2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
October 14-18, 2024. Abu Dhabi, UAE

# CoBL-Diffusion: Diffusion-Based Conditional Robot Planning in Dynamic Environments Using Control Barrier and Lyapunov Functions

Kazuki Mizuta[1] Karen Leung[1,2]

***Abstract*— Equipping autonomous robots with the ability to navigate safely and efficiently around humans is a crucial step toward achieving trusted robot autonomy. However, generating robot plans while ensuring safety in dynamic multi-agent environments remains a key challenge. Building upon recent work on leveraging deep generative models for robot planning in static environments, this paper proposes CoBL-Diffusion, a novel diffusion-based safe robot planner for dynamic environments. CoBL-Diffusion uses Control Barrier and Lyapunov functions to guide the denoising process of a diffusion model, iteratively refining the robot control sequence to satisfy the safety and stability constraints. We demonstrate the effectiveness of CoBL-Diffusion using two settings: a synthetic single-agent environment and a real-world pedestrian dataset. Our results show that CoBL-Diffusion generates smooth trajectories that enable the robot to reach goal locations while maintaining a low collision rate with dynamic obstacles.**

## I. Introduction

Achieving safe and efficient navigation for autonomous robots around humans is essential for building trust in autonomous systems and enabling their widespread adoption. As ensuring safety is paramount, especially when interacting with humans, it is often desirable to leverage control-theoretic tools such as reachability analysis [1], [2] and control invariant set theory [3], [4] to define and enforce safety constraints within a model-based trajectory optimizer. While such approaches produce well-understood and well-behaved robot motions [5]–[7], they quickly become overly conservative and/or intractable for uncertain and complicated settings.

In contrast, emerging data-driven, i.e., deep learning, robot planners provide a scalable and tractable solution as they can leverage data from simulation or real-world interactions [8], [9] and use high-dimensional observations as inputs. However, learning-based approaches inherently lack safety guarantees and can exhibit unpredictable failures, leading to unanticipated behaviors that pose safety risks for robots and their surroundings. Recent work has investigated applying control-theoretic safety filters on learning-based planners to combine the safety and generalizability benefits from each method [10]–[13]. In this work, we study further the advantages of integrating control-theoretic techniques for safety assurance into diffusion-based motion planning frameworks.

Diffusion models [14], [15] have emerged as a powerful class of deep generative models, achieving remarkable success in generating high-quality images and synthesizing speech [16], [17], and more recently, in robot trajectory generation [18]–[21]. However, current Diffusion-based robot trajectory generation methods still lack explicit safety assurances in dynamic environments, prohibiting their application in safety-critical scenarios involving human-robot interaction. In this work, we investigate the use of diffusion models as the foundation for robot navigation planning in dynamic multi-agent environments and present a method that integrates control-theoretic safety and stability constraints into the diffusion model framework.

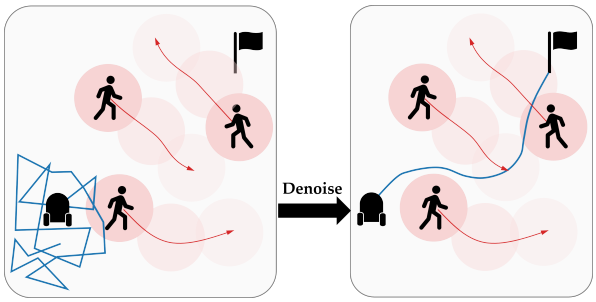



Fig. 1: CoBL-Diffusion uses control barrier and Lyapunov functions to guide a diffusion process to generate a robot controller for goal-reaching while avoiding dynamic obstacles.

Inspired by the Diffuser introduced in [18] which incorporates user-defined reward functions into the diffusion process for flexible conditioning on the output behavior, we propose **CoBL-Diffusion**, a novel diffusion model for safe robot planning leveraging **Co**ntrol **B**arrier Functions (CBFs) and Control **L**yapunov Functions (CLFs) for the enforcement of desired safety and stability (i.e., goal-reaching) properties. CBFs and CLFs are popular and well-understood mathematical frameworks based on invariant set theory for proving and ensuring the safety and stability of dynamical systems [3]. They have been successfully applied to mobile robots for safe collision-free navigation [22]–[24]. In this paper, we present the architecture behind CoBL-Diffusion that generates dynamically feasible trajectories that satisfy the CBF and CLF constraints. We demonstrate and evaluate the performance of CoBL-Diffusion in navigation tasks using synthetically generated and real-world scenarios, showing that it produces robot motions that can safely and smoothly navigate through crowded human environments.

**Statement of Contributions.** The contributions of the paper are summarized as follows:

1) We propose **CoBL-Diffusion**, a diffusion-based robot planner for dynamic environments that incorporates control-theoretic safety and stability constraints. The diffusion process is guided by the gradient of reward functions derived from the CBF and CLF.
2) Our method ensures consistency between the control sequence and the resulting states by explicitly integrating generated controls through system dynamics to obtain the states. Maintaining dynamic consistency during the diffusion steps is crucial, as the CBF and

Kazuki Mizuta is partially supported by the Nakajima Foundation. This work was supported by the UW + Amazon Science Hub Research Award.
[1]University of Washington, Department of Aeronautics and Astronautics, [2] NVIDIA, Contact: {mizuta, kymleung}@uw.edu

CLF reward computations in CoBL-Diffusion require the state trajectory to be consistent with the control inputs.

3) We demonstrate the effectiveness of our approach in generating safe robot motion plans for navigating through crowded dynamic environments. We provide qualitative and quantitative evaluations using synthetic and real-world pedestrian data.

**Organization.** We first contextualize our work in Section II, and then introduce key concepts on diffusion models, CBFs, and CLFs in Section III. In Section IV, we describe CoBL-Diffusion, including the architecture and choice of reward functions. We discuss experimental results of CoBL-Diffusion evaluated on a synthetic and real-world dataset in Section V, and conclude with ideas for future work in Section VI.

## II. Related Work

In this section, we provide a brief overview of robot planning algorithms using CBFs and diffusion models.

**Control Barrier Functions (CBFs).** CBFs provide an elegant mathematical framework based on invariant set theory to construct safe control constraints that prevent a dynamical system from entering an unsafe region in state space [3]. A brief introduction on CBF is given in Section III-C. CBFs have been successfully applied to various robotic applications, ranging from navigation [22], [25], robot manipulation [26], and legged locomotion [27]. For high-dimensional settings (e.g., multi-agent, history-dependence) and when operating with uncertain sensor observations, there has been a recent surge in combining CBFs with learning-based techniques, including learning CBFs from data [28], [29], and integrating CBFs as a safety filter for learning-based controllers [10], [30], [31]. In general, there is a growing interest in using CBFs as a safety filter for learning-enabled systems that otherwise would struggle with confidently exhibiting safe behaviors. In this work, we investigate the integration of CBFs into diffusion models, a recent generative modeling technique that is showing promise for robot planning applications [18], [19], [32]. *Note*: There are strong parallels between CBFs and CLFs; see Section III-D. Many CBF-based approaches discussed also apply to CLFs, which are used for stability [33], [34].

**Diffusion Models for Planning.** Diffusion models, introduced by [14] and further developed by [15], are a class of deep generative models that represent data generation as an iterative denoising process. We provide a brief overview of diffusion models in Section III-A. The Diffuser [18] represents an innovative application of diffusion models to robot trajectory generation. The Diffuser combines the planning and sampling processes into a unified framework via iterative denoising of trajectories, thus enabling the generation of plans that are adaptable to various reward structures and constraints. There are follow-up works that aim to influence the diffusion process to produce trajectories that satisfy some desired properties. Work similar in spirit to our proposed method, SafeDiffuser [32] extends the application of diffusion models to safety-critical tasks by incorporating CBFs into the denoising diffusion procedure to ensure the generated trajectories satisfy safety constraints. This CBF guarantees forward invariance of the diffusion procedure rather than the conventional forward invariance in a time domain addressed in this work. Another recent work [19] proposes guided conditional diffusion in the context of simulating human driving behaviors in driving simulators. They use signal temporal logic specifications to encode desired behaviors such as goal-reaching and staying within speed limits.

