) rather than a heuristic penalty, and we use a *time-dependent weight* γ_t that ramps up across denoising steps to enforce constraints more strongly as samples approach the data manifold. As a result, the constrained distribution focuses its support almost entirely on the feasible region. We substitute the above formula into the score function and get

$$\nabla_x \log p_t(x_t \mid \mathcal{C}) = \nabla_x \log p_t(x_t) - \gamma_t \nabla_x V(x_t;\, \mathcal{C}), \tag{5}$$

where the normalization constant Z_t vanishes when taking the gradient of the log-probability, and hence does not affect the reverse-time dynamics. By substituting the right-hand side of (5) into (4), we obtain the modified reverse SDE that incorporates constraint information:

$$\mathrm{d}x_t = \beta(t)\left[-x_t - (1+\eta)\left(\nabla_x \log p_t(x_t) - \gamma_t \nabla_x V(x_t;\, \mathcal{C})\right)\right]\,\mathrm{d}t + \eta\sqrt{2\beta(t)}\,\mathrm{d}\widetilde{w}_t, \tag{6}$$

where a constant $\eta \in [0,1]$ is introduced to accelerate convergence and enhance the stability of the sampling process [32]. We observe that the first term on the right-hand side of (5) corresponds exactly to the unconstrained score function defined in (4). This term can still be approximated by the neural network $s_\theta(x_t, t)$ trained without considering any constraints. Crucially, the effect of the constraint appears only during the denoising process, in an explicit gradient-based form - as an additive term derived from the constraint potential (e.g., a barrier function). This formulation significantly reduces the need for large amounts of constraint-compliant training data, as the constraint is not encoded in the network itself but *instead injected at inference time*. Moreover, because the constraint enters the reverse SDE as a differentiable time-varying potential, the framework can naturally accommodate dynamic, time-varying constraints.

It is important to note, as pointed out by Bastek et al. [14], that applying constraints to data that is close to pure noise in diffusion models is not meaningful. Therefore, during the denoising process - i.e., as t decreases from one to zero - we gradually increase the value of γ_t starting from zero at $t = 1$. This progressive scheduling is crucial for ensuring the stability of the denoising process. The specific design of γ_t is detailed in Appendix B.

4 Case Study of Autonomous Racing

In this section, we illustrate how to design a constraint-aware barrier function and analyze its impact on inference through a concrete application - obstacle avoidance in autonomous racing. While this example serves to ground our discussion, the barrier function design and constraint-handling mechanisms are task-agnostic. Thus, our framework is not limited to autonomous racing but serves as a general-purpose solution for safety-critical robotics applications. For details on dataset construction, diffusion model architecture, and training procedures, please refer to Appendix D.

4.1 Constraint-Aware Barrier Function

For the considered application, the barrier function, which is instantiated from task constraints (safety and near time-optimality) rather than tuned heuristics, is designed as follows:

$$V\left(\hat{y}, \widehat{\phi};\, \mathcal{C}\right) = \sum_{k=0}^{N-1} \underbrace{\alpha \mathbb{1}\left\{\hat{y}_k \notin \mathcal{C}_k\right\}}_{\text{first part}} + \underbrace{\frac{\epsilon}{2}\left|\hat{y}_k - \hat{y}_{\text{nominal},k}\right|^2 + \frac{\epsilon}{2}\left|\widehat{\phi}_k - \widehat{\phi}_{\text{nominal},k}\right|^2}_{\text{second part}}, \tag{7}$$

where the symbol $\mathbb{1}\{\cdot\}$ denotes the indicator function[2], and the subscript $(\cdot)_{\text{nominal}}$ refers to the time-optimal solution computed offline in the absence of obstacles, which serves as a reliable reference. Here, N represents the number of discrete points obtained by uniformly sampling along the track center line. In our setting, $\hat{y}$ denotes the lateral displacement and $\widehat{\phi}$ represents the yaw angle in the Frenet coordinate system (see Appendix D.1). The feasible region $\mathcal{C}_k$ is also defined in the Frenet frame, capturing the obstacle-free area at each sampled position along the track.

In (7), the first term handles time-varying obstacles, while the second biases toward nominal time-optimal motion when curvature is absent in the Frenet view, aiding dynamic feasibility. Beyond promoting near time-optimality without requiring global geometric knowledge of the track, this part also facilitates the generation of dynamically feasible trajectories. The local representation ensures that the resulting motions adhere more closely to the physical and kinematic constraints of the system. The positive constants α and ϵ are tunable hyperparameters that balance the importance

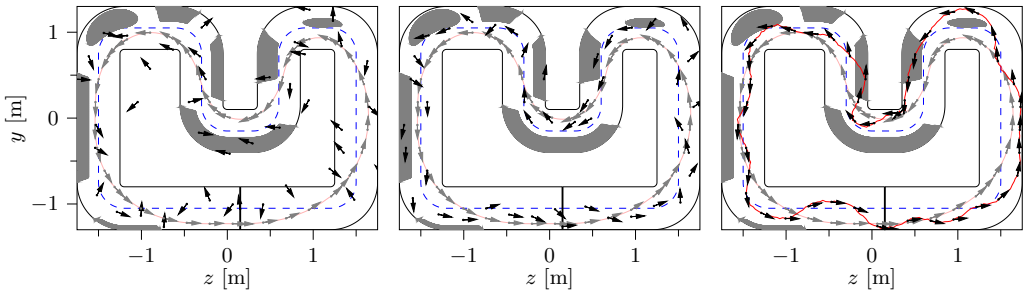
[2]One possible differentiable approximation is provided in Appendix E.

of the two components. Specifically, α modulates the influence of physical safety constraints, while ϵ regulates the adherence to nominal time-optimality.

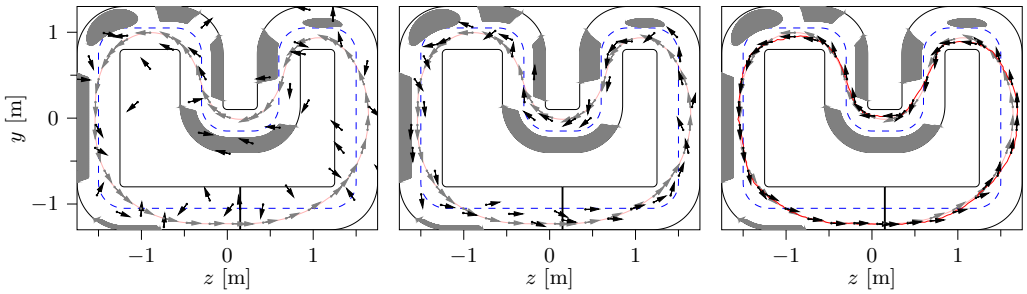
It is worth noting that the design of the barrier function is not unique and can be tailored to the specific task. While such customization may require a small amount of tuning effort, it is negligible compared to the cost of collecting expert demonstrations, especially in robotics domains where data is expensive. This makes our framework both flexible and data-efficient.

4.2 Constraint-Aware Inference

We train the diffusion model as described in Appendix D.2 for 500 epochs. During inference, we applied the Euler-Maruyama discretization (8), which corresponds to the discretized version of (6). The denoising process proceeds from $t = 1.0$ to $t = 0.0$ in 1000 steps, gradually transforming samples from noise to data. The results are shown in Fig. 1, where the gray regions indicate obstacles, the (light) red curve shows the trajectory in the z-y plane. Fig. 1a illustrates the denoising process



(a) Sampling without barrier function.



