2022 IEEE 61st Conference on Decision and Control (CDC)
December 6-9, 2022. Cancún, Mexico

Adaptive Sampling-based Motion Planning with Control Barrier Functions

Ahmad Ahmad, Calin Belta, and Roberto Tron

2022 IEEE 61st Conference on Decision and Control (CDC) | 978-1-6654-6761-2/22/$31.00 ©2022 IEEE | DOI: 10.1109/CDC51059.2022.9993278

***Abstract*— Sampling-based algorithms, such as Rapidly Exploring Random Trees (RRT) and its variants, have been used extensively for motion planning. Control barrier functions (CBFs) have been recently proposed to synthesize controllers for safety-critical systems. In this paper, we combine the effectiveness of RRT-based algorithms with the safety guarantees provided by CBFs in a method called CBF-RRT*. CBFs are used for local trajectory planning for RRT*, avoiding explicit collision checking of the extended paths. We prove that CBF-RRT* preserves the probabilistic completeness of RRT*. Furthermore, in order to improve the sampling efficiency of the algorithm, we equip the algorithm with an adaptive sampling procedure, which is based on the cross-entropy method (CEM) for importance sampling (IS). The procedure exploits the tree of samples to focus the sampling in promising regions of the configuration space. We demonstrate the efficacy of the proposed algorithms through simulation examples.**

I. Introduction

Many state-of-the-art single query motion planning algorithms rely on randomized sampling to explore the configuration space, and build a path from a starting point to a goal region incrementally. Such algorithms are appealing because they avoid building the configuration space explicitly, which might be challenging in high-dimensional spaces. Rather, in the search for a path to the goal, they generate exploration paths and check if they do not coincide with obstacles. Moreover, given the fact that paths are typically built incrementally, one can impose differential constraints on the samples to generate paths that are dynamically feasible.

Rapidly-exploring random trees (RRT) [1] and its variants (see, e.g. [2]) are sampling-based motion planning algorithms are simple to implement and are probabilistically complete [3]. RRTs aim to rapidly explore the configuration space and build a tree rooted at a starting configuration to find a path to a goal region. Karaman and Frazzoli [4] proposed RRT*, where each newly added vertex to the RRT tree is rewired with a possible better connection, for which the cost to reach the rewired vertex from the root vertex is reduced. This approach makes the path asymptotically optimal [5]. Given its success in motion planning, in the past decade, there has been a large number of research efforts to improve RRT* sampling. Examples include informed-RRT* [6], which constructs an informed elliptical sampling region that shrinks as the length of the path decreases, which leads to faster convergence to the optimal path. Kobilarov [7] introduced CE-RRT*, which uses the cross-entropy method (CEM) [8] for importance sampling (IS).

The work in [9] imposes differential constraints on the vertices of RRT* to produce feasible paths according to the robot kinodynamics. Recently, Wu *et al.* [10] developed rapidly-exploring random reachable set trees (R3T), which constrain the expansion of RRT to be in the vertices' approximated reachable sets, which helps with finding dynamically feasible, and collision-free, paths using fewer samples. Recent developments in controlling safety-critical systems using control barrier functions (CBF) ([11] and references therein), are exploited by Yang *et al.* in CBF-RRT [12]. The authors model a safe set that contains the collision-free configurations, which is then used with a CBF-based controller to generate inputs that expand the tree in the safe set. In [13], CBF-RRT is used to generate safe trajectories to navigate in environments with moving humans.

In this work, we develop two variants of RRT* in which we leverage the optimality of RRT* with safety guarantees provided by local planners in which we use CBFs. Based on our knowledge, this is the first work to utilize safety-based local planners with optimal sampling-based motion planner. The contributions of the proposed work are as follows. First, CBF-RRT* (§IV), an RRT* variant that is equipped with two local motion planners that generate CBF-based control inputs for expanding the RRT* tree (§III-A.2), and for steering to desired configurations when rewiring a vertex (§III-A.1). Using these local planners, we avoid the collision-checking procedure, where the trajectories are guaranteed to be in a safety set. Second, Adap-CBF-RRT* (§V), a variant in which we exploit the exploration tree to focus the sampling in promising regions. To do so, we incorporate the algorithm with adaptive sampling procedure using the cross-entropy method (CEM) with nonparametric density estimation (§V-A). The proposed work is validated through simulation example in §VI.