## III. Preliminaries

Our approach consists of a combination of control-theoretic techniques and deep generative models to generate robot behaviors with desired safety and stability properties. In this section, we provide background on (i) planning with diffusion models, (ii) control barrier functions (CBFs), and (iii) control Lyapunov functions (CLFs).

### A. Diffusion Models

Deep generative modeling is the process of using deep neural networks to learn an underlying distribution explaining a dataset and sampling from the learned distribution to produce new examples similar to the dataset. A denoising diffusion probabilistic model (DDPM) [15] is a type of deep generative model consisting of two main processes: *forward diffusion* which turns clean data into white noise, and *reverse diffusion* which turns white noise into clean data (i.e., denoising process). By learning neural network parameters describing the reverse diffusion process, we can generate clean data samples by sampling from, typically, a Gaussian distribution. The forward process is defined by a sequence of conditional Gaussian distributions:

$$q(\boldsymbol{\tau}^i|\boldsymbol{\tau}^{i-1}) = \mathcal{N}(\boldsymbol{\tau}^i; \sqrt{1-\beta_i}\boldsymbol{\tau}^{i-1}, \beta_i \mathbf{I}), \tag{1}$$

where $\boldsymbol{\tau}^0$ is the original data, $\boldsymbol{\tau}^i$ is the data at diffusion step $i$, $\beta_i$ is the noise schedule, and $\mathbf{I}$ is the identity matrix. Intuitively, the forward diffusion process incrementally adds noise to the data, transforming the original distribution into a simple Gaussian distribution after $N$ steps. The reverse process is modeled by a series of conditional distributions:

$$p_\theta(\boldsymbol{\tau}^{i-1}|\boldsymbol{\tau}^i) = \mathcal{N}(\boldsymbol{\tau}^{i-1}; \boldsymbol{\mu}_\theta(\boldsymbol{\tau}^i, i), \boldsymbol{\Sigma}_\theta(\boldsymbol{\tau}^i, i)), \tag{2}$$

where $\boldsymbol{\mu}_\theta$ and $\boldsymbol{\Sigma}_\theta$ are parameterized by neural networks with parameters $\theta$. In this paper, we set $\boldsymbol{\Sigma}_\theta(\boldsymbol{\tau}^i, i) = \boldsymbol{\Sigma}^i$ for the reverse diffusion process to follow the cosine schedule [35]. The goal is to learn neural network parameters $\theta$ such that the reverse process reproduces the original data. The typical training loss uses a variant of the variational lower bound on the log-likelihood,

$$L_{\text{simple}} = \mathbb{E}_{\boldsymbol{\tau}^0, \boldsymbol{\epsilon}, i}[\|\boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta(\sqrt{\bar{\alpha}_i}\boldsymbol{\tau}^0 + \sqrt{1-\bar{\alpha}_i}\boldsymbol{\epsilon}, i)\|^2], \tag{3}$$

where $\boldsymbol{\epsilon}$ is Gaussian noise, $\boldsymbol{\epsilon}_\theta$ is the neural network predicting the noise, and $\bar{\alpha}_i = \prod_{j=1}^{i}(1-\beta_j)$ is the cumulative product of the noise schedule. To generate new samples, one starts from noise and iteratively applies (2) for a predetermined number of iterations.

### B. Trajectory Generation Using Diffusion Models

Diffusion models have demonstrated their potential in various planning and trajectory generation problems. Specifically, Diffuser [18] reformulates the trajectory optimization by integrating the planning process into the modeling framework based on a diffusion model, thus providing the underlying foundation that this (and many other) work builds upon. Unlike traditional algorithms that generate state sequences

(i.e., trajectories) autoregressively, Diffuser generates all timesteps of a plan simultaneously, allowing for anti-causal decision-making and temporal coherence, thereby promoting the global consistency of the sampled plan. In Diffuser, the trajectory consists of both states $\mathbf{s}$ and actions $\mathbf{a}$[1],

$$\boldsymbol{\tau} = \begin{bmatrix} s_0 & s_1 & \dots & s_T \\ a_0 & a_1 & \dots & a_T \end{bmatrix}. \tag{4}$$

The optimality of a state-action pair is denoted by $O_t$, a binary random variable where $p(O_t = 1) = \exp(r(s_t, a_t))$ and $r(s_t, a_t)$ is a user-specified reward function. To influence the denoising process to generate more optimal outputs with respect to $r(s_t, a_t)$, the reward function serves as a classifier guidance [36] to modulate the mean in the denoising steps,

$$p_\theta(\boldsymbol{\tau}^{i-1}|\boldsymbol{\tau}^i, O_{1:T}) \approx \mathcal{N}(\boldsymbol{\tau}^{i-1}; \boldsymbol{\mu} + \boldsymbol{\Sigma}\mathbf{g}, \boldsymbol{\Sigma}), \tag{5}$$

$$\mathbf{g} = \nabla_{\boldsymbol{\tau}} \log\, p(O_{1:T}|\boldsymbol{\tau}) = \sum_{t=0}^{T} \nabla_{\mathbf{s}_t, \mathbf{a}_t} r(\mathbf{s}_t, \mathbf{a}_t), \tag{6}$$

where $\mathbf{g}$ represents the gradient of the log probability of optimality with respect to the trajectory. Furthermore, the trajectory is conditioned on a given goal by fixing the terminal state, analogous to the inpainting problem from image generation [14].

### C. Control Barrier Function (CBF)

Given an unsafe set to avoid, CBFs impose a bound on the rate at which a dynamical system is allowed to approach the unsafe set, enforcing a rate of zero at the boundary of the unsafe set. Consider a control affine robotic system,

$$\dot{\mathbf{x}} = f(\mathbf{x}) + g(\mathbf{x})\mathbf{u}, \tag{7}$$

where $\mathbf{x} \in \mathcal{D} \subseteq \mathbb{R}^n$ is the state, $\mathbf{u} \in \mathcal{U} \subseteq \mathbb{R}^m$ is the control input. Both $f : \mathcal{D} \to \mathbb{R}^n$ and $g : \mathcal{D} \to \mathbb{R}^{n\times m}$ are local Lipschitz continuous functions. Suppose that the safe set $\mathcal{C} \subseteq \mathcal{D}$ is defined as the superlevel set of the continuously differentiable function $h : \mathcal{D} \to \mathbb{R}$ as follows,

$$\mathcal{C} = \{\mathbf{x} \in \mathcal{D} \mid h(\mathbf{x}) \geq 0\}. \tag{8}$$

*Definition 1 (Control Barrier Function [3]):* Suppose the dynamics is (7) and the safe set $\mathcal{C}$ is given by (8). Then, the function $h$ is a control barrier function (CBF) if there exists an extended class $\mathcal{K}_\infty$ function $\alpha$ [3] such that

$$\sup_{\mathbf{u}\in\mathcal{U}} [L_f h(\mathbf{x}) + L_g h(\mathbf{x})\mathbf{u} + \alpha(h(\mathbf{x}))] \geq 0, \ \forall \mathbf{x} \in \mathcal{D}, \tag{9}$$

where $L_X p(x)$ is the Lie derivative of a function $p : \mathcal{P} \to \mathbb{R}$ with respect of a vector field $X$ at point $x \in \mathcal{P}$.
We define the set consisting of control inputs that render $\mathcal{C}$ forward invariant as follows,

$$S_{\text{cbf}}(\mathbf{x}) \coloneqq \{\mathbf{u} \in \mathcal{U} \mid L_f h(\mathbf{x}) + L_g h(\mathbf{x})\mathbf{u} + \alpha(h(\mathbf{x})) \geq 0\}. \tag{10}$$

Given such a definition, then we have the following theorem regarding the forward invariance of CBFs.