(b) Sampling with barrier function.

Figure 1: Intermediate denoising results during sampling at three representative time steps $t = 1\,\mathrm{s},\ 0.591\,\mathrm{s},\ 0.002\,\mathrm{s}$, from left to right. The black arrows denote the generated position and heading of the vehicle, and the gray arrows denote the offline-computed optimal solution.

without using the barrier function, while Fig. 1b shows the effect of the proposed constraint-aware guided generation. Each row depicts intermediate generation results at $t = 1\,\mathrm{s},\ 0.591\,\mathrm{s},\ 0.002\,\mathrm{s}$, from left to right. See Appendix B for the concrete values of the hyperparameters used during inference.

In Fig. 1a, we observe that the diffusion model learns important features of the data distribution. The model respects the fundamental constraints of the racetrack, such as staying within bounds and satisfying the loop closure. However, despite conditioning on constraints during training, the generated samples still fail to consistently avoid obstacles - primarily due to limited training data. Additionally, the lack of curvature information in the local Frenet frame leads to unrealistic and physically implausible results.

In contrast, Fig. 1b incorporates the barrier function as described in Sec. 4.1. The guidance significantly improves the sampling process. The model denoises faster (i.e., the samples become structured earlier), the trajectories successfully avoid all obstacles, and the resulting motion aligns well with a physically consistent motion. Moreover, due to the second part in (7) (albeit with a small weight ϵ), the final trajectories closely follow the nominal time-optimal ones, exhibiting near time-optimal properties. For more details on the near time-optimality of the generated trajectories, please refer to Appendix G.

5 Real-World Experiments

We evaluate CoDiG in experiments on a real-world miniature autonomous racing platform [33, 34]. For more details on the experimental platform, the obstacle setup, and a flowchart illustrating how the CoDiG framework is deployed to achieve real-time obstacle avoidance, please refer to Appendix H.

5.1 Warm-Starting

Real-time obstacle avoidance requires not only safe trajectories but also fast replanning. As shown in Sec. 4.2, our diffusion model with a barrier function produces high-quality trajectories after 1000 denoising steps, but this results in a low sampling frequency of $0.25\,\mathrm{Hz}$, which is insufficient for real-time racing.

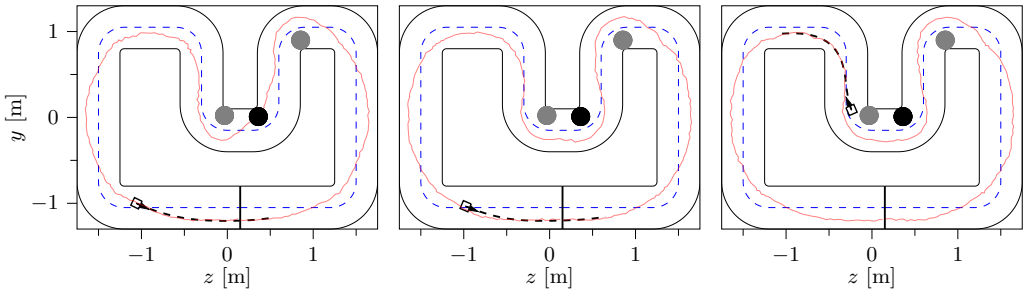
While various acceleration techniques exist [35, 11, 36], we propose a warm-start strategy tailored to robotic control. Unlike standard diffusion generation, which samples each trajectory from pure noise, our proposed warm-start technology perturbs the previous output with small noise and reuses it as the next input. This maintains temporal consistency, reduces trajectory variance, and improves control stability [37]. By promoting smooth transitions between consecutive trajectories, warm-starting significantly lowers the number of denoising steps required and enhances real-time feasibility. A detailed analysis and comparison of results with and without warm-starting are provided in Appendix F.

5.2 Experimental Results

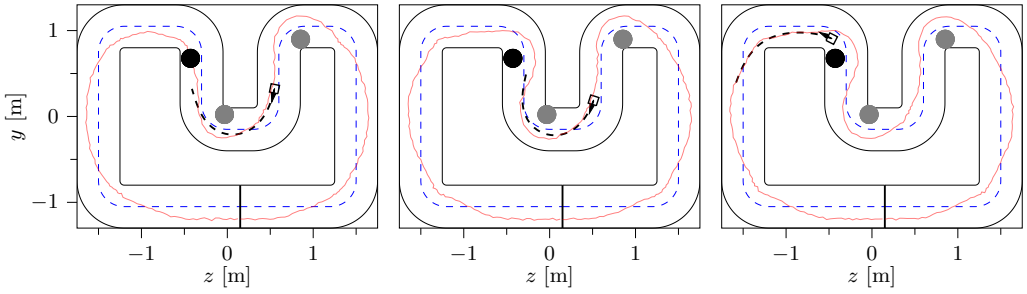
Through the integration of our warm-start technique, we achieve a sampling frequency of $2.5\,\mathrm{Hz}$ on a computer equipped with an `NVIDIA RTX 4090` GPU. While there is still potential for further acceleration, this performance already satisfies the real-time requirements of obstacle avoidance in racing scenarios.

We successfully deployed CoDiG on our real-world autonomous racing platform for real-time trajectory planning. A tracking model predictive control (TMPC) [38, 39] is employed to follow the planned trajectories. Notably, the TMPC operates without any knowledge of obstacles, relying solely on the reference trajectories for control. Fig. 2 illustrates two representative obstacle avoidance maneuvers during autonomous driving. In both Fig. 2a and Fig. 2b, the red lines represent the trajectories planned by CoDiG, the gray circles denote static obstacles, while the black circles indicate dynamic obstacles. The black dashed lines show the predicted trajectories generated by the TMPC as it attempts to follow the red reference trajectory. Each sequence from left to right captures a complete avoidance cycle: (i) Obstacle Encroachment: An obstacle intrudes into a previously feasible trajectory, making it infeasible. (ii) Replanning: The planner detects the encroachment and generates a new collision-free trajectory. (iii) Successful Avoidance: The vehicle safely bypasses the obstacle.

As shown in the figures, the predicted trajectories from the TMPC closely align with the reference trajectories generated by the diffusion model. This highlights that the planned trajectories are closely aligned with physical feasibility, enabled by the barrier function, which is crucial for effective tracking performance. Additionally, even in the presence of obstacles, the generated trajectories maintain near time-optimality, indicating that the planner does not overly sacrifice efficiency for safety.



(a) Obstacle avoidance episode 1.



(b) Obstacle avoidance episode 2.

Figure 2: Real-world demonstration of real-time obstacle avoidance using CoDiG. Red lines represent the planned trajectories generated by the CoDiG diffusion planner. Gray circles indicate static obstacles, and black circles represent dynamic obstacles. Black dashed lines show the predicted trajectory from the TMPC while following the reference plan. Each episode illustrates a complete avoidance cycle: obstacle encroachment, real-time replanning, and successful passage.

Finally, thanks to the warm-start strategy, significant replanning is only triggered when the obstacle actually interferes with the current path. In static conditions, consecutive trajectories remain almost unchanged, ensuring system stability. We conducted five experimental trials, each consisting of 15 racing laps, across ten different obstacle configurations. The framework achieved a $100\,\%$ success rate in obstacle avoidance, demonstrating its robustness and reliability in diverse scenarios.

