2024 IEEE International Conference on Robotics and Automation (ICRA)
May 13-17, 2024. Yokohama, Japan

Efficient Motion Planning for Manipulators with Control Barrier Function-Induced Neural Controller

Mingxin Yu[1], Chenning Yu[2], M-Mahdi Naddaf-Sh[3], Devesh Upadhyay[3], Sicun Gao[2], and Chuchu Fan[1]

***Abstract*— Sampling-based motion planning methods for manipulators in crowded environments often suffer from expensive collision checking and high sampling complexity, which make them difficult to use in real time. To address this issue, we propose a new generalizable control barrier function (CBF)-based steering controller to reduce the number of samples needed in a sampling-based motion planner RRT. Our method combines the strength of CBF for real-time collision-avoidance control and RRT for long-horizon motion planning, by using CBF-induced neural controller (CBF-INC) to generate control signals that steer the system towards sampled configurations by RRT. CBF-INC is learned as Neural Networks and has two variants handling different inputs, respectively: state (signed distance) input and point-cloud input from LiDAR. In the latter case, we also study two different settings: fully and partially observed environmental information. Compared to manually crafted CBF which suffers from over-approximating robot geometry, CBF-INC can balance safety and goal-reaching better without being over-conservative. Given state-based input, our neural CBF-induced neural controller-enhanced RRT (CBF-INC-RRT) can increase the success rate by 14% while reducing the number of nodes explored by 30%, compared with vanilla RRT on hard test cases. Given LiDAR input where vanilla RRT is not directly applicable, we demonstrate that our CBF-INC-RRT can improve the success rate by 10%, compared with planning with other steering controllers. Our project page with supplementary material is at https://mit-realm.github.io/CBF-INC-RRT-website/.**

I. Introduction

Despite the wide adoption of robotic manipulators in many real-world applications, real-time planning of safe execution paths for manipulators in crowded environments can still be challenging due to high-dimensional complex dynamics. Sampling-based motion planning methods, such as Rapidly Exploring Random Trees (RRT) [1], Probabilistic Roadmaps (PRM) [2], and their extensions [3]–[7] have demonstrated their efficacy in generating collision-free paths in complex environments. However, the high sampling complexity of those methods is prominent for manipulators. Moreover, those methods require accurate state estimation before planning, which often assumes static environments and precise knowledge of the shapes and positions of obstacles.

The number of samples used in RRT can be reduced if expanding a node has a higher likelihood of success, which helps to reduce the node number for finding a solution.

This work was supported by the National Science Foundation (NSF) CAREER Award #CCF-2238030 and the MIT-Ford Alliance Program.

[1] Department of Aeronautics and Astronautics, Massachusetts Institute of Technology, USA {yumx35, chuchu}@mit.edu

[2] Department of Computer Science and Engineering, University of California, San Diego, USA {chy010, sicung}@eng.ucsd.edu

[3] The work was done when authors were at Ford Motor Company, USA {mahdinaddaf, deveshu}@gmail.com

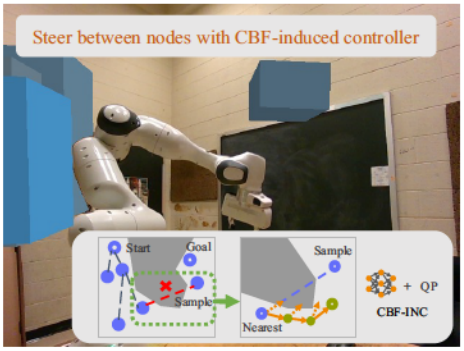

Fig. 1. An example of a Franka Emika Panda robot planning motions using our proposed framework. We train a CBF-induced neural controller and integrate it into the steer function in sampling-based motion planning. The controller is used for safe exploration and steers the edge to collision-free space without being over-conservative.

The expansion of a node will terminate when a collision is detected. On the contrary to recklessly heading towards the sampled state, using a safe steering function will substantially reduce the early termination of expansion. In this paper, we are inspired by the use of control barrier functions (CBF) [8], [9] in multiple robotics applications for safe control to design such a steering function. Prior works [10]–[13] have shown significant advancements in combining the safe CBF controller with sampling-based motion planning algorithms for low-dimensional systems. In the context of safe control, CBF has demonstrated success in rather complex robotic systems [14], [15], including manipulators [16], [17]. However, the CBFs used in the above works are typically manually designed as functions over the state of the environment, requiring both extensive experts' experience and accurate estimation from sensory data. When applying to robotic manipulator systems, the articulated nature of robots also poses an extensive computation burden for state estimation to handle the varying geometry of robots [16], [18]. For the simplicity of construction, geometric shapes of a robot are often over-approximated [18]–[21] and CBFs are often selected in simple forms like signed distance or [16], [18] or in quadratic form [22], [23]. The cost for simplicity is that the CBFs may be over-conservative and a feasible control signal is not guaranteed to be found. To address these challenges, we are inspired by the recent advances in learning CBFs for the efficient synthesis of safe controllers, which has shown success in walking [24], flight [25] and multi-agent setting [26], with some other attempts to extend these neural CBFs to observation-feedback systems [27], [28].

In this work, we build upon the motion planning framework RRT [1] and learn a CBF-INC to steer the system

towards newly sampled configuration. CBF-INC has two variants handling different inputs: state (signed distance) and point cloud input from LiDAR. Given state input, our framework CBF-INC-RRT increases the success rate by and reduces the number of explored nodes by on the most challenging test cases, compared with vanilla RRT and other neural-controller-enhanced RRT. CBF-INC-RRT also doubles the success rate and halves the explored nodes, compared with the hand-crafted CBF-enhanced RRT method by avoiding over-conservativeness. With point cloud input setting, where many methods (like vanilla RRT and hand-crafted CBF) are not directly applicable, CBF-INC-RRT still improve the success rate by on challenging cases, compared with planning with other steering controllers.