*Theorem 1 ( [3]):* Let $\mathcal{C} \subseteq \mathcal{D} \subseteq \mathbb{R}^n$ be the set defined in (8). If $h$ is a CBF on $D$ and $\frac{\partial h}{\partial \mathbf{x}}(\mathbf{x}) \neq 0$ for all $\mathbf{x} \in \partial C$, then any Lipschitz continuous controller $\mathbf{u} : \mathcal{D} \to \mathcal{U}$ such that $\mathbf{u} \in S_{\text{cbf}}(\mathbf{x})$ will render the set $\mathcal{C}$ forward invariant, that is, the system is guaranteed to be safe. Moreover, the set $\mathcal{C}$ is asymptotically stable in $\mathcal{D}$.

[1]In [18], they use $\mathbf{s}, \mathbf{a}$ for state and action (equivalently control) as that is typical notation for the reinforcement learning community. In this work, we use $\mathbf{x}, \mathbf{u}$ for state and control which is typical notation in the controls community.

### D. Control Lyapunov Function (CLF)

Like CBFs, CLFs offer similar conditions but for stability.

*Definition 2 (Control Lyapunov Function):* Given dynamics (7), a continuously differentiable function $V : \mathcal{D} \to \mathbb{R}_{\geq 0}$ is a control Lyapunov function if it is positive definite and there exists a class $\mathcal{K}$ function $\gamma$ such that

$$\inf_{\mathbf{u}\in\mathcal{U}} [L_f V(\mathbf{x}) + L_g V(\mathbf{x})\mathbf{u} + \gamma(V(\mathbf{x}))] \leq 0. \tag{11}$$

The condition (11) ensures that there exists a control input $\mathbf{u}$ that can decrease the value of $V$ along the system's trajectories. We define the set consisting of control inputs that stabilize the system as follows,

$$S_{\text{clf}}(\mathbf{x}) \coloneqq \{\mathbf{u} \in \mathcal{U} \mid L_f V(\mathbf{x}) + L_g V(\mathbf{x})\mathbf{u} + \gamma(V(\mathbf{x})) \leq 0\}. \tag{12}$$

Given such a definition, then we have the following theorem regarding stability.

*Theorem 2:* Let $V$ be a CLF for the system (7) on $\mathcal{D}$. Then, there exists a Lipschitz continuous controller $\mathbf{u} \in S_{\text{clf}}(\mathbf{x})$ asymptotically stabilizes the system to the origin.[2]

## IV. Conditional Motion Planning with Diffusion

In this section, we introduce CoBL-Diffusion, a conditional diffusion model designed for robot motion planning in dynamic environments. Our proposed robot planner generates a controller that enables the robot to reach a goal while avoiding collisions with dynamic obstacles (e.g., pedestrians).

### A. Problem Setting

We denote robot's states as $\mathbf{x} = [\mathbf{x}_0, \ \dots, \ \mathbf{x}_T] \in \mathcal{D} \subseteq \mathbb{R}^n$ and control inputs as $\mathbf{u} = [\mathbf{u}_1, \ \dots, \ \mathbf{u}_T] \in \mathcal{U} \subseteq \mathbb{R}^m$, where $T$ represents the time horizon. Assume that there are $Q$ moving obstacles in the environment and denote the state and control of each obstacle at time $t$ as $\mathbf{x}_{q,t}, \mathbf{u}_{q,t}$ for $q = 1, \dots, Q$. Consider discrete-time control affine dynamics of the robot:

$$\mathbf{x}_{t+1} = f_d(\mathbf{x}_t, \mathbf{u}_t) \tag{13}$$

where $f_d : \mathcal{D} \to \mathcal{D}$ is a continuous function. For socially acceptable robot navigation, it is essential not only to avoid obstacles but also to generate human-like movements. Safe and human-like planning in dynamic environments can be formulated as the following trajectory optimization problem:

$$\min_{\mathbf{u}_{0:T}} \quad J_T(\mathbf{x}_T) + \sum_{t=0}^{T} J(\mathbf{x}_t, \mathbf{u}_t) \tag{14}$$

$$\text{s.t.} \quad \mathbf{x}_{t+1} = f_d(\mathbf{x}_t, \mathbf{u}_t) \tag{15}$$

$$c(\mathbf{x}_t, \mathbf{x}_{q,t}, \mathbf{u}_t, \mathbf{u}_{q,t}) \geq 0, \ q = 1, \dots, Q \tag{16}$$

$$\mathbf{x}_t \in \mathcal{D}, \ \mathbf{u}_t \in \mathcal{U}, \ t = 0, \dots, T \tag{17}$$

where $J(\cdot)$ represents a cost function encompassing various planning objectives (e.g., control efficiency, human-likeness, and smoothness), $J_T(\cdot)$ denotes a terminal cost, and $c(\cdot)$ denotes a safety constraint such as avoiding collisions with obstacles.

However, it is nontrivial to specify a general cost function that encodes fluent, human-like, and socially acceptable behaviors; indeed, synthesizing such behavior is a research area itself [37]–[40]. Instead, we seek to generate human-like robot control sequences $\mathbf{u}_{0:T}$ via a generative model trained

[2]Equivalently, an equilibrium state.

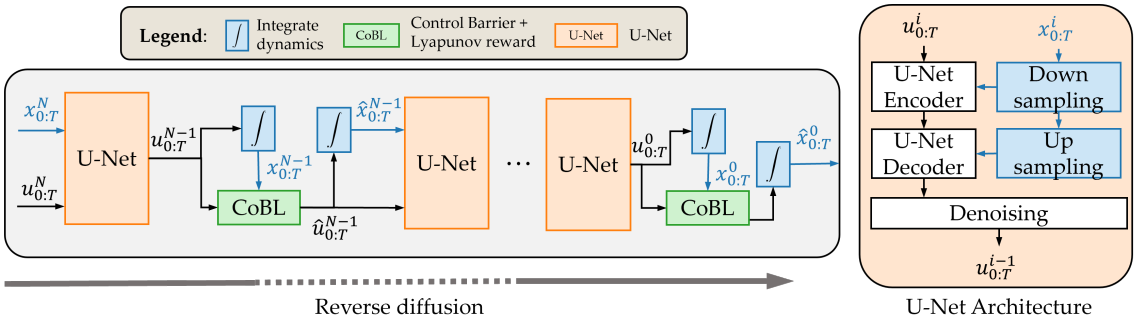



Fig. 2: Illustration of planning with CoBL-Diffusion. The left figure depicts the reverse diffusion process of the proposed model. The right figure illustrates the U-Net architecture employed in the proposed model.

on pedestrian data, while leveraging control-theoretic tools to encode safety constraints and goal objectives, as described by (14)–(17), to be imposed on the robot.

### B. Model Architecture

Our proposed architecture is inspired by Diffuser [18], but there are several key differences. The trajectory generation process of CoBL-Diffusion is shown in Fig. 2

#### 1) U-Net Conditioning on Trajectories

To maintain dynamic consistency, our model generates only control inputs, that is, compared to (4), we have $\boldsymbol{\tau} = \mathbf{u} = [\mathbf{u}_1, \ \ldots, \ \mathbf{u}_T]$, and the states are obtained by feeding the controls into the system dynamics (13) (see the blue box in Fig. 2 left), rather than simultaneously predicting control inputs and states. While Diffuser conditions the goal by replacing the terminal state, the same inpainting method cannot be applied to CoBL-Diffusion since it predicts controls only. Therefore, the states to be satisfied are fed into each resolution of the U-Net [41] (see orange boxes in Fig. 2) to condition the generated controls for the next denoising step. During the training of the proposed model, the states obtained by integrating the control inputs corrupted by noise are fed into the U-Net. This is because the states provided as conditions during the generation process are noisy, obtained by integrating noisy controls in the previous diffusion step with the system dynamics (13).

