2025 American Control Conference (ACC)
July 8-10, 2025. Denver, CO, USA

LQR-CBF-RRT*: Safe and Optimal Motion Planning

Guang Yang[1], Mingyu Cai[2], Ahmad Ahmad[3], Amanda Prorok[1], Roberto Tron[3] and Calin Belta[4]

***Abstract*— We present LQR-CBF-RRT*, an incremental sampling-based algorithm for offline motion planning. Our framework leverages the strength of Control Barrier Functions (CBFs) and Linear Quadratic Regulators (LQR) to generate safety-critical and optimal trajectories for general affine control systems. This work uses CBF for safety guarantees and LQRs for optimal control synthesis during edge extensions. Traditional CBF methods involve Quadratic Programs (QPs), which add computational overhead and can sometimes be infeasible. Conversely, LQR-based controllers typically employ first-order Taylor approximations for nonlinear systems, necessitating consistent recalculations. To enhance motion planning efficiency, our framework directly verifies CBF constraints during the planning process, thereby eliminating the need for QP solutions. Additionally, we cache optimal LQR gain matrices in a hash table to bypass re-computation during local linearizations in the rewiring phase. To further boost sampling efficiency, we integrate the Cross-Entropy Method. Our results demonstrate that the proposed planner outperforms existing algorithms in computational efficiency and exhibits robust performance in real-world experiments.**

I. Introduction

Robot motion planning involves computing an optimal plan that guides a robot safely and efficiently towards a goal. Sampling-based planners, such as Rapid-exploring Random Tree (RRT) [1], have been widely used to solve this problem. Given the popularity of RRT*-based planning algorithms, there has been an extensive effort to reduce the sampling computational complexity by exploiting the problem structure. An example is Informed RRT*, [2], which outperforms RRT* in terms of convergence rate.

Classic RRT or RRT* variants do not consider dynamics during planning and assume a collision checking method exists during trajectory extension and sampling. Several studies, such as those in [3] and [4], have improved the collision checking procedure. However, these improvements can be computationally expensive and require specific solutions for various nonlinear systems. To improve sampling efficiency, the works in [5], [6] employ an importance sampling (IS) algorithm, for efficient exploration and sampling for RRT*. Our work aims to create a complete sampling-based motion planner that can efficiently generate samples from an optimal distribution, while ensuring optimality and safety subject to dynamics constraints.

Control Barrier Function (CBF) The significant advantage of CBFs comes from their guarantees on *forward invariance* [7], i.e., if the system trajectory is initialized in a safe set. The standard formulation of a CBF-based controller involves Quadratic Programs (QPs) and applies the generated control inputs using zero-order hold (ZOH) controllers [8], whereby each QP is constrained by a Control Lyapunov Function (CLF) for stability and CBFs for safety (see [9]). For motion planning, the first CBF-based sampling-based motion planning algorithm was introduced in [10]. In contrast to [11], [12], the *forward invariance* property from CBFs removes the requirement of explicit collision checking. The approach can generalize to any control affine nonlinear system, and the safety sets require no assumption on linearity or convexity. Later works, such as [6], [13], were built on top of it to improve its computational and sampling efficiency. Inspired by RRT* [14] and [10], the work in [6], Adaptive CBF-RRT*, implemented a rewiring procedure to improve the trajectory cost. However, the computation is costly due to the iterative calculation of the QPs.

Linear Quadratic Regulator(LQR) An LQR [15], [16] is a widely used optimal control strategy that minimizes a quadratic cost function over a system's states and control inputs. Other variations, such as iterative LQR (iLQR) [15], [17] can handle nonlinear system dynamics. In these methods, the cost function is approximated using a second-order Taylor expansion.

In prior works [6], [10], CLFs and CBFs are used as QP constraints, and it is solved iteratively to generate controls. However, this approach has a few downsides: (*i*) Solving a sequence of QPs in the steering function can be costly, especially when the step size is selected to be small [10]; (*ii*) The QPs can easily become infeasible [18]; and (*iii*) The QP controllers only ensure sub-optimal controls as the formulation of the objective function only guarantees optimality point-wise in time [6].

Contribution We propose an efficient and safe sampling-based offline motion planner that optimizes an LQR cost, while accounting for system dynamics. The generated state trajectory can then be tracked by feedback controllers online. This work specifically focuses on offline optimal planning using RRT*-like approaches [19]–[21]. Therefore, dynamic obstacles and online planning are not within the scope of this paper. Our proposed approach has been shown to improve efficiency significantly from several aspects. Lastly, we open-sourced our code to increase accessibility [1]. In summary,

- Our sampling based method generates optimal controls through offline planning while guaranteeing safety.
- We eliminate the need to formulate and solve quadratic programs repeatedly, in contrast to [6], [10].
- We validate our approach in both simulation and real-world experiments with a holonomic robot.

[1]https://github.com/mingyucai/LQR_CBF_rrtStar

[1]Guang Yang and Amanda Prorok are with the Department of Computer Science, University of Cambridge, Cambridge, UK.

[2]Mingyu Cai is with the Department of Mechanical Engineering, University of California Riverside, CA, USA.

[3]Ahmad Ahmad, Roberto Tron are with the Division of Systems Engineering, Boston University, Boston, MA, USA

[4]Calin Belta is with the Department of Electrical and Computer Engineering, University of Maryland, College Park, MD, USA

The research is funded in part by a gift from Arm ®. Their support is gratefully acknowledged.

