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

Kazuki Mizuta Karen Leung

***Abstract*— Equipping autonomous robots with the ability to navigate safely 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 the proposed model using a real-world pedestrian dataset, showing that it generates smooth trajectories that enable the robot to reach a goal 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. Diffusion models [10], [11] have emerged as a powerful class of deep generative models, achieving remarkable success in generating high-quality images and synthesizing speech [12], [13], and more recently, in robot trajectory generation [14]–[17]. However, current diffusion-based robot trajectory generation methods still lack explicit safety assurances in dynamic environments, prohibiting their application in safety-critical scenarios, such as those involving human interactions. 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.

Control Barrier Functions (CBFs) and Control Lyapunov Functions (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 [18]–[20].

Kazuki Mizuta is partially supported by the Nakajima Foundation.

University of Washington, Department of Aeronautics and Astronautics
{mizuta, kymleung}@uw.edu

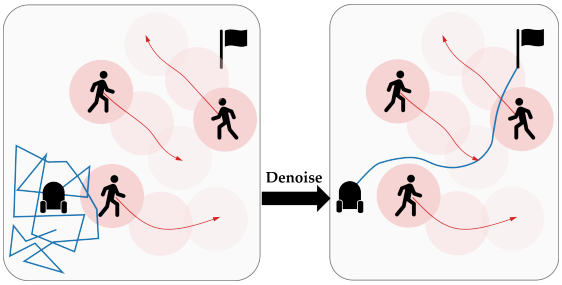

Fig. 1: CoBL-Diffusion uses control barrier and Lyapunov functions to guide a diffusion process to generate a robot controller for goal-reaching while avoiding dynamic obstacles.

In this paper, inspired by the Diffuser introduced in [14] 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 and Control **L**yapunov Functions for the enforcement of desired safety and stability (i.e., goal-reaching) properties. We present the architecture behind CoBL-Diffusion that generates dynamically feasible trajectories that satisfy the CBF and CLF constraints.

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

1) We propose **CoBL-Diffusion**, a novel 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 CBF and CLF theory.
2) Our method ensures dynamic consistency between the control sequence and the resulting states by explicitly integrating generated states and controls through system dynamics.
3) We demonstrate the effectiveness of our approach in generating safe robot motion plans for navigating through crowded dynamic environments.

II. 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.

A. Problem Setting

We denote robot's states as $\mathbf{x} = [\mathbf{x}_0, \ \ldots, \ \mathbf{x}_T] \in \mathcal{D} \subseteq \mathbb{R}^n$ and control inputs as $\mathbf{u} = [\mathbf{u}_1, \ \ldots, \ \mathbf{u}_T] \in \mathcal{U} \subseteq \mathbb{R}^m$, 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 $q =$

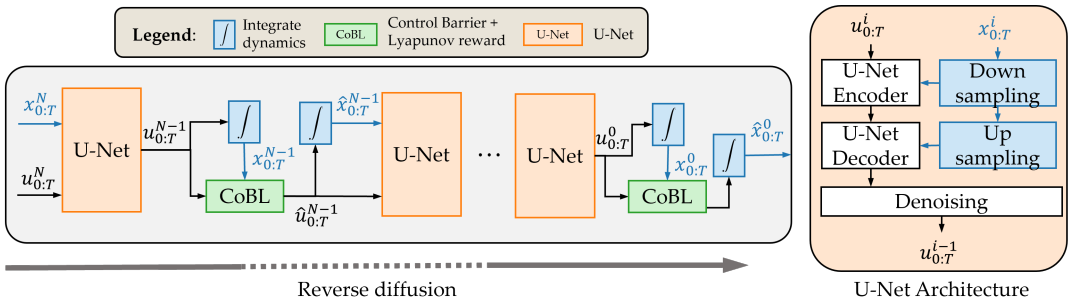

Fig. 2: Illustration of planning with CoBL-Diffusion. The left figure depicts the reverse diffusion process of the proposed model, with the images showing the generated trajectories. The right figure illustrates the U-Net architecture employed in the proposed model.