Contributions. Our contributions are summarized below: (i) We present a CBF-INC specialized for robotic manipulators. Our neural networks tackle high-dimensional observations and complex geometric link shapes. To the best of our knowledge, this is the first CBF-style controller taking raw sensor input in high-dimensional manipulators. (ii) We present a framework - CBF-INC-RRT that incorporates the learned neural CBF into the motion planning algorithm. Such a framework preserves the completeness of traditional motion planning methods while benefiting from the safe exploration of CBF-INC. (iii) Through extensive experiments on 4D and 7D manipulators, we demonstrate that planning algorithms using CBF-INC significantly outperform baselines, in terms of success rate and exploration efficiency. We also show CBF-INC generalizes to dynamic environments and evaluate CBF-INC-RRT on hardware.

II. Related Works

Local Safety for robotic arms. Safe deployment in the real world is crucial for general-purpose robotic arms. Various non-learning techniques have been proposed to tackle the collision avoidance problems on manipulators, ranging from potential field methods [29]–[31], reachability analysis [32], to CBF [17], [18], a trusted tool for ensuring safety in control systems [8], [33]. These methods, including CBF are usually hand-crafted via signed distance [16], [18], [34], quadratic form [22], [23] or minimum uniform scaling factor [17]. While effective for certain environments, they require substantial design efforts and perfect knowledge of the environment. For simplicity, CBFs designed for robotic arms with multiple degrees of freedom (DoF) often use a set of simple convex shapes to over-approximate the rigid body links [18]–[21], leading to over-conservative policies. Extending these approximations from one shape to another is also not straightforward.

In contrast, data-driven methods, exploiting the power of neural networks, have shown promise in addressing complex geometric shapes of manipulators [35], [36], even when only high-dimensional observations are available [37]. Recently, learning-based CBFs have demonstrated potential in resolving these challenges on drones in multi-agent setting [26] and visual navigation systems [27]. However, in robotic manipulator systems, the adoption of neural networks in CBFs is restricted to parameter search guidance [38] due to the convoluted collision-free configuration space.

Inductive bias in sampling-based motion planning. One stream of methods [39]–[42] has been proposed to improve sampling-based planning methods by incorporating inductive bias. The methods fall into two categories. One set of methods seeks to find heuristic functions to prioritize the samples to explore, including Fast Marching Trees [43], sampling-based A* [44], and recent GNN-related works [42], [45]. Some works also consider improving the sampling strategy [39], [46]. However, these methods still suffer from finding a control policy for the planned reference trajectory, especially for complex dynamics. The other class of methods conceives motion planning problems as sequential decision-making problems and relies on neural policies [41], [42], [47] such as imitation learning [48] or reinforcement learning [40] in an end-to-end manner, aiming to find a collision-free trajectory directly with a neural network. Some further consider adding explicit safety constraints during training to accelerate [35], [49]. Though these methods have shown impressive results, they sacrifice the completeness guarantee of many motion planning algorithms.

The most related works to this paper are [10]–[13], which explore incorporating CBF into sampling-based motion planning. [11]–[13] proposed improvements on sampling methods and [12], [13] focused on optimal planning. Our method only modifies the steer function in motion planning, and is therefore able to work directly with many sampling-based planners. While [11]–[13] work in simple 2D environments with a hand-crafted CBF and [10] work on 2D car dynamics, we propose a CBF-induced neural controller component and conduct experiments with high-DoF robotic manipulators in a highly cluttered environment. Another closely related work is [36], which synthesizes a control policy for high-DoF manipulators via a neural signed distance function and quadratic programming (QP). The training of CBF-induced neural function explicitly penalizes the infeasibility of QP, improving the goal-reaching performance of controllers.

III. Preliminaries

We consider a robot with control-affine dynamics , where $\mathcal{C}$ is the robot configuration and $\mathcal{U}$ is the control input, in a cluttered environment $\mathcal{E}$. We assume both configuration space $\mathcal{C}$ and action space $\mathcal{U}$ to be bounded. The robot perceives the environment $\mathcal{E}$ through an observation model $\mathcal{E}$ $\mathcal{O}$. In this work, we consider two different observation models, namely, a signed-distance observation model and a LiDAR-based observation model, whose details are elaborated in IV-B. The state-observation space $\mathcal{X}$ $\mathcal{C}$ $\mathcal{O}$ can be partitioned into three subspaces: an unsafe set $\mathcal{X}$ where the robot collides with or penetrates itself or environmental obstacles, a safe set $\mathcal{X}$ $_{\text{thres}}$ $\mathcal{X}$ where the robot is at least $_{\text{thres}}$ away from the unsafe set, and a boundary set $\mathcal{X}$ $\mathcal{X}$ $\mathcal{X}$ $\mathcal{X}$.

Given a start configuration and goal configuration , satisfying $\mathcal{E}$ $\mathcal{E}$ $\mathcal{X}$, we seek to

find a feasible control sequence $\mathcal{U}$ that steers the system from to $\mathcal{X}_{\text{goal}}$, while ensuring $\mathcal{E}$ $\mathcal{X}$. The goal region $\mathcal{C}_{\text{goal}}$ is defined as $\mathcal{E}$ $_{\text{goal}}$ for some pre-defined radius $_{\text{goal}}$. In this work, we build upon motion planning algorithm RRT [1] and substitute the steer function with a neural-network-based controller for control input.