II. Preliminaries and Problem Formulation

Notation: A function $f : \mathbb{R}^n \mapsto \mathbb{R}^m$ is called *Lipschitz continuous* on $\mathbb{R}^n$ if $\exists \mathcal{L} \in \mathbb{R}^+$, such that $\|f(y) - f(x)\| \leq \mathcal{L}\|y - x\|, \forall x, y \in \mathbb{R}^n$. For a continuously differentiable function $h : \mathbb{R}^n \mapsto \mathbb{R}$, we use $\dot{h}$ to denote the derivative with respect to time t; $\pounds_f^r h(x) := \frac{\partial \pounds^r h(x)}{\partial x} f(x)$ and $\pounds_g^r \pounds_f^{r-1} h(x) := \frac{\partial \pounds_f^{r-1} h(x)}{\partial x} g(x)$ are the r-th-order Lie derivatives [22]. A continuous function $\alpha : [-b, a) \mapsto [-\infty, \infty)$, for some $a > 0, b > 0$, which is strictly increasing and $\alpha(0) = 0$, is called a class $\mathcal{K}$ function. We denote $B_r(x)$ as a ball of radius r centered at $x \in \mathbb{R}^n$.

A. System Dynamics

Consider a continuous time dynamical system

$$\dot{x} = f(x) + g(x)u, \tag{1}$$

with $x \in \mathcal{X} \subset \mathbb{R}^n$, $u \in \mathcal{U} \subset \mathbb{R}^m$, where $\mathcal{X}$ and $\mathcal{U}$ are the state and control spaces, respectively. We assume that $f(x) : \mathbb{R}^n \mapsto \mathbb{R}^n$ and $g(x) : \mathbb{R}^n \mapsto \mathbb{R}^{n \times m}$ are locally Lipschitz continuous. We assume that the obstacles in the workspace are already mapped in the state space and described as regions in $\mathcal{X}$, i.e., $\mathcal{X}_{\text{i,obs}} \subset \mathcal{X}, i = 1, \ldots, \text{N}_{\text{obs}}$. The obstacle-free set is $\mathcal{X}_{\text{safe}} := \mathcal{X} \setminus \cup_{i=1}^{\text{N}_{\text{obs}}} \mathcal{X}_{\text{i,obs}}$.

B. Linear Quadratic Regulator(LQR)

We use LQR to compute optimal control policies. The cost function with infinite time horizon is defined as

$$J = \int_0^\infty x^T Q x + u^T R u \, dt, \tag{2}$$

where $Q = Q^T \succeq 0$ and $R = R^T \succ 0$ are weight matrices for state x and control u, respectively. For systems with linear, time-invariant dynamics, we can compute the closed form solution for the optimal control. Given a linear system

$$\dot{x} = Ax + Bu, \tag{3}$$

and the cost function (2), we can compute the LQR gain matrix K_{LQR} by solving the algebraic Riccati for matrix P:

$$A^T P + PA - PBR^{-1}B^T P + Q = 0. \tag{4}$$

The optimal control policy is in the form

$$\pi^*(x) = -K_{\text{LQR}} x, \tag{5}$$

where $K_{\text{LQR}} = R^{-1}B^T P$ and P is a stabilizing solution [23]. For a nonlinear system, a linearization can be made w.r.t. an equilibrium point (x_{eq}, u_{eq}) using first-order Taylor expansion. Specifically, given a nonlinear system $\dot{x} = F(x, u)$ and an equilibrium $(x_{\text{eq}}, u_{\text{eq}})$, i.e., $F(x_{\text{eq}}, u_{\text{eq}}) = 0$, we have

$$\dot{x} \approx \frac{\partial F(x_{\text{eq}}, u_{\text{eq}})}{\partial x}(x - x_{\text{eq}}) + \frac{\partial F(x_{\text{eq}}, u_{\text{eq}})}{\partial u}(u - u_{\text{eq}}).$$

The linearized system can be written as $\dot{\hat{x}} = \hat{A}\hat{x} + \hat{B}\hat{u}$, with $\hat{A} = \frac{\partial F(x_{\text{eq}}, u_{\text{eq}})}{\partial x}$ and $\hat{B} = \frac{\partial F(x_{\text{eq}}, u_{\text{eq}})}{\partial u}$, and an optimal controller can be designed as described above.

C. Control Barrier Functions

Given a continuously differentiable function $h(x) : \mathbb{R}^n \mapsto \mathbb{R}$, we denote the r_b-th derivative of $h(x)$ with respect to time t as $h^{r_b}(x) = \pounds_f^{r_b} h(x) + \pounds_g^{r_b} \pounds_f^{r_b - 1} h(x) u$. The *relative degree* $r_{\text{b}} \geq 0$ of h with respect to dynamics (1) is defined as the smallest natural number such that $\pounds_g \pounds_f^{r_{\text{b}}-1} h(x) u \neq 0$.

Given a time-varying function $h : \mathbb{R}^n \times [t_0, \infty) \mapsto \mathbb{R}$ that is r_b-th order differentiable, we define

$$\Psi_r(x, t) = \dot{\Psi}_{r-1} + \alpha_r(\Psi_{r-1}(x, t)), r = 1, \ldots, r_b, \tag{6}$$

with $\Psi_0(x, t) = h(x, t)$. These functions define a series of safety sets as:

$$\mathfrak{C}_r = \{x \in \mathbb{R}^n | \Psi_{r_b}(x, t) \geq 0\}, r = 1, \ldots, r_b. \tag{7}$$

Definition 1. *[24] Given the functions defined in* (6) *and safety sets* (7)*, the function $h : \mathbb{R}^n \times [t_0, \infty) \mapsto \mathbb{R}$ is a High Order Control Barrier Function (HOCBF) for system* (1) *if there exists class $\mathcal{K}$ functions $\alpha_1, \ldots, \alpha_{r_b}$ such that*

$$\Psi_{r_b}(x(t), t) \geq 0, \tag{8}$$

$\forall\ (x, t) \in \mathfrak{C} \times [t_0, \infty)$, where $\mathfrak{C} := \mathfrak{C}_0 \cap \cdots \cap \mathfrak{C}_{r_b}$.