II. Problem Formulation and Approach

Consider a robot with a configuration $\boldsymbol{q} \in \mathcal{Q} \subset \mathbb{R}^d$, where $\mathcal{Q}$ is the configuration space and $\mathbb{R}^d$ is the d-dimensional Euclidean space. Let the robot dynamics be modeled as the following nonlinear affine control dynamics,

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

where $\boldsymbol{u} \in \mathcal{U} \subset \mathbb{R}^m$ is the control input, $\mathcal{U}$ is the allowable control set, and $f : \mathbb{R}^d \to \mathbb{R}^d$ and $g : \mathbb{R}^d \to \mathbb{R}^{d\times m}$ are assumed to be locally Lipschitz functions.

This work was partially supported by the NSF under grant IIS-2024606, by the MIT / Lincoln Lab, and by the ONR under MURI N00014-19-1-2571.

The authors are with the Division of System Engineering, Boston University, Boston, MA 02215, USA ahmadgh@bu.edu, cbelta@bu.edu, tron@bu.edu

Obstacle i, $i = 1, \ldots, n$, is denoted by $\mathcal{O}_i \subset \mathcal{Q}$ [1]. The obstacle-free configuration space, which we denote it as the safe configuration space, is given by $\mathcal{Q}_{safe} = \mathcal{Q} \setminus \bigcup_{i=1}^{n} \mathcal{O}_i$.

Following [7], for a time horizon $T \in \mathbb{R}_{>0}$, let $\varphi : [0, T] \times \mathbb{R}_{>0} \rightarrow \mathcal{U} \times \mathcal{Q}$, $\varphi(t, T) := (\boldsymbol{u}(t), \boldsymbol{q}(t))$ be pairs of a control input $\boldsymbol{u}(t) \in \mathcal{U}$, $\forall t \in [0, T]$ and the produced trajectory $\boldsymbol{q}(t) \in \mathcal{Q}$ that satisfies system (1).

For a given starting configuration $\boldsymbol{q}_{start} \in \mathcal{Q}_{safe}$ and a goal region $\mathcal{Q}_{goal} \subset \mathcal{Q}_{safe}$, we define the set $\mathcal{G}$ as the set of control inputs and the produced trajectory pairs, for which the trajectory to be in $\mathcal{Q}_{safe}$, starts at $\boldsymbol{q}_{start}$ and fall in $\mathcal{Q}_{goal}$. I.e., $\mathcal{G} := \{\varphi(t, T) \mid \boldsymbol{q}(0) = \boldsymbol{q}_{start},\ \boldsymbol{q}_T \in \mathcal{Q}_{goal},\ \boldsymbol{q}(t) \in \mathcal{Q}_{safe}, (1), \forall t \in [0, T], T \in \mathbb{R}_{>0}\}$. The cost functional of $\varphi \in \mathcal{G}$ is defined as $J(\varphi) := \int_0^T C(\varphi(t, T))dt$, where $C : \mathcal{U} \times \mathcal{Q} \rightarrow \mathbb{R}_{>0}$ is the running cost.

Problem 2.1 (Optimal Motion Planning Problem (OMPP)): Given a robot with system dynamics (1), a starting configuration $\boldsymbol{q}_{start} \in \mathcal{Q}_{safe}$, a goal region $\mathcal{Q}_{goal} \subset \mathcal{Q}_{safe}$, and the obstacle-free configuration space $\mathcal{Q}_{safe}$, find $\varphi^* \in \mathcal{G}$ that minimizes $J(\varphi)$, i.e., $\varphi^* = \underset{\varphi \in \mathcal{G},\, T \in \mathbb{R}_{>0}}{\arg\min} \ J(\varphi)$.

OMPP imposes a subsequent control problem of generating control inputs that guarantee that the produced system trajectory to be in $\mathcal{Q}_{safe}$. Moreover, OMPP is PSPACE-hard [14]. Kinodynamic RRT* [9] is used to approximate a solution for the problem, where a tree is built incrementally starting at $\boldsymbol{q}_{start}$ and expanded towards $\mathcal{Q}_{goal}$ while satisfying differential constraints on the expanded vertices. φ^* is approached asymptotically by rewiring the vertices of the tree.