Rapid-exploring random trees (RRT). RRT [1] tackles the motion planning problem by starting with sampling a set of configurations in the free space $\mathcal{X}$ $\mathcal{X}$ $\mathcal{X}$. The algorithm then attempts to build or expand a tree to connect these nodes with the start configuration and the goal configuration . This two-step process is repeated until a collision-free path connecting the two configurations is found or until termination. Each edge on the tree has to be collision-free, thus requiring collision checking at the edge construction stage.

Control barrier function. CBF ensures safety in control systems by enforcing the states of the systems to stay in the safe set. Extended CBFs in state-observation space $\mathcal{X}$ are scalar functions such that for some :

$$\begin{array}{l} \mathcal{X} \\ \mathcal{X} \\ \mathcal{X} \\ \qquad \mathcal{U} \end{array} \tag{1}$$

where and denote the Lie derivatives, which capture the rate of change of along the system trajectories induced by and , respectively. Since is also a function of , the calculation of Lie derivatives requires the calculation of — and — . And are small margins to encourage the strict inequality satisfaction of CBF conditions. It is proved in [8] that if the initial state $\mathcal{X}$ and a Lipschitz continuous policy $\mathcal{X}$ $\mathcal{U}$ selects actions from the set $\mathcal{K}_{\text{CBF}}$, then the trajectory does not leave the safe set $\mathcal{X}$.

Here we can see a connection between motion planning and control barrier function. As long as the controller ensures the forward-invariant of the safety set, then the agent never encounters collisions. Using such a CBF controller ensures the collision-free constraint for motion planning.

IV. Methods

Motivated by the connection between the motion planner's collision-free constraint and CBF's safety guarantees, we present a two-stage approach using a CBF-induced neural controller that allows a robot to avoid obstacles and a motion planner that guides the robot to jump out of stuck regions and toward its goal. We first train a neural network CBF-induced neural function (CBF-INF) for the robot. CBF-INF is trained to satisfy all the constraints in (1). Next, we synthesize a controller CBF-INC using the trained CBF-INF and incorporate it into the motion planning framework.

A. Learning Framework for CBF-INF

Training procedures. We utilize an offline strategy to train CBF-INF , where the training data is pre-collected. Our training dataset is a combination of two different parts, both are collected in various training environments : (i) We gather rollout trajectories using classical controllers (e.g. LQR controller). These trajectories are generated with random initial and goal states. (ii) To ensure coverage of less-explored spaces within the rollout, we uniformly sample the robot's pose in the configuration space. The presence of observation allows that the training environments not necessarily be the same as test ones and that CBF-INF can easily generalize to new environments. CBF-INF is trained to minimize an empirical loss function $\mathcal{L}$ like [27]:

$$\begin{array}{l} \mathcal{L} \quad \overline{\qquad}_{\text{safe}} \quad \mathcal{X} \\ \qquad \overline{\qquad}_{\text{unsafe}} \quad \mathcal{X} \qquad\qquad — \end{array} \tag{2}$$

where $\mathcal{X}$ $\mathcal{U}$ are positive tuning parameters, is , $_{\text{safe}}$, $_{\text{unsafe}}$ and are the number of points in the training samples in $\mathcal{X}$, $\mathcal{X}$ and $\mathcal{X}$, respectively. As the term $\mathcal{U}$ is linearly dependent on , the minimum value can be easily found within a bounded action space $\mathcal{U}$ via linear programming (LP). The existence of a feasible control signal in the last condition in (1) can be demonstrated when the third loss term comes to . Note that CBF-INF is trained as a Neural Network from finite samples and therefore not a valid CBF before being verified to satisfy the CBF constraints over the entire space. The latter is a theoretically hard problem [50]. However, the learned CBF-INF can be used as a steering function and provides significant empirical improvements over vanilla RRT (shown in Sec. V). We will also show in Fig 5 that the empirical satisfaction rates of the CBF constraints over finite samples are close to .

Computing Lie-derivatives. Directly computing the third condition in Eq. (1) requires the calculation of — and — . We first assume the obstacle velocities are much smaller than the links of manipulators in the dynamic scenarios, so we can disregard the change of in between two computational steps induced by the change of environment $\mathcal{E}$ when computing Lie-derivatives. Furthermore, we replace the exact calculation of — with numerical differentiation, i.e., — ————————, where is a one-hot vector with . This approach bypasses the explicit expression of forward kinematics of manipulators with many degrees of freedom and only demands a "black-box" access to the numerical values of the kinematics and the Jacobian [22].

B. Specializing Functions for Manipulators: State-based and LiDAR-based CBF-INF

In contrast to hand-crafted CBFs utilized on multi-DoF robotic manipulators in [11], [16], [18], we aim to learn a neural function that encodes the safety constraint of avoiding both self-collision and collision with the exterior environment. Based on different observation models, we propose two types of CBF-INF, both sharing the same training procedure.

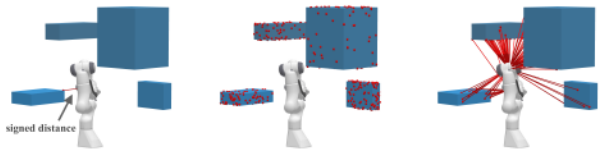

Fig. 2. Illustration of observations. Left: sCBF-INF takes the signed distance to the nearest obstacle as observation. Middle: For *fully-observable* environments, oCBF-INF uses a point cloud sampled uniformly on the obstacles as observation. Right: For *partially-observable* environments, oCBF-INF observes the point cloud from a mounted LiDAR sensor.