Definition 2. *Given an initial state $x_0 = x(0)$, the set $\mathfrak{C}$ is called forward invariant if for every $x_0 \in \mathfrak{C}$, $x(t) \in \mathfrak{C}, \forall t$.*

Theorem 1. *[7] Given system 1 and a continuous function h, if there exists a HOCBF as in Def. 1, then the set $\mathfrak{C}$ is forward invariant (i.e., the system is safe).*

Definition 3 (Safe Trajectory)**.** *We define a safe trajectory for system (1) in $\mathfrak{C}$ as $\sigma := (\mathbf{u}, x)$, where given control inputs $\mathbf{u} : [0, T] \to \mathcal{U}$ the produced system trajectory $x(t)$ is subject to (1) and $x(t) \in \mathfrak{C} \subseteq \mathcal{X}_{\text{safe}},\ \forall t \in [0, T]$.*

Definition 4 (Trajectory Cost)**.** *Given a produced control and state trajectory $\sigma = (\mathbf{u}, x)$, with $\mathbf{u} : [0, T] \to \mathcal{U}$, we define the state and control trajectory cost as follows.*

$$\text{c}(\sigma) := \int_0^T x^T Q x + u^T R u \, dt, \tag{9}$$

where Q and R are the same weight matrices defined in (2)*.*

We are now ready to formulate the problem that we consider in this paper:

Problem 1. Given an initial state $x_{\text{init}} \in \mathcal{X}_{\text{init}} \subset \mathfrak{C} \subseteq \mathcal{X}_{\text{safe}}$, where $\mathcal{X}_{\text{init}}$ is an initial obstacle free set, and a bounded goal region with $\mathcal{X}_{\text{goal}} = B_{r_{\text{goal}}}(x_{\text{goal}})$ for some pre-defined radius r_{goal}, such that $\mathcal{X}_{\text{goal}} \subset \mathfrak{C}$. Find a control input $\mathbf{u} : [0, T] \mapsto \mathcal{U}$, where $T \in \mathbb{R}^+$ is the time horizon, that produces a trajectory σ such that $x(T) \in \mathcal{X}_{\text{goal}}$ and $x(t) \in \mathfrak{C}, \forall t \in [0, T]$, with the optimal cost of the (9), $\text{c}^*(\sigma)$, being the optimal one in $\mathfrak{c}$, i.e.,

$$\text{c}^*(\sigma) := \text{argmin}_{\mathbf{u} \in \mathcal{U}, \forall t \in [0,T],\ T \in \mathbb{R}^+,\ (1)} \text{c}((\mathbf{u}, x)) \tag{10}$$

III. LQR-CBF-RRT*

In this section, we introduce our LQR-CBF-RRT* algorithm (summarized in Alg. 1) as an extension of the CBF-RRT algorithm from [10]. LQR-CBF-RRT* incorporates HOCBF constraints (8) for checking collision avoidance and the LQR method (5) for synthesis of optimal control.

To solve Problem 1, our framework encodes goal-reaching and safety constraints in the *steering* function (Line 5, 11

in Algorithm 1). To ensure asymptotic optimality [25], an *rewiring* procedure (Line 16 in Algorithm 1) is included. For each iteration, a state, denoted as x_{samp} is sampled from a uniform distribution in the configuration space. The x_{samp} is then used as an input in function `NearbyNode` to find a set $\mathcal{X}_{\text{near}}$ of its nearby nodes. The `Nearest` is used to select its closest neighbor node x_{nearest} from the tree. Next, the function `LQR-CBF-Steer` 2 synthesize controls based on the computed optimal gain and steer the state from x_{nearest} to x_{samp}. The resultant trajectory is denoted as σ and its end node in σ is x_{new}. At the stage of `ChooseParent`, the function takes in the computed x_{new} and set $\mathcal{X}_{\text{near}}$ to compute state trajectories and corresponding costs (9) from x_{new} to all the nearby nodes defined in $\mathcal{X}_{\text{near}}$. The trajectory with the minimum cost, denoted as σ_{min}, is used and its end node is denoted as x_{min}. Finally, x_{new} and its edge are added to the tree. To asymptotically optimize the existing tree, the `Rewire` function is used. During this procedure, the `LQR-CBF-Steer` is used to check the cost of each nearby node $x_{\text{nn}} \in \mathcal{X}_{\text{near}}$ can be further optimized. The x_{nn} that returns the minimum costs is rewired to x_{new}, and the corresponding cost is updated.

A. LQR-CBF-RRT* Algorithm

The detailed algorithm is introduced as the following. We define a tree $\mathcal{T} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ is a set of nodes and $\mathcal{E}$ is a set of edges.