$1, \ldots, Q$. Consider discrete-time control affine dynamics of the robot:

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

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:

$$\min_{\mathbf{u}_{0:T}} \quad J_T(\mathbf{x}_T) + \sum_{t=0}^{T} J(\mathbf{x}_t, \mathbf{u}_t) \tag{2}$$

$$\text{s.t.} \quad \mathbf{x}_{t+1} = f_d(\mathbf{x}_t, \mathbf{u}_t) \tag{3}$$

$$c(\mathbf{x}_t, \mathbf{x}_{q,t}, \mathbf{u}_t, \mathbf{u}_{q,t}) \geq 0,\ q = 1, \ldots, Q \tag{4}$$

$$\mathbf{x}_t \in \mathcal{D},\ \mathbf{u}_t \in \mathcal{U},\ t = 0, \ldots, T \tag{5}$$

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 [21]–[24]. Instead, we seek to generate human-like robot control sequences $\mathbf{u}_{0:T}$ via a generative model trained on pedestrian data, while leveraging control-theoretic tools to encode safety constraints and goal objectives, as described by (2)–(5), to be imposed on the robot.

B. Model Architecture

Our proposed architecture is inspired by Diffuser [14], 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 and the states are obtained by feeding the controls into the system dynamics (1), 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 [25] to condition the diffusion process. During the training of the proposed model, ground truth states are corrupted with the same noise as the control inputs in the forward diffusion process. This is because, during the generation process, the states obtained by integrating noisy controls with the system dynamics (1) are noisy.

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 computed on the controls alone, we introduce a loss function L_{traj} that measures the error between the ground truth and denoised trajectories:

$$L_{\text{traj}} = \frac{1}{T+1} \sum_{t=0}^{T} \|\mathbf{x}_t - \hat{\mathbf{x}}_t\|^2, \tag{6}$$

where x_t and $\hat{x}_t$ represent ground truth and generated states at a timestep t respectively. 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 trajectory is determined by the generated control sequence. 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 and reverse diffusion are weighted by $\mathbf{V}$ as follows:

$$\mathbf{V} = \text{diag}(v_0^2, \ldots, v_T^2), \tag{7}$$

where $v_0, \ldots, v_T$ are scheduled to increase monotonically.

C. Reward Function

In [14], a user-defined reward function can be used to guide the reverse diffusion process to generate outputs with some desirable behavior. In this work, 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 $p(O_t = 1) = \exp\left(\sum_{k=1}^{K} W_k(\mathbf{x}_t, \mathbf{u}_t)\right)$. Then, we modify the reverse diffusion process as follows:

$$p_\theta(\mathbf{u}^{i-1}|\mathbf{u}^i, O_{1:T}) \approx \mathcal{N}(\mathbf{u}^{i-1}; \boldsymbol{\mu}_\theta(\mathbf{u}^i, i) + \mathbf{g},\ \mathbf{V}\boldsymbol{\Sigma}^i), \tag{8}$$

where

$$\mathbf{g} = \nabla_{\mathbf{u}} \log\, p(O_{1:T}|\mathbf{u})|_{\mathbf{u}=\boldsymbol{\mu}_\theta(\mathbf{u}^i, i)} \tag{9}$$

$$= \sum_{k=1}^{K} \sum_{t=0}^{T} \nabla_{\mathbf{u}_t} W_k(\mathbf{x}_t, \mathbf{u}_t)|_{\mathbf{u}_t=\boldsymbol{\mu}_\theta(\mathbf{u}_t^i, i)}. \tag{10}$$

Following [26], we drop the scaling factor based on the covariance for the $\mathbf{g}$ shown in (8). 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) 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 (1), we assume the robot and obstacles follow the continuous-time dynamics:

$$\dot{\mathbf{x}} = f(\mathbf{x}) + g(\mathbf{x})\mathbf{u}, \tag{11}$$

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. We define pairwise joint dynamics for the robot and each obstacle q:

$$\dot{\mathbf{X}}_q = F(\mathbf{X}_q) + G(\mathbf{X}_q)\mathbf{U}_q, \tag{12}$$

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 $F = [f(\mathbf{x}), f_q(\mathbf{x}_q)]^T$, $G = \mathrm{diag}(g(\mathbf{x}), g_q(\mathbf{x}_q))$. Suppose the function $h_{\mathrm{cbf}}(\mathbf{X}_q)$ is a CBF to avoid a collision with the obstacle. Control inputs $\mathbf{u}$ that make the following reward function W_{cbf} positive ensure forward invariance of the safe set defined by $h_{\mathrm{cbf}}(\mathbf{X}_q) \geq 0$:

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

where $\alpha(\cdot)$ is an extended class $\mathcal{K}_\infty$ function [3]. Therefore, if the reward function W_{cbf} can be made positive at the end of the denoising process, it is guaranteed that the generated control sequence will avoid a collision with the obstacle. The gradient of this reward function with respect to the control input $\mathbf{u}$ is computed as follows:

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

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 (1). Therefore, it may be more appropriate to use a reward function based on discrete-time CBF [27], [28]. Therefore, we define the reward function based on discrete-time CBF as follows:

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

where α 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 III.