State-based CBF-INF (sCBF-INF). The observation o in sCBF-INF is the minimum signed distance, d, between the obstacles and any robot links. In this setting, it is easy for the CBF-INF to determine, from the sign of d, whether the robot is in a collision-free state with respect to the environment. The only remaining challenge is to learn the self-collision pattern, which solely depends on the configuration q. To address this, we design the neural network to take the concatenated vector q and d as input, and generate a scalar as output, as in Fig. 3.

LiDAR-based CBF-INF (oCBF-INF). sCBF-INF relies heavily on accurate distance information from the environment, which is not available or costly to acquire in a dynamic or partially observable environment. A more flexible implementation is to define CBF-INF as functions of partial observations, e.g., LiDAR. In the LiDAR-based setting, the observation o is composed of N raw points sampled on the surface of obstacles paired with their respective normal vectors, similar to [41]. This observation is represented by a finite set $o = \{(p_i, n_i)\}_{i\in[1,\cdots,N]} \in \mathbb{R}^{N\times 6}$, with the 3 dimensional points p_i and 3 dimensional normal vector n_i in the world frame. The point cloud can be retrieved using the LiDAR sensor or a depth camera mounted on the robot.

For each link of the manipulator, we transform the point cloud into its local frame, concatenate each point with a one-hot vector of the link index, and then feed all the transformed point clouds into a PointNet [51], which encodes the point clouds while ensuring permutation invariance on the order of points. The feature vectors from the PointNet are further fed into an MLP, concatenated with the configuration q. The whole network architecture is shown in Fig. 3.

C. Integrating CBF-INF with Motion Planning

Sampling-based motion planning algorithms are widely adopted to efficiently search high-dimensional spaces via building a space-filling tree. Despite their guaranteed completeness, these algorithms explore the configuration space via random shooting, and fine-grained collision-checking is required for each edge. The edge will be discarded if any part of the checking fails, even though a slight detour may save the edge, which wastes the computation in such a failed exploration. Our framework, however, encourages the planner to explore the configuration space more wisely by using safe controllers, i.e., CBF-INC. By incorporating the controller into the motion planning algorithm, the likelihood of successfully expanding a node is increased, which reduces

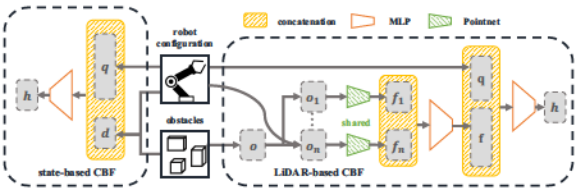

Fig. 3. The overall neural network architecture. Left: The architecture of the sCBF-INF. Right: The architecture of the oCBF-INF.

the exploration cost given the controller's reactivity to obstacles.

Synthesize controller. We construct our controller CBF-INC from CBF theory [8] that modifies any given reference controller u_{nominal} by solving the quadratic programming (QP) problem:

$$\begin{aligned} u(q, o, q_g) = \arg\min_{u\in\mathcal{U}} \|u - u_{\text{nominal}}(q, o, q_g)\|^2, \\ \text{s.t. } L_f h(q, o) + L_g h(q, o) \cdot u + \alpha h \leq 0. \end{aligned} \tag{3}$$

Here, the controller seeks to minimize the squared difference with the nominal control input within the action space $\mathcal{U}$, while satisfying the safety constraint.

Safe-steering RRT. In this study, we focus on Rapidly-exploring Random Trees (RRT) [1] as a representative sampling-based motion planner. In RRT, the steer function generates a prospective edge, which extends the current search tree toward the direction of a randomly selected point. The steer function needs to conduct collision checking in the process. Different from [11], which substitutes explicit checking for the nearest neighbor and changes the original framework, we propose CBF-INC-RRT, to use CBF-INC as the steer function. This is a general approach that can be applied to any sampling-based motion planners using a steer function. Within our steer function, the robot rolls out a trajectory using CBF-INC. The q_g in (3) is set to be the newly sampled point. The generated trajectory and control sequence then serve as the edge added to the search tree. Details of the complete algorithm are provided in the Appendix.

There may be concerns about the completeness of this modified planning paradigm. However, we can still ensure completeness by opting to use CBF-INF to discard unsafe LQR actions instead of modifying them after a certain number of exploration steps, as suggested in [13]. This optional variation allows us to maintain the crucial aspect of completeness while improving safety and efficiency.

V. Experimental Results

Experiment setup. We evaluate our methods on a 4-DoF Dobot Magician in simulation and a 7-DoF Franka Panda both in simulation and the real world. Similar to [16], we consider direct control over the joint velocities, i.e., $\dot{q} = u$. In experiments, obstacles within the environment are depicted as cuboids. CBF-INF is trained in environments with 4 fixed-size obstacles with random poses. We test the method on more challenging environments with 8 obstacles of random sizes and poses unless specified otherwise. The nominal policy u_{nominal} for the QP controller is selected as LQR. The

simulations of continuous-time robot dynamics and control frequency occur at Hz and Hz, respectively.

Baselines. Apart from vanilla RRT (RRT) [1], we also compare against RRT variants with the following steer controllers in both state-based and LiDAR-based settings:

Reinforcement Learning (sRL&oRL) [52]: We design the reward function to encourage goal-reaching and penalize collision, then train the controller with DDPG.

Imitation Learning (sIL&oIL) [53]: Use behavior cloning to mimic the planned trajectories generated from an expert motion planner BIT* [54].

Some baseline methods require complete information of the environment, thus only available in the state-based setting:

Hand-crafted CBF (hCBF) [17]: The construction of this CBF adopts the minimum uniform scaling factor. Only available for 7-DoF Panda robot.