- **Nearby Node** uses an Euclidean distance d to find a set of nearby nodes $\mathcal{X}_{\text{near}}$ in $\mathcal{T}$ that is closest to x_{sample}.
- **Nearest** returns the nearest node from $\mathcal{X}_{\text{near}}$ w.r.t. x_{sample}.
- **ChooseParent** The procedure (Algorithm 1 Line 10-14) tries to find collision-free trajectories between x_{new} w.r.t. all its neighboring nodes. The `ChooseParent` procedure selects the x_{nn} with the lowest cost (2).
- **Rewire** is defined in Algorithm 1 Line 16-20. It evaluates and optimizes the LQR cost (2). This function checks a selected node's neighborhood and calculates the costs w.r.t. all the neighboring nodes. A new state trajectory from the current node to the nearby node is added to $\mathcal{T}$.
- **LQRSolver** The method (Algorithm 1 Line 4) computes optimal gain matrix K_{LQR} from Riccati equation (4). For a nonlinear system, the system can be linearized locally [19] and then computes K_{LQR}.
- **LQR-CBF-Steer** The steering function (Algorithm 2) contains two components: (i) The LQR controller (ii) CBF safety constraints. Given two states $(x_{\text{current}}, x_{\text{next}})$, the LQR controller generates a sequence of optimal controls u_i^*, that steers the state trajectory based on (3). At each time steps, all CBF constraints (8) are checked. If none of the CBF constraints are active, then the `LQR-CBF-steer` is used for steering x_{current} to x_{next}, and node x_{next} is added to the tree. Otherwise, the extension stops at the first encounter of (8) when it is violated.

IV. Efficiency Improvements

In this section, we develop three techniques to improve the efficiency of the sampling process. Notably, when integrating

Algorithm 1: LQR-CBF-RRT*

1 **Initialization:** $\mathcal{T} = (\mathcal{V}, \mathcal{E})$, $\mathcal{V} \leftarrow \{x_{\text{init}}\}$; $\mathcal{E} \leftarrow \emptyset$; $i = 0$; `adapFlag = True`, `optDensityFlag = False`, and $r = \eta$
2 **while** $i < N$ **do**
3 $x_{\text{samp}} \leftarrow$ `Sample`$(\mathcal{G}, \mathcal{V}$, `adapFlag`$)$
4 $x_{nearest} \leftarrow$ `Nearest`$(\mathcal{V}, x_{\text{samp}})$
5 $x_{\text{new}}, \sigma \leftarrow$ `LQR-CBF-Steer`$(x_{\text{nearest}}, x_{\text{samp}})$
6 $\mathcal{X}_{near} \leftarrow$ `NearbyNode`$(\mathcal{V}, x_{nearest})$
7 $r = \min\{\lambda(\log(|\mathcal{V}|)/|\mathcal{V}|)^{1/(d+1)}, \eta\}$
8 $\mathcal{X}_{near} \leftarrow$ `Near`$(\mathcal{V}, r, x_{new})$
9 minCost $\leftarrow \infty$; $x_{\text{min}}, \sigma_{\text{min}} \leftarrow$ None, None
10 **foreach** $x_{\text{near}} \in \mathcal{X}_{\text{near}}$ **do**
11 $\sigma \leftarrow$ `LQR-CBF-Steer` $(x_{\text{new}}, x_{\text{nn}})$
12 **if** $x_{\text{nn}}.cost$+*Cost*$(\sigma) <$ *minCost* **then**
13 minCost $\leftarrow x_{\text{nn}}$.cost+Cost(σ)
14 $x_{\text{min}} \leftarrow x_{\text{nn}}$; $\sigma_{\text{min}} \leftarrow \sigma$
15 $\mathcal{V} \leftarrow \mathcal{V} \cup \{x_{new}\}$; $\mathcal{E} \leftarrow \mathcal{E} \cup \{x_{\text{min}}, x_{\text{new}}\}$
16 **foreach** $x_{\text{nn}} \in \mathcal{X}_{near}$ **do**
17 $\sigma \leftarrow$ `LQR-CBF-Steer`$(x_{\text{new}}, x_{\text{nn}})$
18 **if** $x_{\text{new}}.cost$+*Cost*$(\sigma) < x_{\text{nn}}.cost$ **then**
19 x_{nn}.parent $\leftarrow x_{\text{new}}$
20 x_{new}.cost =Cost(x_{new})
21 $\mathcal{T}$, $\mathcal{G} \leftarrow$ `extToGoal`$(\mathcal{V}, x_{\text{new}}$, `adapFlag`$)$; $i \leftarrow i + 1$
22 **return** $\mathcal{T}$

Algorithm 2: LQR-CBF-Steer$(x_{\text{current}}, x_{\text{next}})$

1 $\mathbf{x} \leftarrow$ None, $\mathbf{u} \leftarrow$ None
2 $\mathbf{x}$.add(x_{current})
3 $\text{K}_{\text{LQR}} \leftarrow$ `LQRsolver`$(x_{\text{current}}, x_{\text{next}})$
4 **foreach** $t <$ T **do**
5 $x^{'}, u \leftarrow$ `Integrator`$(x_{\text{current}}, \text{K}_{\text{LQR}})$
6 **if** *Satisfied* $\leftarrow$ *CBFconstraints*$(x^{'}, u)$ **then**
7 $\mathbf{x}$.add$(x^{'})$; $\mathbf{u}$.add(u)
8 $x_{\text{current}} \leftarrow x^{'}$
9 **else**
10 **return** $\sigma = (\mathbf{x}, \mathbf{u})$

RRT* with both linear and nonlinear systems, section IV-A shows how our LQR steering function has better performance compared to offline MPC and iLQR. Moreover, section IV-B demonstrates a QP-free mechanism to check the satisfaction of CBF constraints. Lastly, we propose an adaptive sampling method to enhance sampling efficiency.

A. Efficient LQR computation

For each steering process, applying the MPC [26] and iLQR [27] offline requires iteratively solving optimization problems during edge extension. These approaches are not efficient as sampling-based methods need explore the state space, which leads to the fundamental motivation for applying LQR in our framework, i.e., providing an efficient steering procedure for exploring the state space. For linear dynamic systems, $\dot{x} = Ax + Bu$. The LQR feedback gain K_{LQR} can be computed through Riccati equation in (4) i.e., $K_{\text{LQR}} = LQR(A, B, Q, R)$, which depends on pre-modeled matrices A, B, Q, R. For nonlinear systems, we can linearize it around a local equilibrium point and extract the linearized system (demonstrated in Section II).