#### 2) Loss Function for Trajectory Error

Conditioning the U-Net on states is still not sufficient for the diffusion model to generate a controller that reaches the goal. Therefore, in addition to the conventional diffusion loss $L_{\text{simple}}$ (3) computed on the controls alone, we introduce a loss function $L_{\text{traj}}$ that measures the error between the ground truth and denoised trajectories:

$$L_{\text{traj}} = \frac{1}{T+1} \sum_{t=0}^{T} \|\mathbf{x}_t - \hat{\mathbf{x}}_t\|^2, \tag{18}$$

where $x_t$ and $\hat{x}_t$ represent ground truth and generated states at a timestep $t$ respectively. Here, the generated states are explicitly computed from the denoised control sequence and system dynamics. This loss function encourages the synthesis of the controller that achieves the given state at each timestep.

#### 3) Temporal Weighting for Covariance

The behavior of a generated trajectory depends on the sequence of control inputs by integrating the dynamics. As a result, errors in control inputs accumulate over time, meaning errors at earlier timesteps can lead to large errors in states later in the horizon. Therefore, denoising the later control sequence has little effect if the earlier control inputs have not converged. To encourage the convergence of the earlier control sequence, the covariances of the forward diffusion (1) and reverse diffusion (2) are weighted by $\mathbf{V}$ as follows:

$$\mathbf{V} = \text{diag}(v_0^2, \ldots, v_T^2), \tag{19}$$

where $v_0, \ldots, v_T$ are scheduled to increase monotonically with respect to the time steps.

### C. Reward Function

As introduced in Section III-B, a user-defined reward function can be used to guide the reverse diffusion process (5) to generate outputs with some desirable behavior. The reverse diffusion process can be guided by multiple reward functions simultaneously, therefore the optimality of the trajectory is denoted using the set of $K$ reward functions $W_k(\mathbf{x}_t, \mathbf{u}_t)$ as $p(O_t = 1) = \exp\Big(\sum_{k=1}^{K} W_k(\mathbf{x}_t, \mathbf{u}_t)\Big)$. Then, we modify the reverse diffusion process as follows:

$$p_\theta(\mathbf{u}^{i-1}|\mathbf{u}^i, O_{1:T}) \approx \mathcal{N}(\mathbf{u}^{i-1}; \boldsymbol{\mu}_\theta(\mathbf{u}^i, i) + \mathbf{g}, \ \mathbf{V}\boldsymbol{\Sigma}^i), \tag{20}$$

where

$$\mathbf{g} = \nabla_{\mathbf{u}} \log p(O_{1:T}|\mathbf{u})|_{\mathbf{u}=\boldsymbol{\mu}_\theta(\mathbf{u}^i, i)} \tag{21}$$

$$= \sum_{k=1}^{K} \sum_{t=0}^{T} \nabla_{\mathbf{u}_t} W_k(\mathbf{x}_t, \mathbf{u}_t)|_{\mathbf{u}_t=\boldsymbol{\mu}_\theta(\mathbf{u}_t^i, i)}. \tag{22}$$

Following [42], we drop the scaling factor based on the covariance for the $\mathbf{g}$ shown in (20). It addresses the issue of the covariance becoming small towards the end of the reverse diffusion, which would render the guidance ineffective. Next, we present the CBF and CLF reward functions used for guiding the diffusion process to generate a safe and goal-reaching trajectory.

#### 1) Control Barrier Function Reward

The reward function based on control barrier functions (CBFs) (see Section III-C) is designed to guide the reverse diffusion process to generate a safe control sequence that avoids collision with dynamic obstacles. We aim to encourage forward invariance of the safe set in the time domain at each diffusion step. Although the actual dynamics of the robot is discrete given by (13), we assume the robot and obstacles follow the continuous-time dynamics (7). We define pairwise joint dynamics for the robot and each obstacle $q$:

$$\dot{\mathbf{X}}_q = F(\mathbf{X}_q) + G(\mathbf{X}_q)\mathbf{U}_q, \tag{23}$$

where $\mathbf{X}_q = [\mathbf{x}, \ \mathbf{x}_q]^T$ and $\mathbf{U} = [\mathbf{u}, \ \mathbf{u}_q]^T$ denote the joint state and control, and $F = [f(\mathbf{x}), f_q(\mathbf{x}_q)]^T$, $G = \text{diag}(g(\mathbf{x}), g_q(\mathbf{x}_q))$. Suppose the function $h_{\text{cbf}}(\mathbf{X}_q)$ is a CBF

to avoid a collision with the obstacle. From Theorem 1, control input $\mathbf{u}$ that makes the following reward function $W_{\text{cbf}}$ positive ensures forward invariance of the safe set defined by $h_{\text{cbf}}(\mathbf{X}_q) \geq 0$:

$$W_{\text{cbf}}(\mathbf{X}_q, \mathbf{U}_q) = L_F h_{\text{cbf}}(\mathbf{X}_q) + L_G h_{\text{cbf}}(\mathbf{X}_q)\mathbf{U}_q + \alpha(h_{\text{cbf}}(\mathbf{X}_q)).$$

Therefore, if the reward function $W_{\text{cbf}}$ is positive for each control action in the entire sequence, collision avoidance is guaranteed. The gradient of this reward function with respect to the control input $\mathbf{u}$ is computed as follows:

$$\nabla_{\mathbf{u}} W_{\text{cbf}}(\mathbf{X}_q, \mathbf{U}_q) = L_G h_{\text{cbf}}(\mathbf{X}_q)[\mathbf{1}, \mathbf{0}]^T = L_g h_{\text{cbf}}(\mathbf{X}_q).$$

This gradient does not depend on the control input or dynamics of the obstacle and can be computed as long as the obstacle's state is known.

The trajectories learned and generated by the diffusion model are discrete (13). Therefore, it may be more appropriate to use a reward function based on discrete-time CBF [43], [44]. Therefore, we define the reward function based on discrete-time CBF as follows:

$$W_{\text{dcbf}}(\mathbf{X}_{q,t}, \mathbf{U}_{q,t}) = \Delta h_{\text{cbf}}(\mathbf{X}_{q,t}, \mathbf{U}_{q,t}) + \alpha(h_{\text{cbf}}(\mathbf{X}_{q,t}))$$
$$\text{where} \quad \Delta h_{\text{cbf}}(\mathbf{X}_{q,t}, \mathbf{U}_{q,t}) = h_{\text{cbf}}(\mathbf{X}_{q,t+1}) - h_{\text{cbf}}(\mathbf{X}_{q,t}),$$

where $\alpha$ is a class $\mathcal{K}$ function satisfying $\alpha(r) < r$ for all $r > 0$. The performance of continuous-time and discrete-time CBF rewards are compared in Section V.