Safe RL method OptLayer (sOpt) [35]: Add additional optimization layer to force the controller satisfy 1, where is a signed distance function instead of CBF.

The baseline methods with steer controllers are abbreviated with a suffix '-steer'. We also conduct ablation studies evaluating the controllers only. All neural controllers are designed and trained using the same environment observations and neural network architectures. The methods are evaluated on randomly generated 1000 easy and 1000 hard testing cases, based on the time required for BIT* [54] to find a solution. All the experiments are conducted with a predefined node limit: for the 4-DoF robot, exploration is restricted to a maximum of 200 nodes, while the limit for the 7-DoF robot is 500 nodes.

A. Motion Planning in Simulation

Evaluation metrics. For motion planning problems in the section, we consider the following metrics: (1) Success rate (SR): A problem is successfully solved only if a collision-free path is found within the node limit. (2) Explored nodes: the attempts of adding a node to the search tree, regardless of success or not. (3) Total time consumption: One common concern about learning-based methods is their running speed due to the frequent calling of a large neural network model at inference time. We separate this metric into two categories: (3.1) online time, which is directly related to control frequency and observation update frequency and must be performed online during execution. This includes neural network inference time, QP solving, and perceiving observations; and (3.2) planning time, proportional to the total timesteps when expanding the search tree, is the remaining time consumption other than online time. This includes planning, running the simulations, and performing collision checking. Because online time highly depends on selected parameters, we only report planning time in the main text. Results for online time can be found in the supplementary.

Performance of state-based methods. Shown in Table I, we see significant improvement in success rate and exploration efficiency (explored nodes) using sCBF-INC-RRT in the state-based setting. Remarkably, performance improvement

TABLE I

EXPERIMENTS UNDER STATE-BASED SETTING. PERFORMANCE ON AVERAGE SUCCESS RATE (SR), NUMBER OF EXPLORED NODES ON 1000 TEST CASES, AND SUMMED PLANNING TIME ON 100 TESTING CASES, AVERAGED OVER 3 RANDOM SEEDS.

(a) Results on 4-DoF Magician robot.

	Easy			Hard		
method	SR↑ (%)	nodes↓	time(s)	SR↑ (%)	nodes↓	time(s)
RRT	91.2	37.4	**42.3**	68.1	81.2	**43.3**
sCBF-INC-RRT	**97.7**	**24.1**	45.8	**84.6**	**54.6**	45.3
sIL-steer	89.4	43.0	55.2	58.2	99.6	80.1
sRL-steer	90.3	40.4	51.5	60.8	94.0	73.4
sOpt-steer	84.6	51.2	65.7	64.5	93.2	70.6

(b) Results on 7-DoF Franka Panda robot.

	Easy			Hard		
method	SR↑ (%)	nodes↓	time(s)	SR↑ (%)	nodes↓	time(s)
RRT	85.7	112.3	166.0	62.8	252.5	**278.6**
sCBF-INC-RRT	**92.0**	**67.5**	**160.5**	**76.1**	162.5	345.8
hCBF-steer	39.2	134.1	472.6	26.3	**160.5**	525.3
sIL-steer	39.1	324.9	459.0	25.3	400.5	547.2
sRL-steer	83.9	124.9	235.9	60.1	266.7	395.3
sOpt-steer	26.5	155.3	902.8	16.4	174.9	942.3

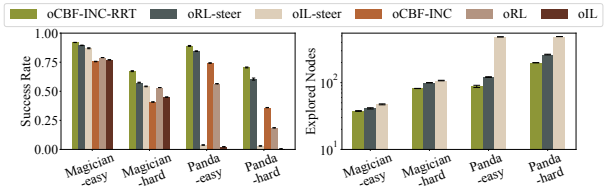

Fig. 4. Motion planning experiments under LiDAR-based setting.

is much more pronounced on challenging hard testing problems. Regarding planning time, sCBF-INC-RRT performs comparably with vanilla RRT and takes considerably less time compared to sRL-steer, sIL-steer methods, and even safe method sOptLayer-steer. It's worth discussing why hCBF-steer performs much worse than CBF-INC-RRT and even RRT. First, hCBF is more conservative because it over-approximates the geometry shapes of the robot. This limits its performance, especially in cluttered environments. Second, the QP controller in hCBF-steer sometimes cannot find a feasible solution. Although we've attempted to relax the optimization problem with a constraint violation penalty term, this compromises the safety guarantee of hCBF-steer.

Performance of LiDAR-based methods. We evaluate algorithms that integrate various controllers into steer functions (e.g., oCBF-INC-RRT) and those that solely leverage controllers to address the planning problems (e.g., oCBF-INC). The experiments are conducted in fully-observable environments. In Fig. 4, we show that motion planning methods significantly outperform end-to-end controllers regarding success rate, demonstrating that motion planning can help improve the feasibility of finding a solution under QP formulations. Among all the motion planning methods, oCBF-INC-RRT outperforms oRL-steer and oIL-steer on success rate and number of explored nodes, especially in hard tests. Regarding the total time consumption, our method

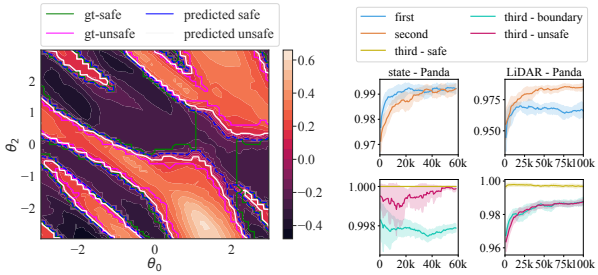

Fig. 5. Left: Slices of oCBF-INF value for Panda, obtained by sweeping across two joints, with all other joint states and obstacle positions held constant. Right: Learning curves of constraint satisfaction rate on Panda.

does take a slightly longer time than vanilla RRT due to frequent inference calls of neural network. However, our method takes about s and s on average to compute the control signals and step the simulation for a 1-second period on Dobot Magician and Franka Panda, respectively. This indicates our planning can be performed faster than real-time, further establishing applicability to the real world.