B. LQR–CBF-RRT* without solving QP

In this work, instead of solving the QPs, we only check whether the sampled state-action pair satisfies CBF constraints (8) and use them as the termination condition of the steering process. For example, given a reference

Algorithm 3: $x_{\text{samp}} \leftarrow \texttt{Sample}(\mathcal{G}, \mathcal{T}, m)$

```
u ~ Uniform(0, 1)
if u ≤ 0.5 ∧ G ≠ ∅ then
    if optDensityFlag then
        X ~ ĝ*(x)
        return (x)
    else
        if mod(|G|, n_v) = 0 then
            E ← Quantile(G, ϱ)                ▷ Assign the elite set
            ĝ(x) ← CE_Estimation(E, m)        ▷ Compute PDF of E
            return x ~ ĝ(x)
        else
            return x ~ Uniform(X)
else
    return x ~ Uniform(X)
```

trajectory $x_0u_0x_1 \dots u_{n-1}x_n$ from LQR optimal control, to reach the sampled goal from current state $x_{current} = x_0$. We check if moving from x_i to x_{i+1} via control u_i for all $i \in \{0, 1, \dots, n-1\}$ satisfies the CBF condition. We terminate the extension process if the CBF condition is violated at any step i, which guarantees that the generated system trajectory is forward invariant in $\mathfrak{C}$ (see Definition 2).

C. Adaptive Sampling

We leverage LQR-CBF-RRT* with an adaptive sampling procedure [6] to focus sampling promising regions of $\mathcal{X}_{\text{safe}}$ in order to approximate the solution of Problem 1 with a fewer number of samples. We define $\mathcal{G}$ as set of control and state trajectories pairs, i.e., $\mathcal{G} := \{(\mathbf{x}(\mathbf{t}), \mathbf{u}(\mathbf{t}))|x(t) \in \mathcal{X}_{\text{safe}}, \forall t \in [0, T]\}$. The Cross Entropy Method (CEM) is used for IS. First, it generates samples from a current Sampling Density Function (SDF) and computes the cost of each sample; second, it chooses an *elite subset*, $\mathfrak{E}$; finally, the elite subset is used to estimate a probability density function (PDF) as if they were drawn as i.i.d samples and we define m to be the number of elite trajectories.

V. Probabilistic Completeness and Optimality

First, we provide the probabilistic proof for our algorithm. Following [14] we assume that Problem 1 is *robustly feasible* with minimum clearance $\varepsilon > 0$. Hence, $\exists \mathbf{u} \in \mathcal{U}$ such that it produces a system (1) trajectory ϕ, where $\phi : [0, T] \mapsto \mathcal{X}_{\text{safe}}$, $\phi(0) = x_{\text{init}}$, $\phi(T) = x_{\text{goal}}$, and $\forall t \in [0, T]$; $\Psi_{r_b}(\phi(t)) \geq 0, \forall \phi \in \mathfrak{C}_0 \cap \dots \cap \mathfrak{C}_{r_b}$.

Lemma 1. *[28] Given two trajectories ϕ and ϕ', as well as a period $T \geq 0$, such that $\phi(0) = \phi'(0) = x_{\text{init}}$, the trajectories can be bounded by control with $K_u, K_x \in \mathbb{R} > 0$.*

$$\|\phi'(T) - \phi(T)\| < K_u \cdot T \cdot e^{K_x \cdot T} \cdot \sup(\|u(t) - u'(t)\|), \quad (11)$$

Lemma 1 ensures any two trajectories with the same initial state, the distance between their end states at time T is bounded by the largest difference in their controls.

Remark 1. Consider steering from any $x_s \in \mathcal{X}_{safe}$ towards any reachable $x_f \in \mathcal{X}_{safe}$ using Algorithm 2. By selecting appropriate class $\mathcal{K}$ functions, α_1, ..., α_ρ, of the HOCBF (see Definition 8), we can produce $\mathbf{u} = u(t_0), t(t_1), \dots, u(T_{OL})$, such that the bound (11) be given as $||x_f - x'_f|| = \mu < \frac{\varepsilon}{4}$, where x_f and x'_f are the states at time T_{OL} of the produced trajectories under the control inputs $\mathbf{u}_{OL}$ and the CBF-QP control inputs $\mathbf{u}_{LP}$, respectively, and $\mu \in \mathbb{R}_{>0}$.