6 Conclusion

In this work, we propose CoDiG (Constraint-Aware Diffusion Guidance), a general, data-efficient framework that leverages diffusion models for real-time, safety-critical motion planning in robotics. While diffusion models have shown strong capabilities in learning complex distributions, their direct application in robotics is hindered by the lack of constraint-awareness and physical feasibility. We address this challenge by introducing a barrier function into the denoising process, guiding the generated trajectories toward safe and dynamically consistent regions without requiring extensive expert data. A warm-start inference strategy further improves inference speed and temporal consistency for real-time deployment.

We demonstrate CoDiG on autonomous racing with dynamic obstacles, achieving robust real-world performance with reliable obstacle avoidance, precise tracking, and near time-optimal planning at $2.5\,\mathrm{Hz}$. These results highlight the potential of diffusion-based methods for constraint-aware planning and control, offering a promising direction for safe, efficient, and generalizable robotic decision-making in varying environments.

7 Limitations

While our proposed CoDiG framework has demonstrated promising results in generating safe, dynamically consistent trajectories with high data efficiency, we acknowledge several limitations and potential areas for improvement:

Sampling Frequency Constraints. Although we have introduced a warm-start strategy to significantly accelerate the sampling process, the current planning frequency of $2.5\,\mathrm{Hz}$ may still fall short in scenarios involving fast-moving or rapidly appearing obstacles. In such cases, the planner may fail to respond quickly enough to ensure safe maneuvering.

Inability to Detect Infeasible Scenarios. Our current framework does not explicitly identify situations where no feasible trajectory exists - such as when the track is completely blocked. Due to the nature of diffusion models, the network continues to generate trajectories regardless of feasibility, which can result in solutions that appear smooth but are physically invalid. This limitation arises from the data distribution encountered during training, where such infeasible configurations are typically not represented.

Heuristic Design of Barrier Function. The construction of the barrier function, which plays a central role in enforcing safety and physical consistency, currently relies on heuristic parameter tuning. While this provides sufficient flexibility to adapt to various environments, it lacks theoretical grounding or systematic design principles. Developing more principled methods for parameter selection remains an open challenge.

Acknowledgments

We thank Matteo Facchino for providing code related to time-optimal control solvers. We also gratefully acknowledge Jan-Hendrik Bastek for the insightful discussions on constraint handling in diffusion models. We thank the German Research Foundation and the Max-Planck ETH Center for Learning Systems for the support.

References

[1] J. Sohl-Dickstein, E. Weiss, N. Maheswaranathan, and S. Ganguli. Deep Unsupervised Learning using Nonequilibrium Thermodynamics. In *International Conference on Machine Learning*, pages 2256–2265, 2015.

[2] J. Ho, A. Jain, and P. Abbeel. Denoising Diffusion Probabilistic Models. In *Advances in Neural Information Processing Systems*, pages 6840–6851, 2020.

[3] J. Ho, C. Saharia, W. Chan, D. J. Fleet, M. Norouzi, and T. Salimans. Cascaded Diffusion Models for High Fidelity Image Generation. *Journal of Machine Learning Research*, 23(47): 1–33, 2022.

[4] G. Batzolis, J. Stanczuk, C.-B. Schönlieb, and C. Etmann. Conditional Image Generation with Score-Based Diffusion Models. *arXiv:2111.13606*, pages 1–19, 2021.

[5] M. Jeong, H. Kim, S. J. Cheon, B. J. Choi, and N. S. Kim. Diff-TTS: A Denoising Diffusion Model for Text-to-Speech. *arXiv:2104.01409*, pages 1–5, 2021.

[6] J. Ho, T. Salimans, A. Gritsenko, W. Chan, M. Norouzi, and D. J. Fleet. Video Diffusion Models. In *Advances in Neural Information Processing Systems*, pages 8633–8646, 2022.

[7] R. Wolf, Y. Shi, S. Liu, and R. Rayyes. Diffusion Models for Robotic Manipulation: A Survey. *arXiv:2504.08438*, pages 1–28, 2025.

[8] C. Chi, Z. Xu, S. Feng, E. Cousineau, Y. Du, B. Burchfiel, R. Tedrake, and S. Song. Diffusion policy: Visuomotor policy learning via action diffusion. *The International Journal of Robotics Research*, 0(0):1–21, 2024.

[9] J. Urain, N. Funk, J. Peters, and G. Chalvatzaki. SE(3)-DiffusionFields: Learning smooth cost functions for joint grasp and motion optimization through diffusion. In *International Conference on Robotics and Automation*, pages 5923–5930, 2023.

[10] K. Kondo, A. Tagliabue, X. Cai, C. Tewari, O. Garcia, M. Espitia-Alvarez, and J. P. How. CGD: Constraint-Guided Diffusion Policies for UAV Trajectory Planning. *arXiv:2405.01758*, pages 1–8, 2024.

[11] N. D. Palo, L. Hasenclever, J. Humplik, and A. Byravan. Diffusion Augmented Agents: A Framework for Efficient Exploration and Transfer Learning. In *Conference on Lifelong Learning Agents*, pages 268–284, 2025.

[12] K. M. Lee, S. Ye, Q. Xiao, Z. Wu, Z. Zaidi, D. B. D'Ambrosio, P. R. Sanketi, and M. Gombolay. Learning Diverse Robot Striking Motions with Diffusion Models and Kinematically Constrained Gradient Guidance. *arXiv:2409.15528*, pages 1–8, 2025.

[13] Z. Liang, Y. Mu, M. Ding, F. Ni, M. Tomizuka, and P. Luo. AdaptDiffuser: Diffusion Models as Adaptive Self-evolving Planners. In *International Conference on Machine Learning*, pages 20725–20745, 2023.

[14] J.-H. Bastek, W. Sun, and D. M. Kochmann. Physics-Informed Diffusion Models. *arXiv:2403.14404*, pages 1–26, 2025.

[15] G. Giannone, A. Srivastava, O. Winther, and F. Ahmed. Aligning Optimization Trajectories with Diffusion Models for Constrained Design Generation. In *Advances in Neural Information Processing Systems*, pages 51830–51861, 2023.

[16] T. Power, R. Soltani-Zarrin, S. Iba, and D. Berenson. Sampling Constrained Trajectories Using Composable Diffusion Models. In *International Conference on Intelligent Robots and Systems*, pages 1–5, 2023.

[17] J. Carvalho, A. T. Le, M. Baierl, D. Koert, and J. Peters. Motion Planning Diffusion: Learning and Planning of Robot Motions with Diffusion Models. In *International Conference on Intelligent Robots and Systems*, pages 1916–1923, 2023.

[18] J. K. Christopher, S. Baek, and F. Fioretto. Constrained Synthesis with Projected Diffusion Models. *arXiv:2402.03559*, pages 1–20, 2024.

[19] W. Xiao, T.-H. Wang, C. Gan, and D. Rus. SafeDiffuser: Safe Planning with Diffusion Probabilistic Models. *arXiv:2306.00148*, pages 1–19, 2023.

[20] R. Römer, A. von Rohr, and A. P. Schoellig. Diffusion Predictive Control with Constraints. *arXiv:2412.09342*, pages 1–14, 2024.

[21] W. Yu, J. Peng, H. Yang, J. Zhang, Y. Duan, J. Ji, and Y. Zhang. LDP: A Local Diffusion Planner for Efficient Robot Navigation and Collision Avoidance. In *International Conference on Intelligent Robots and Systems*, pages 5466–5472, 2024.

[22] A. Ajay, Y. Du, A. Gupta, J. Tenenbaum, T. Jaakkola, and P. Agrawal. Is Conditional Generative Modeling All You Need for Decision-Making? *arXiv:2211.15657*, pages 1–24, 2023.