B. Ablation Study in Simulation

We first visualize CBF contour in configuration space in Fig 5. We also plot the learning curves of satisfaction rates of each CBF constraint on our neural CBF-induced neural controller-enhanced RRT. All the constraints are satisfied over and on the validation sets for state-based and LiDAR-based settings after training, respectively.

We then evaluate our controller oCBF-INC, by unrolling its control output without integrating it into the motion planning framework, over 1000 planning problems with obstacles. This experiment showcases how different neural controllers balance goal-reaching and safety. We conduct experiments in two distinct environments: **(i)** the environment is static and fully observable, where the observation contains points uniformly sampled on the surfaces of obstacles; **(ii)** obstacles are dynamic and move at a constant speed and the environment is only partially observable, where the observation is acquired by two 3D LiDARs mounted on the manipulators.

Evaluation metrics. We evaluate the end-to-end controllers based on three measures averaged over the testing problems: (1) goal-reaching rate: A goal configuration is identified as reached, only if the agent does not encounter any collision during the rollout, and eventually reaches the goal within a limited time horizon. (2) safety rate: the ratio of collision-free states along the entire trajectory. (3) makespan: rollout steps for succeeded cases.

Performance. Fig 6 shows the performance of both Dobot Magician and Franka Panda robots in the considered environments. In the static and fully observable environment, oCBF-INC outperforms baselines by approximately on the goal-reaching rate and safety rate for the Magician robot and by more than for Franka Panda. Although all algorithms face a substantial performance drop in the dynamic and partial-observable environment, our controller still notably exceeds oRL and oIL baselines. The makespan performances are generally comparable across algorithms, while oCBF-INC is slightly better. This demonstrates the method achieves a great balance between efficiency and safety.

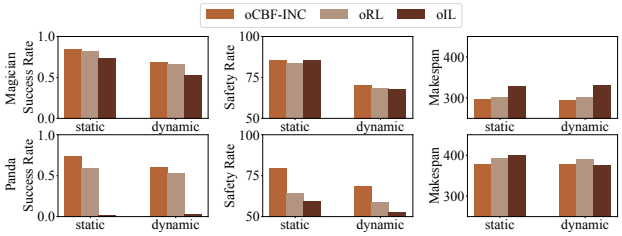

Fig. 6. Safety and goal-reaching performance of LiDAR-based controllers in an end-to-end manner (without motion planning module).

C. Hardware Demonstration

Finally, we validate our proposed method on a real Franka Emika Panda controlled at Hz, the same as in the simulation. We randomly select several planning problems in V-A. In order to avoid the floating blocks, we construct the obstacles and vision modules in the simulation, then synthesize real-world video with simulated obstacles, similar to [17]. The components are communicated via ROS.

As shown in Fig 7 and supplementary video, our method solves the planning problems successfully. On the right of Fig 7, we also show the signed distance of the robot to the environment. The experiments confirm that the planned trajectory is safe and robust to the noise in execution.

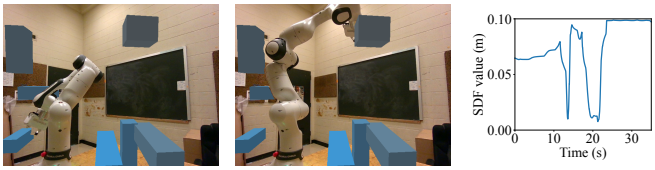

Fig. 7. Left & middle: snapshots of solving a motion planning problem with our method. Right: the curve of minimum signed distance to all obstacles along a trajectory. Videos are included in the supplementary.

VI. Discussions and Conclusion

This paper explores a direction for robotic safety control by integrating CBF-induced neural controller - CBF-INC into motion planning. Instead of looking for a certified CBF, we train CBF-INF for robotic manipulators under different observation settings and incorporate the synthesized controller into sampling-based motion planning algorithms. We evaluate the proposed methods in various environments, including 4-DoF and 7-DoF arms, and in the real world. We demonstrate that CBF-INC generalizes well to unseen scenarios and the overall framework outperforms other methods in terms of goal-reaching rate and exploration efficiency.

However, there are several limitations of our paper: (1) Since oCBF-INF takes the raw sensor data as input, the performance is directly dependent on the sensor data quality. Quantifying the input cloud's uncertainty precisely remains an open-ended problem. (2) oCBF-INF requires transforming the input point cloud into each link frame, which poses potential scalability issues for robots with higher DoF. (3) Our computation of the Lie derivative assumes the moving speeds of obstacles are small in dynamic scenarios. We hope to relax this assumption in future work.

REFERENCES

[1] S. M. LaValle and J. J. Kuffner Jr, "Randomized kinodynamic planning," *The international journal of robotics research*, vol. 20, no. 5, pp. 378–400, 2001.

[2] L. Kavraki, P. Svestka, J.-C. Latombe, and M. Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration spaces," *IEEE Transactions on Robotics and Automation*, vol. 12, no. 4, pp. 566–580, 1996.

[3] D. J. Webb and J. Van Den Berg, "Kinodynamic rrt*: Asymptotically optimal motion planning for robots with linear dynamics," in *2013 IEEE international conference on robotics and automation*. IEEE, 2013, pp. 5054–5061.