Theorem 2. *The LQR-CBF-RRT* (Algorithm 1) is probabilistically complete.*

Proof. The completeness of RRT* is implied by the completeness of RRT [14]. Therefore, we need to prove the completeness of LQR-CBF-RRT. One can implement LQR-CBF-RRT* by mitigating Lines 9-14 and Lines 16-21 in Algorithm 1. Given that the local motion planner 2, and by leveraging Theorem 2 in [29], we prove the incremental state trajectory will propagate to a sequence of intermediate states until reaching $\mathcal{X}_{\text{goal}}$. Assuming the trajectory ϕ of the solution of Problem 1 with ε clearance and has a length L. Considering $m + 1$ equidistant states $x_i \in \phi,\ i = 1, \dots, m + 1$, where $m = \lfloor \frac{4L}{\varepsilon} \rfloor$, we define a sequence of balls with radius $\varepsilon/4$ that are centered at these states. For state x_i, such ball is given by: $\mathfrak{B}_{\frac{\varepsilon}{4}}(x_i) := \{x_b \mid ||x_i - x_b|| \leq \frac{\varepsilon}{4}\}$. For the consecutive states $x_i, x_{i+1} \in \varphi_x$, we want to prove that starting from $x_s \in \mathfrak{B}_{\frac{\varepsilon}{2}}(x_i)$ the steering function `LQR-CBF-Steer` is able to generate a control and state trajectories that its end state x'_f fall in $\mathfrak{B}_{\frac{\varepsilon}{4}}(x_{i+1})$. Given Remark 1, we assign $\eta = \frac{\varepsilon}{4} + \mu + 2\iota$ and $0 < \iota < \frac{\varepsilon}{4} - \mu$. Next, $\mathfrak{B}_\eta(x_s)$ and $\mathfrak{B}_{\frac{\varepsilon}{4}-\mu-\iota}(x_{i+1})$ are assigned at x_s and x_{i+1}, respectively. Let $\mathcal{S} := \mathfrak{B}_\eta(x_s) \cap \mathfrak{B}_{\frac{\varepsilon}{4}-\mu-\iota}(x_{i+1})$ denotes the successful potential end-states set. For any $x_f \in \mathcal{S}$, `LQR-CBF-Steer` generates trajectories that fall in $\mathfrak{B}_\mu(x_f) \subset \mathfrak{B}_{\frac{\varepsilon}{4}}(x_{i+1})$. We denote $|.|$ as the Lebesgue measure, then, for x_s, the probability of generating states in $\mathcal{S}$ is $p = \frac{|\mathcal{S}|}{|\mathcal{X}|}$ and is strictly positive. The probability p is the success probability of the k Bernoulli trials process [29] that models generating m successful outcomes of sampling states that incrementally reach $\mathcal{X}_{goal}$. The rest of the proof follows the proof of Theorem 1 in [29]. □

Next, we present the proof of asymptotic optimality for our LQR-CBF-RRT* algorithm.

Assumption 1. *We assume that we have access to a (conservative) safety set $\mathfrak{C} \subseteq \mathcal{X}_{safe}$ in which the procedure LQR-CBF-Steer$(x_{\text{current}}, x_{\text{next}})$ is able to produce a trajectory that converges to x_{next}.*

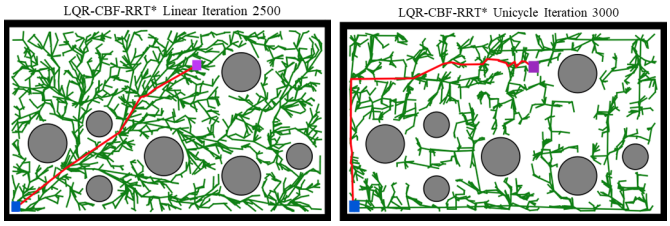
Assumption 1 suggests the algorithm yields an asymptotically optimal trajectory within a conservative subset of the environment. Provided that our algorithm is an RRT* variant with an LQR-based local motion planner with a CBF-based sample rejection method, in the following theorem, we provide an asymptotic optimality result. At the i-th iteration of LQR-CBF-RRT*, we consider the trajectory, σ^i, which connects $x_{\text{init}}(0)$ to an $x(T) \in \mathcal{X}_{\text{goal}}$, as a concatenation of trajectories produced by `LQR-CBF-Steer`.

Theorem 3. *Consider the cost of the optimal solution of Problem 1 to be $\mathrm{c}^*(\sigma)$; the LQR-CBF-RRT* (Algorithm 1) produces an asymptotically optimal solution in $\mathfrak{C}$ to Problem 1. i.e.,*

$$P(\lim_{i \to \infty} \mathrm{c}(\sigma^i) = \mathrm{c}^*(\sigma)) = 1 \quad (12)$$

Proof. (Sketch) Given Assumption 1, the asymptotic optimality of the solution follows directly from Theorem 5 in [30].

□



(a) (b)

Fig. 1: Demonstration of our planner in simulations on a double integrator (1a) and a unicycle model (1b). The blue square represents the starting position and the purple square is the goal.

VI. Experimental Results

In this section, we illustrate our approach on different systems (Fig. 1), with the initial position $p_{\text{init}} = [2, 2]$ and the goal position $p_{\text{goal}} = [30, 24]$ with circular obstacles. The simulations are performed on a MacBook Pro with an M1 Pro CPU.

A. Baseline Summary

We compare several CBF-RRT like planning techniques against our framework: (1) avoiding the construction QP, (2) storing previous LQR feedback gain, and (3) adaptive sampling. For the nonlinear system, we compare: (1) Our method. (2) LQR-CBF-RRT* without storing feedback gain for nonlinear systems, i.e., naive adaptive LQR-CBF-RRT*; (3) LQR-CBF-RRT* without both adaptive sampling and storing feedback gain. (4) QP based LQR-CBF-RRT*. For the linear system, we compare: (1) Our method. (5) LQR-CBF-RRT* with QP-solver during steering processes, i.e., LQR-CBF-RRT*-QP.

Nonlinear System Given unicycle model, we performed five experiments with 2000 iterations for each baseline with different random seeds. In table I, we list the time it takes to complete the motion planning for different baselines.

Based on the result, implementing a hash table for optimal gain K_{LQR} can significantly improve the performance, as it avoids the repeated linearization and recalculation of the gain matrices. The graphical result can be found in Fig. 2.