Approach We develop an RRT* variant, in which we use local motion planners that generate control inputs which render the safe set $\mathcal{Q}_{safe}$ forward invariant in system (1). That is, under such control inputs, for an initial state that lies in $\mathcal{Q}_{safe}$, the system trajectory will lie in $\mathcal{Q}_{safe}$ for all future times. Furthermore, we improve the sampling performance by biasing the sampling density function (SDF) towards generating more samples in promising regions of $\mathcal{Q}_{safe}$.

III. Local Motion Planning

In this section, we develop local motion planners that we use with RRT* (§IV). In such planners, we use controller syntheses in which CLFs and higher order CBFs (HOCBFs) are utilized to generate control inputs to steer system (1) to a desired equilibrium state or to steer towards an exploratory sample while avoiding obstacles.

1) Control Lyapunov Functions: Consider steering the state of (1) to an equilibrium state $\boldsymbol{q}_{eq}$ (i.e. $f(\boldsymbol{q}_{eq}) = 0$).

Definition 3.1 (CLF [15]): Let $V(\boldsymbol{q}) : \mathcal{Q} \rightarrow \mathbb{R}$ be continuously differentiable function. $V(\boldsymbol{q})$ is said to be CLF if there exist $c_1, c_2, c_3 > 0$, such that

$$\begin{aligned} c_1||\boldsymbol{q} - \boldsymbol{q}_{eq}||^2 \leq V(\boldsymbol{q}) \leq c_2||\boldsymbol{q} - \boldsymbol{q}_{eq}||^2, \\ \dot{V}(\boldsymbol{q}) = \pounds_f V(\boldsymbol{q}) + \pounds_g V(\boldsymbol{q})\boldsymbol{u}, \\ \inf_{\boldsymbol{u} \in \mathcal{U}} [\dot{V}(\boldsymbol{q}) + c_3 V(\boldsymbol{q})] \leq 0, \quad \forall \boldsymbol{q} \in \mathcal{Q}. \end{aligned} \tag{2}$$

[1]We present the obstacles in the workspace directly as their image in $\mathcal{Q}$, i.e., robot's configurations that cause it to collide with obstacles.

where $\pounds_f V(\boldsymbol{q}) = \frac{\partial V(\boldsymbol{q})}{\partial \boldsymbol{q}} f(\boldsymbol{q})$, and $\pounds_g V(\boldsymbol{q}) = \frac{\partial V(\boldsymbol{q})}{\partial \boldsymbol{q}} g(\boldsymbol{q})$.

Theorem 3.1 ([11]): Let $V(\boldsymbol{q})$ be a CLF, then any Lipschitz continuous control input $\boldsymbol{u} \in \{\boldsymbol{u} \in \mathcal{U} | \pounds_f V(\boldsymbol{q}) + \pounds_g V(\boldsymbol{q})\boldsymbol{u} + c_3 V(\boldsymbol{q}) \leq 0\}$ asymptotically stabilizes (1) to $\boldsymbol{q}_{eq}$.

2) Higher Order Control Barrier Functions: Consider system (1) and a differentiable function $h(\boldsymbol{q}) : \mathcal{Q} \rightarrow \mathbb{R}$ with relative degree $\rho > 0$, where ρ reads as the number of times that we need to differentiate $h(\boldsymbol{q})$ until the control input $\boldsymbol{u}$ appears. Let a series of functions $\psi_j(.) : \mathcal{Q} \rightarrow \mathbb{R}$, $j = 0, 1, ..., \rho$, be defined as follows. $\psi_0 := h(\boldsymbol{q})$, and for $j \geq 1$, $\psi_j := \dot{\psi}_{j-1} + \alpha_j(\psi_{j-1})$, where $\alpha_j : \mathcal{Q} \rightarrow \mathbb{R}$ is a class $\mathcal{K}$ function [16].

Having defined ψ_j, let the sets $\mathcal{C}_j$, $j = 1, \ldots, \rho$, be defined by $\mathcal{C}_j := \{\boldsymbol{q} \in \mathcal{Q} \mid \psi_{j-1}(\boldsymbol{q}) \geq 0\}$.