*2) Control Lyapunov Function Reward*

The reward function based on Control Lyapunov Functions (CLFs) in Section III-D guides the reverse diffusion process to generate a control sequence that encourages convergence to a given goal $\mathbf{x}_g$. Similar to CBF reward, we assume that the dynamics of the robot is (7). Suppose $h_{\text{clf}}$ is a CLF, then, according to Theorem 2, control inputs $\mathbf{u}$ that make the following reward function $W_{\text{clf}}$ positive will drive the state of the robot $\mathbf{x}$ to $\mathbf{x}_g$:

$$W_{\text{clf}}(\mathbf{x}, \mathbf{u}) = -L_f h_{\text{clf}}(\mathbf{x}) - L_g h_{\text{clf}}(\mathbf{x})\mathbf{u} - \alpha(h_{\text{clf}}(\mathbf{x})) \tag{24}$$

The gradient of this reward function with respect to the control input $\mathbf{u}$ is computed as follows:

$$\nabla_{\mathbf{u}} W_{\text{clf}}(\mathbf{x}, \mathbf{u}) = -L_g h_{\text{clf}}(\mathbf{x}) \tag{25}$$

Same as the CBF reward, to compare with the continuous-time CLF reward, we define the discrete-time CLF reward:

$$W_{\text{dclf}}(\mathbf{x}_t, \mathbf{u}_t) = -\Delta h_{\text{clf}}(\mathbf{x}_t, \mathbf{u}_t) - \gamma \|\mathbf{x}_t - \mathbf{x}_g\|^2$$
$$\text{where} \quad \Delta h_{\text{clf}}(\mathbf{x}_t, \mathbf{u}_t) = h_{\text{clf}}(\mathbf{x}_{t+1}) - h_{\text{clf}}(\mathbf{x}_t),\ \gamma > 0.$$

### D. Conditional Motion Planning with Reverse Diffusion

In this section, we describe how to guide the reverse diffusion process to generate a safe and goal-reaching controller in dynamic environments by evaluating the trajectory with the reward functions designed in Section IV-C. The proposed algorithm is shown in Algorithm 1.

First, we observe start $\mathbf{x}_s$ and goal $\mathbf{x}_g$ of the trajectory, then initialize the trajectory $\mathbf{x}^N$, where $N$ is the number of diffusion steps. If there is prior information about the environment, we initialize based on that; otherwise, we simply initialize the trajectory as a straight line connecting the start and goal points. Next, we sample $\mathbf{u}^{N-1}$ conditioned on $\mathbf{x}^N$ according to the following distribution:

$$\mathbf{u}^{N-1} \sim \mathcal{N}(\boldsymbol{\mu}_\theta(\mathbf{u}^N,\ \mathbf{x}^N),\ \mathbf{V}\boldsymbol{\Sigma}^N). \tag{26}$$

**Algorithm 1** Conditional Planning with CoBL-Diffusion

1: Observe start $\mathbf{x}_s$ and goal $\mathbf{x}_g$
2: initialize controller $\mathbf{u}^N \sim \mathcal{N}(0,\ \mathbf{VI})$
3: initialize trajectory $\mathbf{x}^N$
4: $\mathbf{u}^{N-1} \sim \mathcal{N}(\boldsymbol{\mu}_\theta(\mathbf{u}^N,\ \mathbf{x}^N), \mathbf{V}\boldsymbol{\Sigma}^N)$
5: **for** $i = N-1, \ldots, 1$ **do**
6: Compute $\mathbf{x}^i$ by $\mathbf{x}_s,\ \mathbf{u}^i$ and dynamics
7: `// evaluate the reward functions`
8: $\mathbf{g} = \sum_{k=1}^{K} \sum_{t=0}^{T} \nabla_{\mathbf{u}_t^i} W_k(\mathbf{x}_t^i, \mathbf{u}_t^i)$
9: Check $W_k(\mathbf{x}_t^i, \mathbf{u}_t^i)$ for positive and mask $\mathbf{g}$
10: `// apply the guidance of rewards`
11: $\hat{\mathbf{u}}^i = \mathbf{u}^i + \mathbf{g}$
12: Compute $\hat{\mathbf{x}}^i$ by $\mathbf{x}_s,\ \hat{\mathbf{u}}^i$ and dynamics
13: `// constrain the start and goal points`
14: $\hat{\mathbf{x}}_0^i,\ \hat{\mathbf{x}}_T^i \leftarrow \mathbf{x}_s,\ \mathbf{x}_g$
15: `// execute the denoising step`
16: $\mathbf{u}^{i-1} \sim \mathcal{N}(\boldsymbol{\mu}_\theta(\hat{\mathbf{u}}^i, \hat{\mathbf{x}}^i),\ \mathbf{V}\boldsymbol{\Sigma}^i)$
17: **end for**

Then, the control inputs are fed into the system dynamics (13) to obtain the trajectory $\mathbf{x}^{N-1}$ as in line 6 of Algorithm 1. At this point, it is important to emphasize that, unlike conventional diffusion model-based trajectory generation, the control sequence and states are consistent. Then, the trajectory is evaluated by the reward functions defined in Section IV-C. If each reward function is positive at a certain time step, it indicates the trajectory already satisfies each condition at that time. Therefore, the gradient of the reward function is masked with 0 at that point. Before fed the updated states, initial and terminal states are replaced with the given start $\mathbf{x}_s$ and goal $\mathbf{x}_g$ to condition the controls as in line 10 of Algorithm 1. Then, the controls $\hat{\mathbf{u}}^i$ are denoised based on the states $\hat{\mathbf{x}}^i$. This approach aims to iteratively generate controls that gradually satisfy the conditions, instead of employing a common approach of solving a quadratic program (QP) to enforce satisfaction of CBF/CLF constraints which can be computationally expensive over many diffusion and timesteps. Since there is no need to solve a QP, there is no concern about the feasibility of the trajectories satisfying the constraints. When using multiple reward functions, the prioritization of conditions can be managed by adjusting their coefficients. For example, safety constraints should be emphasized over other constraints, thus necessitating a higher coefficient.

## V. Experiments and Discussion

In this section, we validate the effectiveness of the proposed CoBL-Diffusion to synthesize safe and goal-reaching controllers in dynamic environments. First, we investigate the characteristics of different planners in a single-agent environment. Then, we validate the efficacy in a multi-agent environment using the UCY pedestrian dataset [45].

### A. Model Training and Environment Setup

We trained CoBL-Diffusion using the ETH pedestrian dataset [46]; we trained for 500 epochs on 276,874 8-second trajectories. The architecture and hyperparameters of the diffusion model are based on an open-source implementation[3]. The Adam optimizer [47] was used for training with a learning rate set to $2\times10^{-5}$ and $\beta = (0.9, 0.999)$. The covariance weight (19) was set to increase linearly from 0.5

[3] https://github.com/jannerm/diffuser

to 1. We evaluated the proposed model in two simulation environments created using Trajdata [48]. The first simulation is a toy environment with a single moving obstacle. The second environment is from the UCY pedestrian dataset [45] for challenging multi-agent yet feasible settings.

We assumed the robot follows single integrator dynamics, $\dot{\mathbf{x}} = \mathbf{u}$. We use the following CBF and CLF to guide the reverse diffusion process:

$$h_{\text{cbf}}(\mathbf{X}_q) = (x - x_q)^2 + (y - y_q)^2 - r^2,$$
$$h_{\text{clf}}(\mathbf{x}) = \|\mathbf{x} - \mathbf{x}_g\|^2,$$

where $\mathbf{x} = [x, y]$, $\mathbf{x}_q = [x_q, y_q]$ represent the states of the robot and the $q$-th obstacle respectively, $r$ is a barrier radius, and $\mathbf{x}_g = [x_g, y_g]$ is a given goal point.