[23] Z. Gong, A. Kumar, and P. Varakantham. Offline Safe Reinforcement Learning Using Trajectory Classification. In *Conference on Artificial Intelligence*, pages 16880–16887, 2025.

[24] N. Botteghi, F. Califano, M. Poel, and C. Brune. Trajectory Generation, Control, and Safety with Denoising Diffusion Probabilistic Models. *arXiv:2306.15512*, pages 1–18, 2023.

[25] P. Dhariwal and A. Nichol. Diffusion Models Beat GANs on Image Synthesis. In *Advances in Neural Information Processing Systems*, pages 8780–8794, 2021.

[26] C. Lu, H. Chen, J. Chen, H. Su, C. Li, and J. Zhu. Contrastive Energy Prediction for Exact Energy-Guided Diffusion Sampling in Offline RL. In *International Conference on Machine Learning*, pages 22825–22855, 2023.

[27] Z. Sheebaelhamd, M. Tschannen, M. Muehlebach, and C. Vernade. Quantization-Free Autoregressive Action Transformer. *arXiv:2503.14259*, pages 1–15, 2025.

[28] Y. Song, J. Sohl-Dickstein, D. P. Kingma, A. Kumar, S. Ermon, and B. Poole. Score-Based Generative Modeling through Stochastic Differential Equations. In *International Conference on Machine Learning*, pages 10362–10383, 2021.

[29] B. Øksendal. *Stochastic Differential Equations*. Springer Berlin Heidelberg, 1995.

[30] B. D. Anderson. Reverse-Time Diffusion Equation Models. *Stochastic Processes and their Applications*, 12(3):313–326, 1982.

[31] J. Ho and T. Salimans. Classifier-Free Diffusion Guidance. *arXiv:2207.12598*, pages 1–14, 2022.

[32] Y. Song and S. Ermon. Improved Techniques for Training Score-Based Generative Models. In *Advances in Neural Information Processing Systems*, pages 12438–12448, 2020.

[33] S. Bodmer, L. Vogel, S. Muntwiler, A. Hansson, T. Bodewig, J. Wahlen, M. N. Zeilinger, and A. Carron. Optimization-Based System Identification and Moving Horizon Estimation Using Low-Cost Sensors for a Miniature Car-Like Robot. *arXiv:2404.08362*, pages 1–11, 2024.

[34] A. Carron, S. Bodmer, L. Vogel, R. Zurbrügg, D. Helm, R. Rickenbach, S. Muntwiler, J. Sieber, and M. N. Zeilinger. Chronos and CRS: Design of a miniature car-like robot and a software framework for single and multi-agent robotics and control. In *International Conference on Robotics and Automation*, pages 1371–1378, 2023.

[35] J. Song, C. Meng, and S. Ermon. Denoising Diffusion Implicit Models. *arXiv:2010.02502*, pages 1–22, 2022.

[36] Q. Zhang and Y. Chen. Fast Sampling of Diffusion Models with Exponential Integrator. *arXiv:2204.13902*, pages 1–33, 2023.

[37] M. Morari and J. H. Lee. Model predictive control: past, present and future. *Computers & Chemical Engineering*, 23(4-5):667–682, 1999.

[38] D. Limon, I. Alvarado Aldea, T. Alamo, and E. F. Camacho. MPC for tracking piecewise constant references for constrained linear systems. *Automatica*, 44(9):2382–2387, 2008.

[39] R. Soloperto, J. Köhler, and F. Allgöwer. A Nonlinear MPC Scheme for Output Tracking Without Terminal Ingredients. *Transactions on Automatic Control*, 68(4):2368–2375, 2023.

[40] T. Sauer. Numerical Solution of Stochastic Differential Equations in Finance. In *Handbook of Computational Finance*, pages 529–550. Springer, 2012.

[41] R. Verschueren, M. Zanon, R. Quirynen, and M. Diehl. Time-optimal Race Car Driving using an Online Exact Hessian based Nonlinear MPC Algorithm. In *European Control Conference*, pages 141–147, 2016.

[42] H. C. Crenshaw and L. Edelstein-Keshet. Orientation by Helical Motion—II. Changing the Direction of the Axis of Motion. *Bulletin of Mathematical Biology*, 55(1):213–230, 1993.

[43] R. Rombach, A. Blattmann, D. Lorenz, P. Esser, and B. Ommer. High-Resolution Image Synthesis with Latent Diffusion Models. *arXiv:2112.10752*, pages 1–45, 2022.

[44] J. Hensman, N. Durrande, and A. Solin. Variational Fourier Features for Gaussian Processes. *Journal of Machine Learning Research*, 18(151):1–52, 2018.

A Comparison of Constraint-Enforcement Strategies

This section provides a concise, side-by-side comparison of how constraints are handled across methods in Table 1: whether constraints are enforced during training, inference, or both; the guidance source (e.g., classifier, energy, projection, or physics-based barrier); whether learned auxiliary models or projection steps are required; and whether any real-time evidence is reported.

B Hyperparameters

During inference, the hyperparameters are set as follows:

$$\eta = 0.1, \quad \alpha = 15.0, \quad \epsilon = 0.4.$$

In practice, we set α in the range 10–15, whereas ϵ is chosen at a smaller magnitude, typically 0.1–0.5. We also observe that our method is not particularly sensitive to these hyperparameters across environments: the same settings generalize well on both training and test scenarios.

In particular, the time-varying weight $\gamma(t)$ is assigned non-uniform values according to the following scheme:

$$\gamma(t) = \frac{\hbar_1}{1 + \exp\left(-\hbar_2\left(\hbar_3 - t\right)\right)}, \quad t \in [0, 1],$$

where $\hbar_1 = 1.0$, $\hbar_2 = 50.0$, and $\hbar_3 = 0.7$. The choice of the function γ is not unique. The guiding principle is to introduce the constraint progressively during denoising so that its gradient increasingly shapes the samples as they approach the data manifold. Any monotone schedule that follows this principle (e.g., linear or logistic ramp-up) works in practice.

C Discrete-Time Integration

Assuming a denoising process over $M \in \mathbb{N}_{++}$ steps, we partition the interval $[0, 1]$ non-uniformly as follows:

$$t_k = \left(1 - \frac{k}{M}\right)^p, \quad k = 0, \ldots, M,$$

where $p = 2.2$ in our case. Starting from an initial sample x_0 drawn from a standard Gaussian distribution, we perform denoising according to the following discrete Euler-Maruyama [40] update scheme:

$$\begin{aligned} \bar{x}_{k+1} &= x_k + \beta(t_k)\left[-x_k - (1+\eta)\left(s_\theta(x_k, t_k) - \gamma_{t_k} \nabla_x V(x_k;\, \mathcal{C})\right)\right] \Delta t_k, \\ x_{k+1} &= \bar{x}_{k+1} + \eta \sqrt{2\beta(t_k)}\, \sqrt{|\Delta t_k|}\, \sigma_k, \quad k = 0, \ldots, M-1, \end{aligned} \tag{8}$$

where $\Delta t_k = t_{k+1} - t_k$ denoting the step size between successive time points. The noise term $\sigma_k \in \mathbb{R}^d$ is sampled from a standard Gaussian distribution. Here, $\bar{x}$ denotes the mean estimate at each step, while x denotes the noisy sample.