2) Control Lyapunov Function Reward

The reward function based on Control Lyapunov Functions (CLFs) guides the reverse diffusion process to generate a control sequence that ensures convergence to a given goal $\mathbf{x}_g$. Similar to CBF reward, we assume that the dynamics of the robot is (11). Control inputs $\mathbf{u}$ that make the following reward function W_{clf} positive will drive the state of the robot $\mathbf{x}$ to $\mathbf{x}_g$:

$$W_{\mathrm{clf}}(\mathbf{x}, \mathbf{u}) = -L_f h_{\mathrm{clf}}(\mathbf{x}) - L_g h_{\mathrm{clf}}(\mathbf{x})\mathbf{u} - \alpha(h_{\mathrm{clf}}(\mathbf{x})) \tag{13}$$

Therefore, guiding the reverse diffusion process to make this reward function W_{clf} positive ensures convergence to the goal. The gradient of this reward function with respect to the control input $\mathbf{u}$ is computed as follows:

$$\nabla_{\mathbf{u}} W_{\mathrm{clf}}(\mathbf{x}, \mathbf{u}) = -L_g h_{\mathrm{clf}}(\mathbf{x}) \tag{14}$$

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

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

Algorithm 1 Conditional Planning with Reverse Diffusion

Observe start $\mathbf{x}_s$ and goal $\mathbf{x}_g$
initialize controller $\mathbf{u}^N \sim \mathcal{N}(0,\ \mathbf{VI})$
initialize trajectory $\mathbf{x}^N$
$\mathbf{u}^{N-1} \sim \mathcal{N}(\boldsymbol{\mu}_\theta(\mathbf{u}^N,\ \mathbf{x}^N), \mathbf{V}\boldsymbol{\Sigma}^N)$
for $i = N-1, \ldots, 1$ **do**
  Compute $\mathbf{x}^i$ by $\mathbf{x}_s,\ \mathbf{u}^i$ and dynamics (1)
  $\mathbf{g} = \sum_{k=1}^{K}\sum_{t=0}^{T} \nabla_{\mathbf{u}_t^i} W_k(\mathbf{x}_t^i, \mathbf{u}_t^i)$
  Check $W_k(\mathbf{x}_t^i, \mathbf{u}_t^i)$ for positive and mask $\mathbf{g}$
  $\hat{\mathbf{u}}^i = \mathbf{u}^i + \mathbf{g}$
  Compute $\hat{\mathbf{x}}^i$ by $\mathbf{x}_s,\ \hat{\mathbf{u}}^i$ and dynamics (1)
  $\hat{\mathbf{x}}_0^i,\ \hat{\mathbf{x}}_T^i \leftarrow \mathbf{x}_s,\ \mathbf{x}_g$
  $\mathbf{u}^{i-1} \sim \mathcal{N}(\boldsymbol{\mu}_\theta(\hat{\mathbf{u}}^i, \hat{\mathbf{x}}^i),\ \mathbf{V}\boldsymbol{\Sigma}^i)$
end for

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 II-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 can 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:

$$\mathbf{u}^{N-1} \sim \mathcal{N}(\boldsymbol{\mu}_\theta(\mathbf{u}^N,\ \mathbf{x}^N),\ \mathbf{V}\boldsymbol{\Sigma}^N). \tag{15}$$

Then, the control inputs are fed into the system dynamics (1) to obtain the trajectory $\mathbf{x}^{N-1}$. At this point, it is important to emphasize that the control sequence and states are consistent. Then, the trajectory is evaluated by the reward functions defined in Section II-C. If each reward function is positive at a certain timestep, 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. Then, the guided controls $\hat{\mathbf{u}}^i$ are obtained by adding the gradient to the denoised controls $\mathbf{u}^i$ and the updated states $\hat{\mathbf{x}}^i$ are obtained using $\hat{\mathbf{u}}^i$. 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. 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.