[4] R. Kala, "Rapidly exploring random graphs: motion planning of multiple mobile robots," *Advanced Robotics*, vol. 27, no. 14, pp. 1113–1122, 2013.

[5] M. Brunner, B. Brüggemann, and D. Schulz, "Hierarchical rough terrain motion planning using an optimal sampling-based method," in *2013 IEEE International Conference on Robotics and Automation*. IEEE, 2013, pp. 5539–5544.

[6] J. Wang, W. Chi, C. Li, C. Wang, and M. Q.-H. Meng, "Neural rrt*: Learning-based optimal path planning," *IEEE Transactions on Automation Science and Engineering*, vol. 17, no. 4, pp. 1748–1758, 2020.

[7] B. Ichter, E. Schmerling, T.-W. E. Lee, and A. Faust, "Learned critical probabilistic roadmaps for robotic motion planning," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2020, pp. 9535–9541.

[8] A. D. Ames, J. W. Grizzle, and P. Tabuada, "Control barrier function based quadratic programs with application to adaptive cruise control," in *53rd IEEE Conference on Decision and Control*, 2014, pp. 6271–6278.

[9] A. D. Ames, X. Xu, J. W. Grizzle, and P. Tabuada, "Control barrier function based quadratic programs for safety critical systems," *IEEE Transactions on Automatic Control*, vol. 62, no. 8, pp. 3861–3876, 2016.

[10] A. Manjunath and Q. Nguyen, "Safe and robust motion planning for dynamic robotics via control barrier functions," in *2021 60th IEEE Conference on Decision and Control (CDC)*. IEEE, 2021, pp. 2122–2128.

[11] G. Yang, B. Vang, Z. Serlin, C. Belta, and R. Tron, "Sampling-based motion planning via control barrier functions," in *Proceedings of the 2019 3rd International Conference on Automation, Control and Robots*, 2019, pp. 22–29.

[12] A. Ahmad, C. Belta, and R. Tron, "Adaptive sampling-based motion planning with control barrier functions," in *2022 IEEE 61st Conference on Decision and Control (CDC)*. IEEE, 2022, pp. 4513–4518.

[13] G. Yang, M. Cai, A. Ahmad, C. Belta, and R. Tron, "Efficient lqr-cbf-rrt*: Safe and optimal motion planning," *arXiv preprint arXiv:2304.00790*, 2023.

[14] G. Wu and K. Sreenath, "Safety-critical control of a planar quadrotor," in *2016 American control conference (ACC)*. IEEE, 2016, pp. 2252–2258.

[15] Q. Nguyen, A. Hereid, J. W. Grizzle, A. D. Ames, and K. Sreenath, "3d dynamic walking on stepping stones with control barrier functions," in *2016 IEEE 55th Conference on Decision and Control (CDC)*. IEEE, 2016, pp. 827–834.

[16] A. Singletary, W. Guffey, T. G. Molnar, R. Sinnet, and A. D. Ames, "Safety-critical manipulation for collision-free food preparation," *IEEE Robotics and Automation Letters*, vol. 7, no. 4, pp. 10 954–10 961, 2022.

[17] B. Dai, R. Khorrambakht, P. Krishnamurthy, V. Gonçalves, A. Tzes, and F. Khorrami, "Safe navigation and obstacle avoidance using differentiable optimization based control barrier functions," *arXiv preprint arXiv:2304.08586*, 2023.

[18] C. T. Landi, F. Ferraguti, S. Costi, M. Bonfè, and C. Secchi, "Safety barrier functions for human-robot interaction with industrial manipulators," in *2019 18th European Control Conference (ECC)*, 2019, pp. 2565–2570.

[19] C. Chang, M. J. Chung, and B. H. Lee, "Collision avoidance of two general robot manipulators by minimum delay time," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 24, no. 3, pp. 517–522, 1994.

[20] E. Rimon and S. P. Boyd, "Obstacle collision detection using best ellipsoid fit," *Journal of Intelligent and Robotic Systems*, vol. 18, pp. 105–126, 1997.

[21] H.-C. Lin, Y. Fan, T. Tang, and M. Tomizuka, "Human guidance programming on a 6-dof robot with collision avoidance," in *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2016, pp. 2676–2681.

[22] A. Singletary, P. Nilsson, T. Gurriet, and A. D. Ames, "Online active safety for robotic manipulators," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2019, pp. 173–178.

[23] A. Singletary, S. Kolathaya, and A. D. Ames, "Safety-critical kinematic control of robotic systems," *IEEE Control Systems Letters*, vol. 6, pp. 139–144, 2022.

[24] F. Castaneda, J. J. Choi, B. Zhang, C. J. Tomlin, and K. Sreenath, "Gaussian process-based min-norm stabilizing controller for control-affine systems with uncertain input effects and dynamics," in *2021 American Control Conference (ACC)*. IEEE, 2021, pp. 3683–3690.

[25] D. Sun, S. Jha, and C. Fan, "Learning certified control using contraction metric," in *Conference on Robot Learning*. PMLR, 2021, pp. 1519–1539.

[26] Z. Qin, K. Zhang, Y. Chen, J. Chen, and C. Fan, "Learning safe multi-agent control with decentralized neural barrier certificates," *arXiv preprint arXiv:2101.05436*, 2021.

[27] C. Dawson, B. Lowenkamp, D. Goff, and C. Fan, "Learning safe, generalizable perception-based hybrid control with certificates," *IEEE Robotics and Automation Letters*, vol. 7, no. 2, pp. 1904–1911, 2022.

[28] S. Dean, N. Matni, B. Recht, and V. Ye, "Robust guarantees for perception-based control," in *Learning for Dynamics and Control*. PMLR, 2020, pp. 350–360.