D Data and Model Pipeline

In this section, we describe the pipeline used to train our diffusion model for obstacle avoidance in racing scenarios. We begin by presenting our data collection process, where expert demonstrations are gathered to reflect optimal driving behaviors in the presence of obstacles. Then, we introduce a data augmentation strategy that diversifies the training distribution while preserving expert intent. Next, we detail the architecture of our proposed diffusion model, which is adapted to handle time-conditioned inputs and spatial constraints relevant to the racing task. Finally, we present the training results of the diffusion model under various input configurations, demonstrating how different modalities affect the training performance.

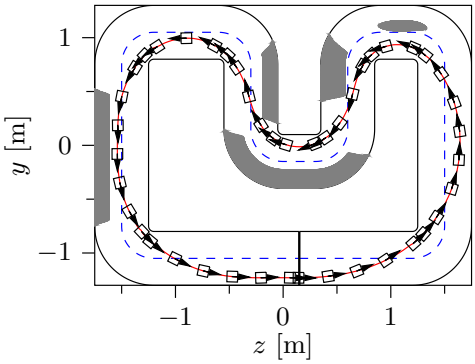
Method	Stage	Guidance source	Aux. model	Projection	RT evidence
CoDiG (ours)	Inference	Barrier gradient (physics & safety)	✗	✗	✓ (2.5 Hz)
Dhariwal and Nichol [25]	Inference	Classifier $\nabla \log p\,(y\|x)$	✓	✗	–
Lu et al. [26]	Inference	Energy gradient	✓	✗	–
Christopher et al. [18] Xiao et al. [19]	Inference	Projection operator	◯	✓	–
Römer et al. [20]	Inference	Model-based constraints	◯	◯	–
Yu et al. [21]	Inference	Local collision-free conditioning	◯	✗	–
Bastek et al. [14]	Train	Physics-informed training loss	✗	✗	–
Giannone et al. [15]	Train	Align to constrained opt. paths	✗	✗	–
Power et al. [16]	Train	Multi-constraint training	◯	✗	–
Ajay et al. [22]	Both	Constrained decision diffusion	◯	◯	–
Gong et al. [23]	Both	Trajectory-level diffusion w/ constraints	◯	◯	–
Botteghi et al. [24]	Both	Safe priors + runtime filtering	◯	◯	–

Table 1: Comparison across enforcement phases. "Aux." indicates learned auxiliary networks; "Projection" indicates projection steps; RT = real-time. "✓" means the item is required, "✗" means it is not needed, "◯" denotes that it is optional, and "–" means that no result is reported.

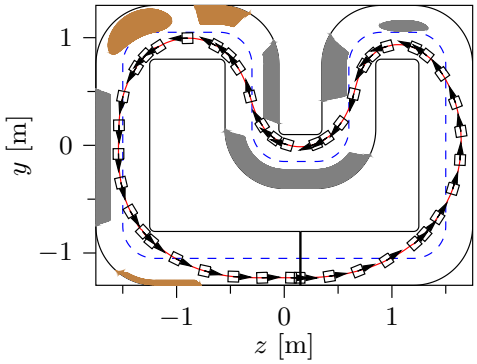
D.1 Learning-Efficient Dataset Construction

Even on a miniature autonomous racing platform, collecting expert demonstrations via manual teleoperation is highly challenging and time-consuming. Therefore, we generate expert data by solving a time-optimal control problem [41], including car states and control inputs.

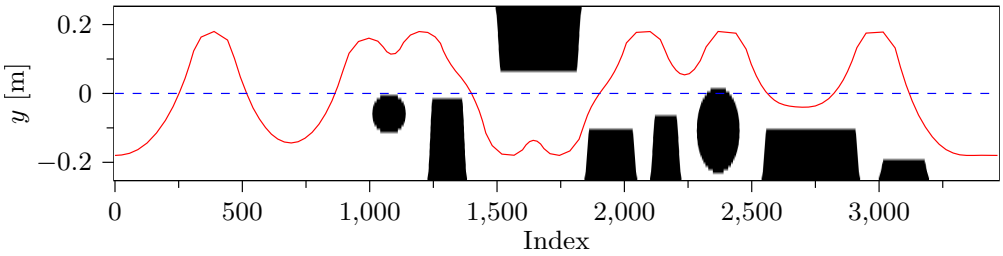
To collect expert data, we randomly place obstacles on the track and solve the aforementioned time-optimal control problem to obtain optimal driving trajectories with continuous looping and corresponding control inputs. An example is shown in Fig. 3a, where the gray regions indicate obstacles. The red curve shows the trajectory in the z-y plane, and the black rectangles and arrows illustrate the approximate shape and orientation of the vehicle, respectively, reflecting the fact that the vehicle is not treated as a point mass to account for the system dynamics.



(a) Time-optimal expert trajectory.



(b) Redundant obstacle augmentation.



(c) Flattened Frenet representation.

Figure 3: (a) A time-optimal trajectory (red line) computed for a given obstacle configuration (gray regions) on the racing track. Black rectangles and arrows indicate the approximate vehicle shape and heading. (b) Redundant obstacles (brown regions) added in areas that do not affect the trajectory, providing data augmentation without solving additional optimal control problems. (c) A flattened track representation in the local Frenet coordinate system, visualizing both the trajectory (red line) and obstacles (black regions).

As previously mentioned, collecting expert data is expensive. Solving a single time-optimal control problem takes around 10 minutes on average. To address this limitation, we propose a method for dataset augmentation. We observe that once the time-optimal solution is obtained for a given map (with a specific obstacle configuration), adding extra obstacles within the safe region that do not interfere with the trajectory will not alter the time-optimal solution. These redundant obstacles - illustrated as brown regions in Fig. 3b - can be arbitrarily placed without affecting the outcome. Based on this observation, we first collect 100 trajectories by solving time-optimal problems with randomly placed obstacles, which takes approximately 16 hours in total. We then expand this dataset to 10 000 trajectories by adding random redundant obstacles in safe regions, using 80 % of them when training the diffusion model.

During training, we only use the pose information - namely y and yaw angle ϕ - which are transformed into a local Frenet coordinate system [42]. This yields the local variables $\hat{y}$ and $\hat{\phi}$, representing the lateral displacements and heading relative to the reference path. Together with the obstacle representation, this results in a flattened map as shown in Fig. 3c. In this map, the presence of obstacles naturally induces an obstacle-free region, denoted by $\mathcal{C}$, which is already defined in the local Frenet frame. For notational simplicity, we omit the explicit time index τ, but we emphasize that $\mathcal{C}$ is inherently time-varying, reflecting the dynamic nature of the environment. The set $\mathcal{C}$ provides a time-varying constraint in the planning process and is considered in the definition of our constraint-aware barrier function.

By performing this transformation, we deliberately discard information about the global curvature of the track. This enhances the generalization capability of the trained diffusion model, enabling the model to handle arbitrary (even moving) obstacles. However, this also means that the generated trajectories may not inherently account for curvature constraints, an issue we address using a barrier function in the denoising process, which is detailed in Sec. 4.2.

D.2 Diffusion Model Architecture

As illustrated in Fig. 4, we adopt a time-conditioned U-Net architecture as the backbone of our diffusion model [43]. The network follows a classic encoder-decoder structure, augmented with time and conditional information to support trajectory generation in dynamic environments.