TABLE I: Efficiency Comparison (Unit: Seconds)

Baseline	Seed 0	Seed 20	Seed 42	Seed 45	Seed 100	Mean	Std
Ours	**10.04**	**9.06**	**9.56**	**10.27**	**11.14**	**10.01**	**0.70**
(2)	32.07	27.8	29.41	34.86	35.66	31.96	3.03
(3)	74.34	61.6	109.92	104.64	92.24	88.55	18.20
(4)	41.32	T/O	45.57	42.18	43.22	43.07	1.84

Linear System For the linear system comparison, our proposed method is about 86% faster than using LQR-CBF-RRT*-QP (Baseline (5)). The result can be found in Table II. Our method is compared with an offline MPC with

TABLE II: Efficiency Comparison (Unit: Seconds)

Baseline	Seed 0	Seed 20	Seed 42	Seed 45	Seed 100	Mean	Std
Ours	**11.3**	**11.15**	**9.75**	**10.91**	**12.42**	**11.11**	**0.95**
(5)	80.76	79.17	78.42	81.90	80.72	80.194	1.39

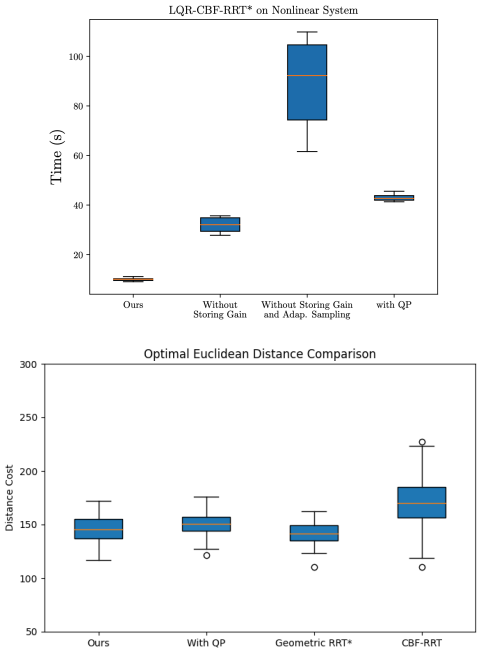


Fig. 2: Our method outperforms the other cases by a significant margin for a nonlinear system. The last column (with QP) was the method implemented in [10].

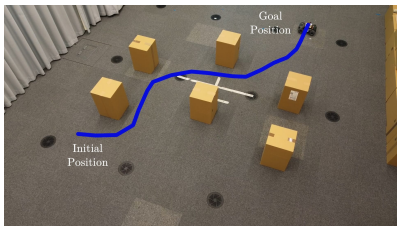


Fig. 3: LQR-CBF-RRT* on an holonomic ground robot.

time horizon of 5 and discrete time interval of 0.05. The offline MPC method went through 2000 iterations over 41 minutes. Finally, we performed an optimal Euclidean distance comparison Fig. 2 to show LQR-CBF-RRT* optimizes the euclidean distance.

Example 1 In this example, we performed our sampling-based motion planner on a double integrator model with the linear dynamics with the state $[x_1, x_2, x_3, x_4] : \ddot{x_1} = u_1$ and $\ddot{x_3} = u_2$, where $[x_1, x_3]$ is the position, $[x_2, x_4]$ is the velocity. We control the system using acceleration $[u_1, u_2]$. For the i-th obstacle, the corresponding i-th safety set can be defined based on the function $h_i(x) = (x_1 - x_{1,i,o})^2 + (x_3 - x_{3,i,o})^2 - r_i^2$, where $[x_{1,i,o}, x_{3,i,o}]$ is the centroid of the i-th obstacle and r_i is the radius. We iterate through in total of 2500 steps, and the final result is shown in Fig. 1a.

Example 2 In this example, we validate our framework on a unicycle simulator with abstracted system dynamics. The control input $u = [v, \omega]$ consists of the translational and angular velocity. The algorithm performs 3000 iterations, and the result can be found in Fig. 1b.

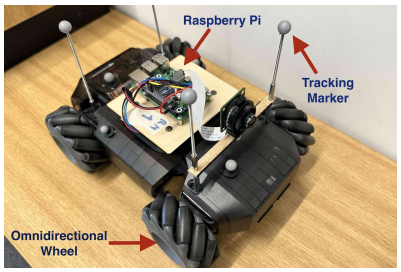


Fig. 4: Cambridge Robomaster [31].

B. Hardware Experiment

To evaluate the effectiveness of our planner in real-world scenario, an omnidirectional robot is used to validate our method. The obstacles are over-approximated with circular shapes with a radius of 0.25 meters for constructing the CBFs. To accurately track the robot's position, we employed an external telemetry system (OptiTrack), effectively emulating an outdoor GPS-enabled environment. The onboard Raspberry Pi computer is running ROS2 for communicating and processing control commands. A customized software stack [32] handles the robot's low-level control. The result is shown in Fig. 3. The video is available at [2].

VII. Conclusion

In this paper, we developed an optimal motion planner with safety guarantee. We employ the CEM method, further improving the efficiency of our algorithm during the sampling phase. To streamline the process further, LQR gain matrices are stored in a hash table to speed up the rewiring phase for nonlinear systems. Notably, our technique outperforms benchmark counterparts in comparative tests and demonstrates robust results through real-world experiments.

References

[1] S. M. LaValle *et al.*, "Rapidly-exploring random trees: A new tool for path planning," 1998.

[2] J. D. Gammell, S. S. Srinivasa, and T. D. Barfoot, "Informed rrt: Optimal sampling-based path planning focused via direct sampling of an admissible ellipsoidal heuristic," in *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2014, pp. 2997–3004.

[3] J. Bialkowski, S. Karaman, M. Otte, and E. Frazzoli, "Efficient collision checking in sampling-based motion planning," in *Algorithmic Foundations of Robotics X: Proceedings of the Tenth Workshop on the Algorithmic Foundations of Robotics*. Springer, 2013, pp. 365–380.