III. Experiments and Discussion

A. Model Training and Environment Setup

We trained CoBL-Diffusion using the ETH pedestrian dataset [29] for 500 epochs on 276,874 8-second trajectories. The architecture and hyperparameters of the diffusion model are based on an open-source implementation[1]. The covariance weight (7) was set to increase linearly from 0.5 to 1. The simulation environment is from the UCY pedestrian dataset [30] for challenging multi-agent yet feasible settings. The environment used for the simulation has eight pedestrians, as shown in Fig. 3. We selected eight goal locations and ran 200 simulations, 25 for each goal.

We assumed the robot follows single integrator dynamics, $\dot{\mathbf{x}} = \mathbf{u}$, noting that our algorithm can easily be extended to other types of dynamics. We use the following CBF and CLF:

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

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, and **Velocity Obstacle (VO)** [31], which is a geometric approach for collision avoidance that uses the relative velocity between a robot and an obstacle to identify potential collision velocities. 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; **CoBL-Diffusion$^-$ (CoBL$^-$)** without covariance weight to investigate the effect of the covariance weight (7); **DCoL-Diffusion (DCoL)**, which uses CLF and a distance reward W_{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.

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. Since the simulations are discrete, the CBF reward cannot completely guarantee the safety. As a result, it is necessary to introduce a buffer between the barrier radius and collision radius, set to 1 m and 0.7 m, respectively.

Each method is compared against three planning 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 if the plan is followed. The smoothness is measured by the worst-case root squared difference in control inputs.

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

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

Planner	Coll. (↓)	Goal-Reach. (↓)	Smoothness (↓)
CoBL	**0.5 %**	$\mathbf{0.42 \pm 0.22}$	$\mathbf{0.07 \pm 0.03}$
CoB	17.5 %	3.82 ± 2.00	0.20 ± 0.05
dCoBL	46.5 %	0.37 ± 0.16	0.05 ± 0.01
DCoL	40.5 %	0.63 ± 0.74	0.07 ± 0.06
CoBL$^-$	8.5 %	0.48 ± 0.34	0.08 ± 0.04
CBF-QP	62.5 %	0.07 ± 0.02	0.50 ± 0.42
VO	0 %	0.31 ± 0.54	1.78 ± 0.32
Human	74.5 %	0	0.08 ± 0.04

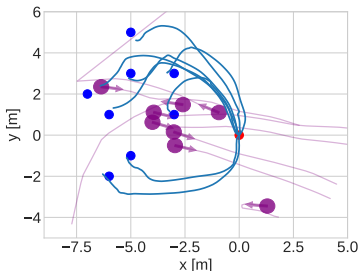

Fig. 3: Visualization of the planning environment in a realistic dynamic setting with pedestrians. The initial position of the robot is marked by a red dot, and the target goals are indicated by blue dots. The generated trajectories are represented by blue lines, while the purple circles and lines represent the humans and their respective trajectories.

C. Results

The results of each model 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 in the dynamic multi-agent environment. On the other hand, CBF-QP is unable to avoid moving obstacles. 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. Using dCoBL or DCoL rewards makes it challenging to ensure safety. Comparing CoBL with CoB, the contribution of CLF reward to goal-reaching is prominent in this 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 for the covariance (7) to promote convergence of the earlier control sequence slightly reduces the collision rate.

IV. Conclusion and Future Directions

Our proposed CoBL-Diffusion synthesizes robot controllers for safe planning in dynamic multi-agent environments. During the denoising process, the model guides the reverse diffusion process with reward functions based on CBF and CLF, iteratively improving the control sequence to satisfy the safety and stability constraints. In the experiment, we found that our model can smoothly reach goal locations within an acceptable distance while having very low collision rates. To promote real-time applications, we plan to explore how to reduce the inference time of our models, such as leveraging ideas from DDIM [32] and consistency models [33], 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] J. Sohl-Dickstein, E. Weiss, N. Maheswaranathan, and S. Ganguli, "Deep unsupervised learning using nonequilibrium thermodynamics," in *Int. Conf. on Machine Learning*, 2015.

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

[12] 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.

[13] 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.

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

[15] 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.

[16] 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.

[17] 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.

[18] 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.

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

[20] 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.

[21] 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.

[22] C. I. Mavrogiannis, P. Alves-Olivera, 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.

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

[24] 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.

[25] 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.

[26] 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.

[27] 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.

[28] 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.

[29] 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.

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

[31] 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.

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

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