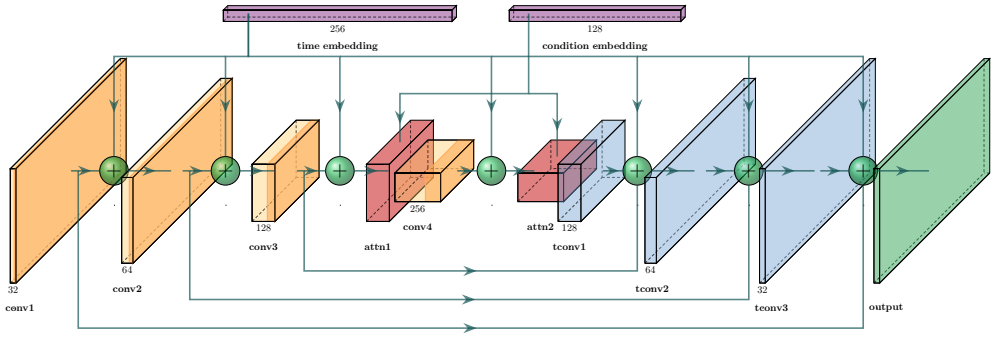


Figure 4: Architecture of the proposed time-conditioned score-based generative model. The U-Net backbone extracts multi-scale features through a sequence of convolutional and deconvolutional layers, with temporal embeddings injected via dense layers. Spatial transformer modules enable conditional attention guided by task-specific context. Skip connections ensure spatial consistency across scales.

The input is a single-channel spatial-temporal representation of the trajectory, and the output preserves the same spatial resolution. Temporal conditioning is achieved via Gaussian Fourier features [44], which embed the diffusion time step into a high-dimensional representation. This embedding is injected at every resolution level to inform the network of the denoising progress.

The encoder consists of a sequence of down-sampling convolutional blocks, each followed by time embedding fusion and group normalization. To enhance spatial reasoning and enable conditional generation, spatial transformer modules are inserted at deeper layers, where they incorporate context information - such as a reference track - encoded via a lightweight convolutional neural network.

The decoder mirrors the encoder with up-sampling blocks and skip connections, allowing the network to reconstruct high-resolution outputs by fusing low-level and high-level features. Each decoding layer is also conditioned on time to ensure consistency with the diffusion process.

This architecture is designed to be data-efficient, modular, and generalizable. It supports plug-and-play conditional guidance and is easily extendable to other tasks in robotics beyond the case study of autonomous racing.

D.3 Training of the Network

We experiment with different input modalities for the diffusion model. Specifically, we considered: (i) the lateral displacement $\hat{y}$ after transforming into the Frenet coordinate system; (ii) both the lateral displacement $\hat{y}$ and the yaw angle $\widehat{\phi}$ in the Frenet frame; (iii) the states including $\hat{x},\ \hat{y},\ \widehat{\phi}$ along with their corresponding velocities $\hat{v}_x,\ \hat{v}_y,\ \widehat{\omega}$ in the Frenet frame. For each input configuration, we train the model for 500 epochs and explore different values of r_1 and r_0 in constructing the noise schedule $\beta(t) = r_1 t^2 + r_0$ for $t \in [0, 1]$. The training results are shown in Fig. 5, where the three plots from left to right correspond to the aforementioned three input configurations, respectively.

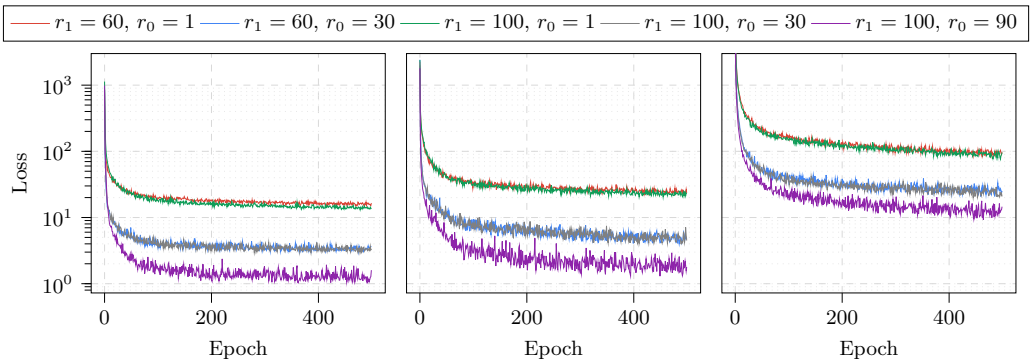


Figure 5: Training performance of the diffusion model under different input configurations and noise schedules. From left to right, the three plots correspond to using (i) lateral displacement $\hat{y}$ in the Frenet frame only, (ii) lateral displacement $\hat{y}$ and yaw angle $\widehat{\phi}$ in the Frenet frame, and (iii) the states $\hat{x},\ \hat{y},\ \widehat{\phi}$ along with velocities $\hat{v}_x,\ \hat{v}_y,\ \widehat{\omega}$ as model inputs. Each setting was trained for 500 epochs while varying the parameters r_1 and r_0 in the noise schedule.

Our experiments show that varying r_1 has negligible impact on the final training performance. In contrast, increasing r_0 generally improves training outcomes, suggesting that larger initial noise levels may facilitate better learning. However, due to the limited size of our training dataset, excessively large values of r_0 can lead to overfitting risks. Additionally, we observe that as the input dimensionality increases, the training performance degrades, likely due to the increased complexity of the data distribution and the limited model capacity under fixed training resources. Based on these observations, we choose to use only the lateral displacement $\hat{y}$ and the yaw angle $\widehat{\phi}$ in the Frenet frame as inputs in our final framework, setting $r_1 = 100.0$ and $r_0 = 30.0$.

It is important to emphasize that although we adopt a simplified input representation in this work, our approach remains general and can naturally extend to handle higher-dimensional or multimodal inputs. This flexibility paves the way toward directly modeling control inputs using diffusion models in future work.

E Differentiable Approximation of the Indicator

To facilitate reproducibility, we detail a specific approximation of the indicator in (7), while noting that alternative formulations are possible. We approximate the indicator $\mathbb{1}\{\hat{y}_k \notin \mathcal{C}_k\}$ with a *piecewise-linear function* defined on the signed distance to the nearest obstacle center. Let $\mathrm{dist}(\hat{y}_k)$ be the signed distance that is *positive inside obstacles*, zero on the boundary, and negative outside. We use the normalized linear map

$$\widehat{\mathbb{1}}\{\hat{y}_k \notin \mathcal{C}_k\} = \min\left(1, \max\left(0, \frac{\mathrm{dist}(\hat{y}_k)}{\rho(\mathcal{C}_k)}\right)\right)$$

so that points on the obstacle edge map to zero and (approximately) the obstacle center maps to one. Here ρ is a normalization scale corresponding to the obstacle half-width (for disks, the radius;

for general shapes, the inradius or a fixed calibration constant). In practice, we compute $\mathrm{dist}(\cdot)$ from the obstacle *binary mask* using a (Euclidean) distance transform; outside obstacles $\mathrm{dist}(\cdot) \leq 0$ hence $\widehat{\mathbb{1}} = 0$, while inside obstacles it increases linearly with the interior distance and saturates at one.

F Warm-Start Evaluation

In this work, we incorporate a warm-starting strategy to accelerate the sampling process, thereby enabling real-time obstacle avoidance. This section presents a quantitative analysis of the effects introduced by this partial diffusion strategy on trajectory generation performance.

Fig. 6 illustrates the reference trajectories generated with and without the application of the warm start technique under an identical obstacle configuration, sampled at consistent time instances. In the figures, gray circles denote static obstacles, while black circles denote dynamic obstacles. The nine subfigures are arranged sequentially from left to right and top to bottom. In each subfigure, the black solid line represents the trajectory obtained using the standard diffusion model, which initiates from standard Gaussian noise and progresses through 500 denoising steps. In contrast, the red solid line corresponds to the trajectory generated with the warm start method, which undergoes 50 denoising steps of partial noised initial trajectory.