### B. Comparison Methods and Evaluation Metrics

We compare **CoBL-Diffusion (CoBL)** with the baseline method **CBF-QP**, which solves a QP with CBF at every timestep given a nominal straight line trajectory to the goal, **Velocity Obstacle (VO)** [49], which is a geometric approach for collision avoidance that uses the relative velocity between a robot and an obstacle to identify potential collision velocities, and **Diffuser** [18], which is a planning with a diffusion model and the generated plan is modified by CBF-QP to ensure collision avoidance. Variants of the proposed CoBL-Diffusion are also investigated: **dCoBL-Diffusion (dCoBL)**, which uses discrete-time CBF and CLF rewards; **CoB-Diffusion (CoB)**, which only uses a CBF reward; **DCoL-Diffusion (DCoL)**, which uses CLF and a distance reward $W_{\text{dist}}$ instead of a CBF reward:

$$W_{\text{dist}}(\mathbf{x}_t, \mathbf{u}_t) = \|\mathbf{x}_t - \mathbf{x}_{q,t}\|^2 - r^2,$$

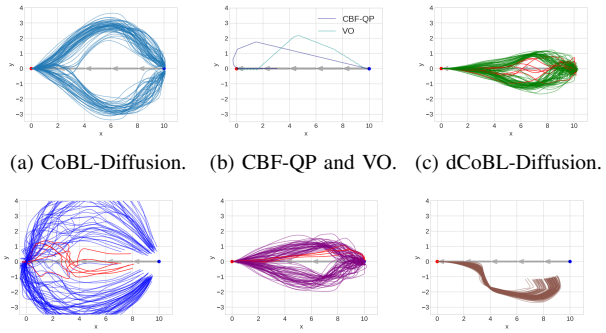
where $r$ is a barrier radius. Comparing the CBF reward, the distance reward considers only the distance to obstacles and ignores the dynamics. The coefficients for (discrete-time) CBF and CLF rewards are fixed to $0.3$ and $0.1$, and for distance reward is fixed to $0.3$. In this experiment, we set the barrier radius to $1$ $m$ and the collision radius to $1$ $m$. For a clear comparison, the reward functions for collision avoidance consider only the nearest obstacle, although considering multiple obstacles could potentially improve performance and is deferred for future work.

Each method is evaluated based on three metrics: collision rate, goal-reaching, and smoothness. The collision rate is calculated as the ratio of simulations that violate the collision radius among all simulations. The goal-reaching is evaluated by the root squared error between the goal and the final point. The smoothness is measured by the worst-case root squared difference in control inputs (in these experiments, velocity).

### C. Safe Planning in Single-Agent Environment

We evaluate the diffusion model guided by each reward function in a single-agent environment. The environment contains an obstacle moving from location $(10, 0)$ to $(0, 0)$ (right to left). The robot aims to reach location $(10, 0)$ from $(0, 0)$ (left to right) while avoiding the oncoming obstacle. The position and velocity of the obstacle are provided in advance for generating a plan. The initial trajectory given at the beginning of reverse diffusion is a straight line connecting the given start and goal points, and is therefore infeasible as it collides with the obstacle.

The results of simulations are summarized in Table I, with the trajectories illustrated in Fig. 3. Note that the CBF-QP and VO methods are deterministic. The CoBL successfully generated collision-free and smooth plans. While the CBF-QP and VO methods successfully avoided obstacles, they produced less smooth trajectories, with VO also exhibiting the poor goal-reaching performance. Although the Diffuser avoided collisions with CBF-QP, it resulted in the worst deviation from the goal. Ablation studies revealed that using discrete-time rewards to adapt to system dynamics resulted in a decline in the safety of the generated plans. In terms of goal-reaching performance, CoBL was comparable with the other CoBL variants; DCoL achieved the highest goal-reaching performance but was outperformed by CoBL in terms of collision avoidance and smoothness. Fig. 3 indicates that dCoBL tends to generate more aggressive trajectories, likely due to the stronger influence of the CLF in the discrete-time formulation. Comparing CoBL and DCoL, using a reward function derived from CBF ensures stronger safety than merely considering the distance from obstacles. When comparing CoBL with CoB, the contribution of the CLF reward to the goal convergence performance is not evident in the simple environment. Even without the CLF reward, the proposed diffusion model's ability to synthesize a controller to achieve the given states allows it to reach the goal. We next investigate CoBL-Diffusion in a dynamic multi-agent scenario.




(a) CoBL-Diffusion. (b) CBF-QP and VO. (c) dCoBL-Diffusion.

(d) CoB-Diffusion. (e) DCoL-Diffusion. (f) Diffuser.

Fig. 3: Generated trajectories in the single-agent environment with various planning methods. Red trajectories experience collisions.

### D. Safe Planning in Real Crowd Environment

In this section, we evaluate the performance of our proposed model in a multi-agent environment from a real-world pedestrian dataset. We selected a scene from the UCY pedestrian dataset [45] that provides a challenging yet comprehensible environment for safe trajectory generation. The environment used for the simulation has eight pedestrians, as shown in Fig. 4. We selected eight goal locations and ran 200 simulations, 25 for each goal.

**Full trajectory knowledge.** First, we assume that the robot knows the trajectories of all humans in the field for the entire horizon. In this setting, the coefficients of CBF, CLF, and Distance reward were set to $0.3$, $0.01$, and $0.3$, respectively to emphasize safety constraints since the number of obstacles increases and the complexity of movement becomes more challenging for ensuring safety. The CBF barrier radius was set to $1$ $m$, providing a buffer for a collision radius of $0.7$ $m$. We included **CoBL-Diffusion$^-$ (CoBL$^-$)** without covariance weight in our evaluation to investigate the effect of the weight on covariance (19) in encouraging convergence of the earlier control sequence. The results of each model

TABLE I: Simulation results for the single and multi-agent environments.

| Planner | Reward | | Single-Agent environment | | | Multi-Agent environment | | |
|---|---|---|---|---|---|---|---|---|
| | Safety | Goal | Coll. (↓) | Goal-Reach. (↓) | Smoothness (↓) | Coll. (↓) | Goal-Reach. (↓) | Smoothness (↓) |
| **CoBL** | $W_{\text{CBF}}$ | $W_{\text{CLF}}$ | **0 %** | $\mathbf{0.20 \pm 0.10}$ | $\mathbf{0.45 \pm 0.16}$ | **0.5 %** | $\mathbf{0.41 \pm 0.20}$ | $\mathbf{0.65 \pm 0.31}$ |
| CoB | $W_{\text{CBF}}$ | – | 6 % | $1.36 \pm 0.51$ | $2.01 \pm 0.19$ | 20.0 % | $3.60 \pm 1.74$ | $1.66 \pm 0.45$ |
| dCoBL | $W_{\text{dCBF}}$ | $W_{\text{dCLF}}$ | 8 % | $0.18 \pm 0.08$ | $0.53 \pm 0.15$ | 48.0 % | $0.38 \pm 0.16$ | $0.45 \pm 0.14$ |
| DCoL | $W_{\text{dist}}$ | $W_{\text{CLF}}$ | 4 % | $0.13 \pm 0.06$ | $0.60 \pm 0.24$ | 42.5 % | $1.10 \pm 1.89$ | $1.52 \pm 1.90$ |
| CoBL$^-$ | $W_{\text{CBF}}$ | $W_{\text{CLF}}$ | – | – | – | 8.5 % | $0.48 \pm 0.34$ | $0.73 \pm 0.39$ |
| CBF-QP | – | – | 0 % | 0.13 | 10.78 | 62.5 % | $0.07 \pm 0.02$ | $4.98 \pm 4.20$ |
| VO | – | – | 0 % | 0.33 | 2.00 | 0 % | $0.31 \pm 0.54$ | $1.78 \pm 0.32$ |
| Diffuser | – | – | 0 % | $2.06 \pm 0.49$ | $0.78 \pm 0.04$ | 29 % | $7.94 \pm 3.22$ | $1.74 \pm 0.31$ |
| Human | – | – | – | – | – | 74.5 % | 0 | $0.08 \pm 0.04$ |

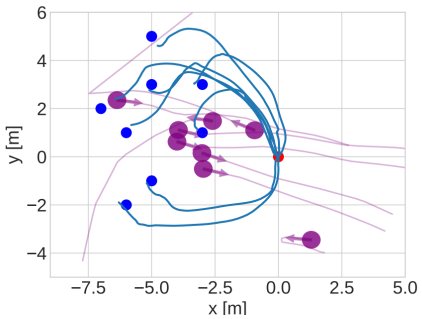



Fig. 4: Visualization of the planning environment in a realistic dynamic setting with pedestrians. The initial position of the robot is marked by a red dot, and the target goals are indicated by blue dots. The generated trajectories are represented by blue lines, while the purple circles and lines represent the humans and their respective trajectories.

are summarized in Table I. For reference, the actual 200 pedestrian data with $3 - 8$ $m$ movements are shown as Human. The high collision rate of Human is attributed to the defined collision radius being larger than the actual one.