[29] O. Khatib, "Real-time obstacle avoidance for manipulators and mobile robots," in *Proceedings. 1985 IEEE International Conference on Robotics and Automation*, vol. 2, 1985, pp. 500–505.

[30] A. De Santis, A. Albu-Schaffer, C. Ott, B. Siciliano, and G. Hirzinger, "The skeleton algorithm for self-collision avoidance of a humanoid manipulator," in *2007 IEEE/ASME international conference on advanced intelligent mechatronics*, 2007, pp. 1–6.

[31] F. Flacco, T. Kröger, A. De Luca, and O. Khatib, "A depth space approach to human-robot collision avoidance," in *2012 IEEE International Conference on Robotics and Automation*, 2012, pp. 338–345.

[32] P. Holmes, S. Kousik, B. Zhang, D. Raz, C. Barbalata, M. Johnson-Roberson, and R. Vasudevan, "Reachable sets for safe, real-time manipulator trajectory design," *arXiv preprint arXiv:2002.01591*, 2020.

[33] P. Wieland and F. Allgöwer, "Constructive safety using control barrier functions," *IFAC Proceedings Volumes*, vol. 40, no. 12, pp. 462–467, 2007.

[34] J. Haviland and P. Corke, "Neo: A novel expeditious optimisation algorithm for reactive motion control of manipulators," *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 1043–1050, 2021.

[35] T.-H. Pham, G. De Magistris, and R. Tachibana, "Optlayer-practical constrained optimization for deep reinforcement learning in the real world," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2018, pp. 6236–6243.

[36] M. Koptev, N. Figueroa, and A. Billard, "Neural joint space implicit signed distance functions for reactive robot manipulator control," *IEEE Robotics and Automation Letters*, vol. 8, no. 2, pp. 480–487, 2023.

[37] P. Liu, K. Zhang, D. Tateo, S. Jauhri, Z. Hu, J. Peters, and G. Chalvatzaki, "Safe reinforcement learning of dynamic high-dimensional robotic tasks: navigation, manipulation, interaction," *arXiv preprint arXiv:2209.13308*, 2022.

[38] S. McIlvanna, N. N. Minh, Y. Sun, M. Van, and W. Naeem, "Reinforcement learning-enhanced control barrier functions for robot manipulators," *arXiv preprint arXiv:2211.11391*, 2022.

[39] B. Ichter, J. Harrison, and M. Pavone, "Learning sampling distributions for robot motion planning," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2018, pp. 7087–7094.

[40] T. Jurgenson and A. Tamar, "Harnessing reinforcement learning for neural motion planning," *arXiv preprint arXiv:1906.00214*, 2019.

[41] R. Strudel, R. G. Pinel, J. Carpentier, J.-P. Laumond, I. Laptev, and C. Schmid, "Learning obstacle representations for neural motion planning," in *Conference on Robot Learning*. PMLR, 2021, pp. 355–364.

[42] R. Zhang, C. Yu, J. Chen, C. Fan, and S. Gao, "Learning-based motion planning in dynamic environments using gnns and temporal encoding," in *Advances in Neural Information Processing Systems*, 2022.

[43] L. Janson, E. Schmerling, A. Clark, and M. Pavone, "Fast marching tree: A fast marching sampling-based method for optimal motion

planning in many dimensions," *The International journal of robotics research*, vol. 34, no. 7, pp. 883–921, 2015.
[44] S. M. Persson and I. Sharf, "Sampling-based a* algorithm for robot path-planning," *The International Journal of Robotics Research*, vol. 33, no. 13, pp. 1683–1708, 2014.
[45] C. Yu and S. Gao, "Reducing collision checking for sampling-based motion planning using graph neural networks," *Advances in Neural Information Processing Systems*, vol. 34, pp. 4274–4289, 2021.
[46] C. Zhang, J. Huh, and D. D. Lee, "Learning implicit sampling distributions for motion planning," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2018, pp. 3654–3661.
[47] M. Pfeiffer, M. Schaeuble, J. Nieto, R. Siegwart, and C. Cadena, "From perception to decision: A data-driven approach to end-to-end motion planning for autonomous ground robots," in *2017 ieee international conference on robotics and automation (icra)*. IEEE, 2017, pp. 1527–1533.
[48] A. H. Qureshi, A. Simeonov, M. J. Bency, and M. C. Yip, "Motion planning networks," in *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, 2019, pp. 2118–2124.
[49] X. Wang, "Ensuring safety of learning-based motion planners using control barrier functions," *IEEE Robotics and Automation Letters*, vol. 7, no. 2, pp. 4773–4780, 2022.
[50] X. Huang, D. Kroening, W. Ruan, J. Sharp, Y. Sun, E. Thamo, M. Wu, and X. Yi, "A survey of safety and trustworthiness of deep neural networks: Verification, testing, adversarial attack and defence, and interpretability," *Computer Science Review*, vol. 37, p. 100270, 2020.
[51] C. R. Qi, H. Su, K. Mo, and L. J. Guibas, "Pointnet: Deep learning on point sets for 3d classification and segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 652–660.
[52] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," *arXiv preprint arXiv:1509.02971*, 2015.
[53] F. Torabi, G. Warnell, and P. Stone, "Behavioral cloning from observation," *arXiv preprint arXiv:1805.01954*, 2018.
[54] J. D. Gammell, S. S. Srinivasa, and T. D. Barfoot, "Bit*: Batch informed trees for optimal sampling-based planning via dynamic programming on implicit random geometric graphs," *arXiv preprint arXiv:1405.5848*, 2014.