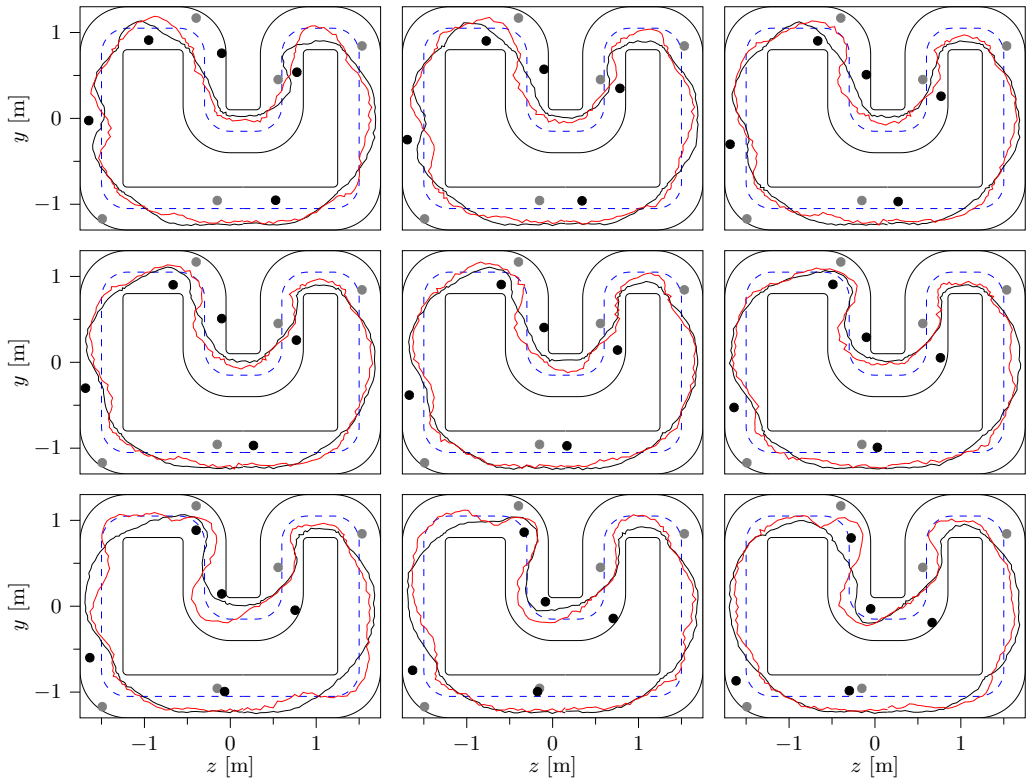


Figure 6: Comparison of reference trajectories generated with and without the warm start technique under an identical obstacle configuration. Gray circles denote static obstacles, and black circles denote dynamic obstacles. The black solid lines represent trajectories produced by the standard diffusion model after 500 denoising steps starting from standard Gaussian noise. The red solid lines represent trajectories generated using the warm start approach, where 50 denoising steps are performed. The warm start method accelerates the sampling process while maintaining successful obstacle avoidance, albeit with slightly coarser trajectory profiles and more conservative motion planning behavior.

As evidenced by the results, both approaches successfully achieve obstacle avoidance at all time steps, demonstrating their respective effectiveness. Nevertheless, the trajectories generated via the warm start technique exhibit a coarser structure, primarily due to the incomplete denoising process inherent to partial diffusion. Furthermore, from the perspective of physical feasibility, the trajectories derived from the standard diffusion model better adhere to realistic vehicle dynamics. Specifically, the final subfigure demonstrates that the warm start method tends to converge to a local solution and favors a more conservative path - remaining closer to the previous time point - to avoid obstacles. Despite this conservatism, the warm start approach proves crucial, as it reduces the sampling time by approximately a factor of three, thereby making real-time obstacle avoidance feasible. Moreover, the conservative behavior introduced by warm start contributes positively to the overall system stability.

G Near Time-Optimality

In this section, we demonstrate the near time-optimality of the trajectories generated by CoDiG by comparing them with trajectories obtained by solving an offline time-optimal control problem [41]. As illustrated in Fig. 7, we present several representative obstacle configurations extracted from a real-world experiment. In each scenario, the red trajectory denotes the real-time obstacle-avoidance path generated by the CoDiG framework, while the black trajectory represents the time-optimal path computed offline under the same obstacle layout.

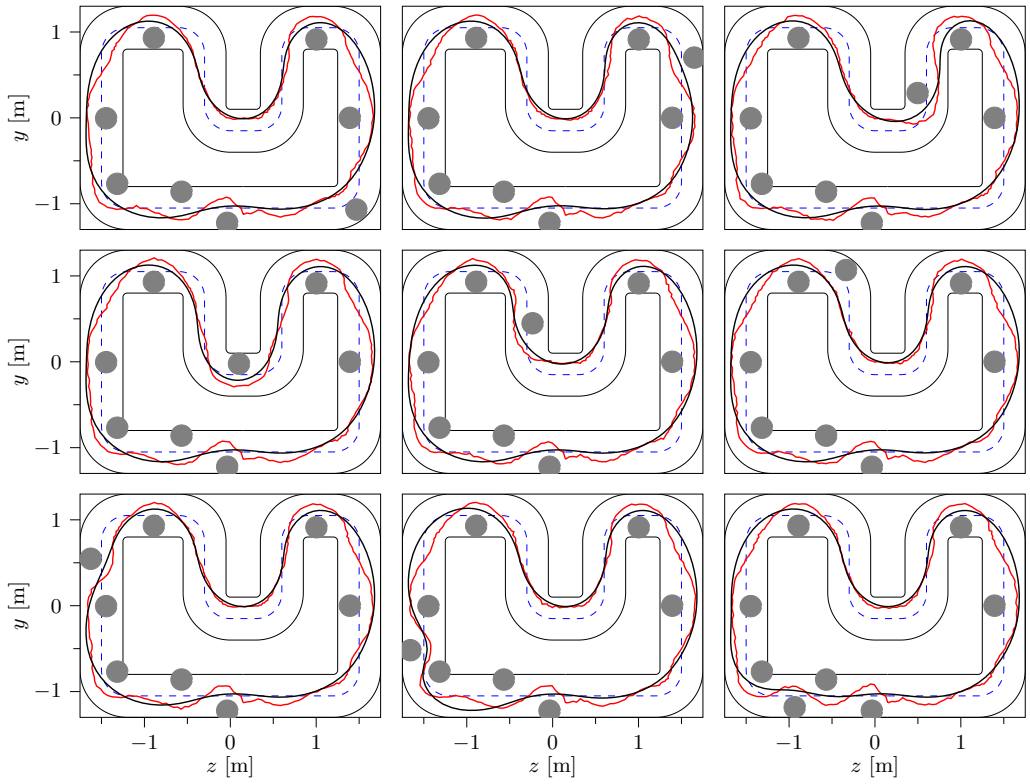


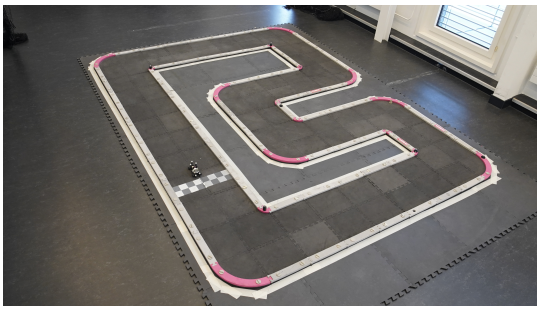
Figure 7: Comparison between trajectories generated in real time by CoDiG (red) and offline-computed time-optimal trajectories (black) under various obstacle configurations.