CoBL demonstrated its ability to generate safe plans even in complex environments with multiple moving people, with an average generation time of 4.8 seconds. On the other hand, CBF-QP is unable to avoid moving obstacles in a dynamic multi-agent environment. Although the trajectories generated by VO can safely navigate around multiple obstacles, they often lack smoothness and may become impractical or unpredictable for surrounding humans. The result of the Diffuser has the worst distance from the goal because they are modified by the CBF-QP after generating a plan. In contrast, our model integrates collision avoidance into the planning process, enabling the synthesis of safe trajectories from the outset. Similar to simulations in the single-agent environment, using dCoBL or DCoL rewards makes it challenging to ensure safety. Comparing CoBL with CoB, while equivalent quality trajectories were generated in the simple environment, the contribution of CLF reward to goal-reaching is more prominent in this more complex setting; the trajectory is constantly modified by the CBF reward, leading to deviations from the goal without the CLF reward. When comparing the CoBL with the CoBL$^-$, it seems that the temporal weighting introduced for the covariance (19) to promote convergence of the earlier control sequence slightly reduces the collision rate; these results prompt further investigation to better understand the impact of the temporal weighting.

**Partial trajectory knowledge.** Next, we consider a more realistic scenario where the robot only knows the past trajectories of the humans in the field. In this simulation, we make a constant velocity assumption—obstacles continue to move straight with their last known position and velocity. The results are shown in Table II for the following scenarios: FullTraj, where the robot knows the entire 8-second trajectories of the obstacles; FourSecTraj, where only the first 4 seconds of the trajectory is known (and a constant velocity assumption is made after 4 seconds); TwoSecTraj, where only the first 2-seconds of the trajectory is known; and InitTraj, where only the current position and velocity of the obstacles. We see that for CoBL, it is found that assuming the future motion of obstacles as constant linear motion does not significantly compromise safety guarantees—we still outperform the other baseline methods and CoBL variants from Table I. It is important to note that in this scene, the obstacles were humans and approximating their movement as constant linear was adequate for a short period. It is expected that combining our model with more advanced motion prediction models can ensure stronger safety, and we plan on investigating this in future work.

TABLE II: Simulation results for varying levels of knowledge in the multi-agent environment.

| Scenario | Collision | Goal-Reaching | Smoothness |
|---|---|---|---|
| FullTraj | 1 % | $0.41 \pm 0.20$ | $0.74 \pm 0.34$ |
| FourSecTraj | 0.5 % | $0.40 \pm 0.20$ | $0.68 \pm 0.29$ |
| TwoSecTraj | 4 % | $0.36 \pm 0.20$ | $0.67 \pm 0.32$ |
| InitTraj | 19.5 % | $0.45 \pm 0.30$ | $0.67 \pm 0.32$ |

## VI. Conclusion and Future Directions

Our proposed CoBL-Diffusion synthesizes robot controllers for safe planning in dynamic multi-agent environments. Algorithmically, CoBL-Diffusion guides the reverse diffusion process with reward functions based on CBF and CLF. During the denoising process, CoBL-Diffusion iteratively improves the control sequence towards satisfying the safety and stability constraints. To evaluate the efficacy of the proposed model for safe planning in dynamic environments, we compared it with existing methods and variants of our proposal. We found that CoBL-Diffusion can smoothly reach goal locations within an acceptable distance while having very low collision rates. However, the collision rates increased as the uncertainty in the behaviors of obstacles increased. These results prompt the need for further investigation in methods to incorporate behavior prediction within the diffusion model. Additionally, to promote real-time applications, we plan to explore how to reduce the inference time of our models, such as leveraging ideas from DDIM [50] and consistency models [51], and investigate the effectiveness of our conditioning approach with them.

## REFERENCES


[1] M. Althoff and J. Dolan, "Online verification of automated road vehicles using reachability analysis," *IEEE Transactions on Robotics*, vol. 30, no. 4, pp. 903–918, 2014.

[2] M. Althoff, G. Frehse, and A. Girard, "Set propagation techniques for reachability analysis," *Annual Review of Control, Robotics, and Autonomous Systems*, vol. 4, no. 1, pp. 369–395, 2021.

[3] A. D. Ames, S. Coogan, M. Egerstedt, G. Notomista, K. Sreenath, and P. Tabuada, "Control barrier functions: Theory and applications," in *European Control Conference*, 2019.

[4] I. M. Mitchell, A. M. Bayen, and C. J. Tomlin, "A time-dependent Hamilton-Jacobi formulation of reachable sets for continuous dynamic games," *IEEE Transactions on Automatic Control*, vol. 50, no. 7, pp. 947–957, 2005.

[5] K. Leung, E. Schmerling, M. Zhang, M. Chen, J. Talbot, J. C. Gerdes, and M. Pavone, "On infusing reachability-based safety assurance within planning frameworks for human-robot vehicle interactions," *Int. Journal of Robotics Research*, vol. 39, pp. 1326–1345, 2020.

[6] S. Kousik, P. Holmes, and R. Vasudevan, "Safe, aggressive quadrotor flight via reachability-based trajectory design," in *Proc. ASME Dynamic Systems and Control Conference*, 2019.

[7] M. Chen and C. J. Tomlin, "Hamilton–Jacobi reachability: Some recent theoretical advances and applications in unmanned airspace management," *Annual Review of Control, Robotics, and Autonomous Systems*, vol. 1, no. 1, pp. 333–358, 2018.

[8] N. Rhinehart, R. McAllister, and S. Levine, "Deep imitative models for flexible inference, planning, and control," in *Int. Conf. on Learning Representations*, 2020.

[9] B. Ivanovic, K. Leung, E. Schmerling, and M. Pavone, "Multimodal deep generative models for trajectory prediction: A conditional variational autoencoder approach," *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 295–302, 2021.

[10] W. Xiao, T. Wang, R. Hasani, M. Chahine, A. Amini, X. Li, and D. Rus, "BarrierNet: Differentiable control barrier functions for learning of safe robot control," *IEEE Transactions on Robotics*, 2023.

[11] S. Yang, S. Chen, V. M. Preciado, and R. Mangharam, "Differentiable safe controller design through control barrier functions," *IEEE Control Systems Letters*, vol. 7, pp. 1207–1212, 2022.

[12] W. Xiao, T.-H. Wang, M. Chahine, A. Amini, R. Hasani, and D. Rus, "Differentiable control barrier functions for vision-based end-to-end autonomous driving," *Available at https://arxiv.org/abs/2203.02401*, 2022.

[13] T. Phan-Minh, F. Howington, T.-S. Chu, S. U. Lee, M. S. Tomov, N. Li, C. Dicle, S. Findler, F. Suarez-Ruiz, R. Beaudoin, B. Yang, S. Omari, and E. M. Wolff, "DriveIRL: Drive in real life with inverse reinforcement learning," in *Proc. IEEE Conf. on Robotics and Automation*, 2023.

[14] J. Sohl-Dickstein, E. Weiss, N. Maheswaranathan, and S. Ganguli, "Deep unsupervised learning using nonequilibrium thermodynamics," in *Int. Conf. on Machine Learning*, 2015.