[4] J. Pan and D. Manocha, "Fast probabilistic collision checking for sampling-based motion planning using locality-sensitive hashing," *The International Journal of Robotics Research*, vol. 35, no. 12, pp. 1477–1496, 2016.

[5] M. Kobilarov, "Cross-entropy motion planning," *The International Journal of Robotics Research*, vol. 31, no. 7, pp. 855–871, 2012.

[6] 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.

[7] X. Xu, P. Tabuada, J. W. Grizzle, and A. D. Ames, "Robustness of control barrier functions for safety critical control," *IFAC-PapersOnLine*, vol. 48, no. 27, pp. 54–61, 2015.

[8] 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.

[9] R. Cheng, M. J. Khojasteh, A. D. Ames, and J. W. Burdick, "Safe multi-agent interaction through robust control barrier functions with learned uncertainties," in *2020 59th IEEE Conference on Decision and Control (CDC)*. IEEE, 2020, pp. 777–783.

[10] 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.

[11] A. Weiss, C. Danielson, K. Berntorp, I. Kolmanovsky, and S. Di Cairano, "Motion planning with invariant set trees," in *2017 IEEE Conference on Control Technology and Applications (CCTA)*. IEEE, 2017, pp. 1625–1630.

[12] A. Shkolnik, M. Walter, and R. Tedrake, "Reachability-guided sampling for planning under differential constraints," in *2009 IEEE International Conference on Robotics and Automation*. IEEE, 2009, pp. 2859–2865.

[13] 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.

[14] S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning," *The international journal of robotics research*, vol. 30, no. 7, pp. 846–894, 2011.

[15] W. Sun, J. van den Berg, and R. Alterovitz, "Stochastic extended lqr for optimization-based motion planning under uncertainty," *IEEE Transactions on Automation Science and Engineering*, vol. 13, no. 2, pp. 437–447, 2016.

[16] R. Tedrake, I. R. Manchester, M. Tobenkin, and J. W. Roberts, "Lqr-trees: Feedback motion planning via sums-of-squares verification," *The International Journal of Robotics Research*, vol. 29, no. 8, pp. 1038–1052, 2010.

[17] J. Van Den Berg, "Iterated lqr smoothing for locally-optimal feedback control of systems with non-linear dynamics and non-quadratic cost," in *2014 American control conference*. IEEE, 2014, pp. 1912–1918.

[18] J. Zeng, B. Zhang, Z. Li, and K. Sreenath, "Safety-critical control using optimal-decay control barrier function with guaranteed point-wise feasibility," in *2021 American Control Conference (ACC)*. IEEE, 2021, pp. 3856–3863.

[19] A. Perez, R. Platt, G. Konidaris, L. Kaelbling, and T. Lozano-Perez, "Lqr-rrt*: Optimal sampling-based motion planning with automatically derived extension heuristics," in *2012 IEEE International Conference on Robotics and Automation*. IEEE, 2012, pp. 2537–2542.

[20] F. Islam, J. Nasir, U. Malik, Y. Ayaz, and O. Hasan, "Rrt*-smart: Rapid convergence implementation of rrt* towards optimal solution," in *2012 IEEE international conference on mechatronics and automation*. IEEE, 2012, pp. 1651–1656.

[21] Y. Li, W. Wei, Y. Gao, D. Wang, and Z. Fan, "Pq-rrt*: An improved path planning algorithm for mobile robots," *Expert systems with applications*, vol. 152, p. 113425, 2020.

[22] H. K. Khalil, "Nonlinear systems third edition," *Patience Hall*, vol. 115, 2002.

[23] D. Bertsekas, *Dynamic programming and optimal control: Volume I*. Athena scientific, 2012, vol. 4.

[24] W. Xiao and C. Belta, "Control barrier functions for systems with high relative degree," in *Proc. of 58th IEEE Conference on Decision and Control*, Nice, France, 2019, pp. 474–479.

[25] K. Solovey, L. Janson, E. Schmerling, E. Frazzoli, and M. Pavone, "Revisiting the asymptotic optimality of rrt," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2020, pp. 2189–2195.

[26] E. F. Camacho and C. B. Alba, *Model predictive control*. Springer science & business media, 2013.

[27] Y. Tassa, T. Erez, and E. Todorov, "Synthesis and stabilization of complex behaviors through online trajectory optimization," in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2012, pp. 4906–4913.

[28] Y. Li, Z. Littlefield, and K. E. Bekris, "Asymptotically optimal sampling-based kinodynamic planning," *The International Journal of Robotics Research*, vol. 35, no. 5, pp. 528–564, 2016.

[29] M. Kleinbort, K. Solovey, Z. Littlefield, K. E. Bekris, and D. Halperin, "Probabilistic completeness of rrt for geometric and kinodynamic planning with forward propagation," *IEEE Robotics and Automation Letters*, vol. 4, no. 2, pp. x–xvi, 2019.

[30] S. Karaman and E. Frazzoli, "Optimal kinodynamic motion planning using incremental sampling-based methods," in *49th IEEE conference on decision and control (CDC)*. IEEE, 2010, pp. 7681–7687.

[31] J. Blumenkamp, A. Shankar, M. Bettini, J. Bird, and A. Prorok, "The cambridge robomaster: An agile multi-robot research platform," *arXiv preprint arXiv:2405.02198*, 2024.

[32] A. Shankar, S. Elbaum, and C. Detweiler, "Freyja: A full multirotor system for agile & precise outdoor flights," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2021, pp. 217–223.

[2] https://www.youtube.com/watch?v=yE2yhEQSSUY