Overall, we observe a high degree of similarity between the real-time and offline trajectories, which highlights the near time-optimal generation of CoDiG in practice. The main discrepancies are observed in two typical situations. First, to achieve faster cornering, the offline time-optimal solution tends to favor a larger turning radius in curved sections. Second, when navigating near obstacles,

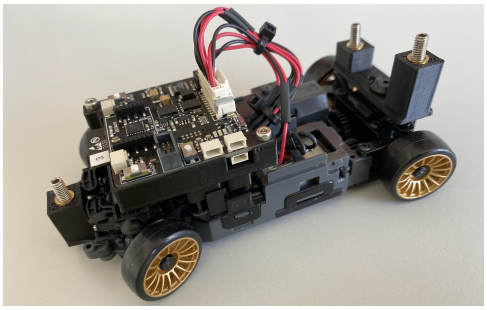
the CoDiG-generated trajectory increases its clearance for safety, resulting in a slight deviation from the time-optimal path. This trade-off ensures safety while maintaining strong time-efficiency.

H Experimental Platform and the CoDiG Framework

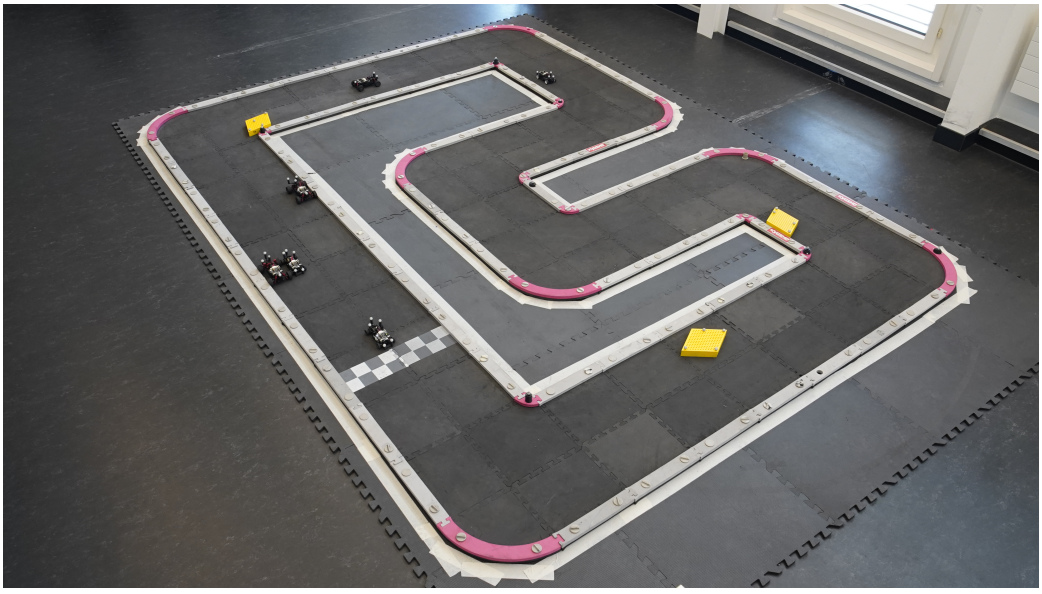
Fig. 8 illustrates the experimental platform used to evaluate the performance of the CoDiG framework for real-time obstacle avoidance in autonomous racing. The platform consists of a down-scaled race track (Fig. 8a), a custom-built autonomous car (Fig.8b), and a motion capture system (not shown in the figure). This setup enables agile maneuvering and real-time control in dynamic, safety-critical scenarios such as obstacle avoidance. It provides a reproducible environment to evaluate our approach under realistic conditions.



(a) Down-scaled race track.



(b) Custom-built autonomous car.



(c) Obstacle configuration.

Figure 8: Experimental platform used to evaluate the performance of the CoDiG framework for real-time obstacle avoidance in autonomous racing. The setup includes (a) a down-scaled race track, (b) a custom-built autonomous vehicle, and (c) an obstacle configuration that simulates a challenging and realistic racing scenario.

Additionally, Fig. 8c depicts the obstacle configuration used during the experiments. The vehicle positioned at the starting line is the one under our control, responsible for executing the obstacle avoidance task. Yellow boxes represent static obstacles, while the remaining vehicles serve as either dynamic or static obstacles. This setup faithfully simulates a complex and challenging racing envi-

ronment, emphasizing the effectiveness and robustness of our framework under realistic and difficult conditions.

The flowchart illustrating how the CoDiG framework enables real-time obstacle avoidance for autonomous racing on the experimental platform is shown in Fig. 9. The core component of the CoDiG framework is a trained diffusion planner module, which generates a safe reference trajectory y_{ref} capable of avoiding all obstacles. This is achieved by incorporating map and obstacle information, and guiding the sampling process via gradients provided by a constraint-aware guidance mechanism.

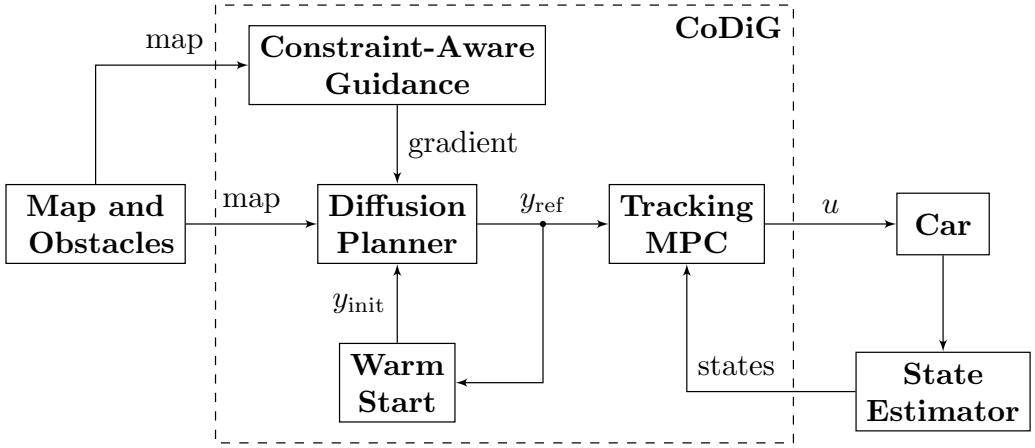


Figure 9: Flowchart of the proposed CoDiG framework for real-time obstacle avoidance in autonomous racing. The framework integrates a diffusion-based trajectory planner, a constraint-aware guidance module that guides the denoising process, a warm start strategy to accelerate sampling, and a tracking MPC controller. All modules operate within the experimental platform described in Fig. 8.

To improve sampling efficiency, the reference trajectory generated at the current time point is further used to construct the initial input y_{init} for the diffusion process at the next time step, via a warm start strategy. This replaces the conventional use of standard Gaussian noise as the initial condition, thereby accelerating the trajectory generation process.

Subsequently, a tracking MPC module computes the control input u required to follow the reference trajectory y_{ref}, based on the current vehicle state estimated by a state estimator module. Finally, the control input u is applied to the vehicle to execute real-time obstacle avoidance.