[15] J. Ho, A. Jain, and P. Abbeel, "Denoising diffusion probabilistic models," in *Conf. on Neural Information Processing Systems*, 2020.

[16] R. Rombach, A. Blattmann, D. Lorenz, and O. B., "High-resolution image synthesis with latent diffusion models," in *IEEE Conf. on Computer Vision and Pattern Recognition*, 2022.

[17] R. Huang, Z. Zhao, H. Liu, J. Liu, C. Cui, and Y. Ren, "Prodiff: Progressive fast diffusion model for high-quality text-to-speech," in *ACM Int. Conf. on Multimedia*, 2022.

[18] M. Janner, Y. Du, J. Tenenbaum, and S. Levine, "Planning with diffusion for flexible behavior synthesis," in *Int. Conf. on Machine Learning*, 2022.

[19] Z. Zhong, D. Rempe, D. Xu, Y. Chen, S. Veer, T. Che, B. Ray, and M. Pavone, "Guided conditional diffusion for controllable traffic simulation," in *Proc. IEEE Conf. on Robotics and Automation*, 2023.

[20] U. Mishra, S. Xue, Y. Chen, and D. Xu, "Generative skill chaining: Long-horizon skill planning with diffusion models," in *Conf. on Robot Learning*, 2023.

[21] C. Jiang, A. Cornman, C. Park, B. Sapp, Y. Zhou, and D. Anguelov, "Motiondiffuser: Controllable multi-agent motion prediction using diffusion," in *IEEE Conf. on Computer Vision and Pattern Recognition*, 2023.

[22] M. Srinivasan, A. Dabholkar, S. Coogan, and P. Vela, "Synthesis of control barrier functions using a supervised machine learning approach," in *IEEE/RSJ Int. Conf. on Intelligent Robots & Systems*, 2020.

[23] M. Khan, T. Ibuki, and A. Chatterjee, "Gaussian control barrier functions: Non-parametric paradigm to safety," *IEEE Access*, 2022.

[24] K. Mizuta, Y. Hirohata, J. Yamauchi, and M. Fujita, "Safe persistent coverage control with control barrier functions based on sparse bayesian learning," in *IEEE Conf. on Control Technology and Applications*, 2022.

[25] K. Long, C. Qian, J. Cortes, and N. Atanasov, "Learning barrier functions with memory for robust safe navigation," *IEEE Robotics and Automation Letters*, 2021.

[26] W. Cortez, D. Oetomo, C. Manzie, and P. Choong, "Control barrier functions for mechanical systems: Theory and application to robotic grasping," *IEEE Transactions on Control Systems Technology*, 2019.

[27] R. Grandia, A. Taylor, A. Ames, and M. Hutter, "Multi-layered safety for legged robots via control barrier functions and model predictive control," in *Proc. IEEE Conf. on Robotics and Automation*, 2021.

[28] H. Yu, C. Hirayama, C. Yu, S. Herbert, and S. Gao, "Sequential neural barriers for scalable dynamic obstacle avoidance," in *IEEE/RSJ Int. Conf. on Intelligent Robots & Systems*, 2023.

[29] Z. Qin, K. Zhang, Y. Chen, J. Chen, and C. Fan, "Learning safe multi-agent control with decentralized neural barrier certificates," in *Int. Conf. on Learning Representations*, 2021.

[30] M. Pereira, Z. Wang, I. Exarchos, and E. Theodorou, "Safe optimal control using stochastic barrier functions and deep forward-backward SDEs," in *Conf. on Robot Learning*, 2021.

[31] K. P. Wabersich, A. J. Taylor, J. J. Choi, K. Sreenath, C. J. Tomlin, A. D. Ames, and M. N. Zeilinger, "Data-driven safety filters: Hamilton-Jacobi reachability, control barrier functions, and predictive methods for uncertain systems," *IEEE Control Systems Magazine*, vol. 43, no. 5, pp. 137–177, 2023.

[32] W. Xiao, T. Wang, C. Gan, and D. Rus, "SafeDiffuser: Safe planning with diffusion probabilistic models," *Available at https://arxiv.org/abs/2306.00148*, 2023.

[33] C. Dawson, S. Gao, and C. Fan, "Safe control with learned certificates: A survey of neural lyapunov, barrier, and contraction methods," *Available at https://arxiv.org/abs/2202.11762*, 2022.

[34] Y. Chang, N. Roohi, and S. Gao, "Neural lyapunov control," in *Conf. on Neural Information Processing Systems*, 2019.

[35] A. Nichol and P. Dhariwal, "Improved denoising diffusion probabilistic models," in *Int. Conf. on Machine Learning*, 2021.

[36] P. Dhariwal and A. Nichol, "Diffusion models beat gans on image synthesis," *Advance in Neural Information Processing Systems*, 2021.

[37] W. Schwarting, A. Pierson, J. Alonso-Mora, S. Karaman, and D. Rus, "Social behavior for autonomous vehicles," *Proceedings of the National Academy of Sciences*, vol. 116, no. 50, pp. 24 972–24 978, 2019.

[38] C. I. Mavrogiannis, P. Alves-Oliveira, W. Thomason, and R. A. Knepper, "Social momentum: Design and evaluation of a framework for socially competent robot navigation," *ACM Transactions on Human-Robot Interaction*, vol. 37, no. 4, 2021.

[39] J. Geldenbott and K. Leung, "Legible and proactive robot planning for prosocial human-robot interactions," in *Proc. IEEE Conf. on Robotics and Automation*, 2024, (accepted).

[40] H. Kretzschmar, M. Spies, C. Sprunk, and W. Burgard, "Socially compliant mobile robot navigation via inverse reinforcement learning," *Int. Journal of Robotics Research*, vol. 35, no. 11, pp. 1289–1307, 2016.

[41] P. Ronneberger, O. Fischer and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *Int. Conf. on Medical Image Computing and Computer-Assisted Intervention*, 2015.

[42] J. Carvalho, A. Le, M. Baierl, D. Koert, and J. Peters, "Motion planning diffusion: Learning and planning of robot motions with diffusion models," in *IEEE/RSJ Int. Conf. on Intelligent Robots & Systems*, 2023.

[43] A. Agrawal and K. Sreenath, "Discrete control barrier functions for safety-critical control of discrete systems with application to bipedal robot navigation," in *Robotics: Science and Systems*, 2017.

[44] Y. Xiong, D. Zhai, M. Tavakoli, and Y. Xia, "Discrete-time control barrier function: High-order case and adaptive case," *IEEE Transactions on Cybernetics*, 2022.

[45] A. Lerner, Y. Chrysanthou, and D. Lischinski, "Crowds by example," *Computer Graphics Forum*, vol. 26, no. 3, pp. 655–664, 2007.

[46] S. Pellegrini, A. Ess, Schindler, and L. Gool, "You'll never walk alone: Modeling social behavior for multi-target tracking," in *IEEE Int. Conf. on Computer Vision*, 2009.

[47] D. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Int. Conf. on Learning Representations*, 2015.

[48] B. Ivanovic, G. Song, I. Gilitschenski, and M. Pavone, "trajdata: A unified interface to multiple human trajectory datasets," in *Conf. on Neural Information Processing Systems*, 2023.

[49] P. Fiorini and Z. Shiller, "Motion planning in dynamic environments using velocity obstacles," *Int. Journal of Robotics Research*, vol. 17, no. 10-11, pp. 760–772, 1998.

[50] J. Song, C. Meng, and S. Ermon, "Denoising diffusion implicit models," in *Int. Conf. on Learning Representations*, 2021.

[51] Y. Song, P. Dhariwal, and I. Sutskever, "Consistency models," in *Int. Conf. on Machine Learning*, 2023.