Definition 3.2 (HOCBF [16]): Given $\psi_0, ...\psi_\rho$ with the corresponding series of sets $\mathcal{C}_1, ..., \mathcal{C}_\rho$, the differentiable function $h(\boldsymbol{q})$ is said to be HOCBF of relative degree ρ for system (1), if there are $\alpha_1, ..., \alpha_\rho$ class $\mathcal{K}$ functions such that $\forall \boldsymbol{q} \in \mathcal{C}_1 \cap \mathcal{C}_2 \cap ... \cap \mathcal{C}_\rho$,

$$\begin{aligned} \pounds_f^\rho h(\boldsymbol{q}) + \pounds_g \pounds_f^{\rho-1} h(\boldsymbol{q})\boldsymbol{u} + \frac{\partial^\rho h(\boldsymbol{q})}{\partial t^\rho} + O(h(\boldsymbol{q})) + \\ \alpha_\rho\left(\psi_{\rho-1}(\boldsymbol{q})\right) \geq 0 \end{aligned} \tag{3}$$

where $O(h(\boldsymbol{q}))$ is the partial derivatives with respect to t with relative degree $\leq \rho - 1$ and the remaining Lie derivatives along f [16].

Theorem 3.2 ([16]): Let $h(\boldsymbol{q})$ be a HOCBF, then any Lipschitz continuous control input $\boldsymbol{u}$, such that, $\boldsymbol{u} \in \{\boldsymbol{u} \in \mathcal{U} \mid \pounds_f^\rho h(\boldsymbol{q}) + \pounds_g \pounds_f^{\rho-1} h(\boldsymbol{q})\boldsymbol{u} + \frac{\partial^\rho h(\boldsymbol{q})}{\partial t^\rho} + O(h(\boldsymbol{q})) + \alpha_\rho\left(\psi_{\rho-1}(\boldsymbol{q})\right) \geq 0\}$, renders the set $\mathcal{C}_1 \cap \mathcal{C}_2 \cap ... \cap \mathcal{C}_\rho$ forward invariant in (1).

A. Formulation of Local Motion Planning

The work in [11] uses CLF-CBF-QP controller synthesis, where the control inputs are generated in a discrete-time manner. At each time step t, a control input is computed by solving a quadratic program (QP) subject to CLF and CBF constraints, and then applied for Δt time to evolve system (1). The CLF constraint certifies *liveness* of the trajectory, that is the trajectory is progressing towards a desired equilibrium state, $\boldsymbol{q}_f$. The CBF constraints certify the *safety* of the trajectory, which reads that $\mathcal{Q}_{safe}$ is forward invariant in system (1). Let $V(\boldsymbol{q}(t))$ be a CLF as defined in Definition 3.1 and $h(\boldsymbol{q}(t))$ be a HOCBF as defined in Definition 3.2, we reformulate the CLF-CBF-QP controller synthesis by constraining it with CLF and HOCBF constraints; the corresponding QP is given by:

$$\begin{aligned} \boldsymbol{u}_{LP}(t) = \underset{\boldsymbol{u}(t) \in \mathcal{U}}{\arg\min} \, ||\boldsymbol{u}(t) - \boldsymbol{u}_{ref}(t)||^2 + \delta^2 \\ \text{s.t.} \quad \pounds_f V(\boldsymbol{q}(t)) + \pounds_g V(\boldsymbol{q}(t))\boldsymbol{u}(t) + c_3 V(\boldsymbol{q}(t)) \leq \delta \\ \pounds_f^\rho h(\boldsymbol{q}(t)) + \pounds_g \pounds_f^{\rho-1} h(\boldsymbol{q}(t))\boldsymbol{u}(t) + \\ \frac{\partial^\rho h(\boldsymbol{q}(t))}{\partial t^\rho} + O(h(\boldsymbol{q}(t))) + \alpha_\rho\left(\psi_{\rho-1}(\boldsymbol{q}(t))\right) \geq 0 \end{aligned} \tag{4}$$

where δ is a slack variable to ensure the feasibility of the HOCBF constraint; $\boldsymbol{u}_{ref}(t)$ is a reference control input

which could be assigned if it is desirable to track reference control inputs while certifying safety and liveness; and $\boldsymbol{u}_{LP}(t)$ denotes the local planner control input at time t.

Remark 3.1 ([16]): CBF [11] is HOCBF with $\rho = 1$. In this paper, we use HOCBFs instead of CBFs to make the proposed algorithm amenable for planning for systems with relative degree $\rho \geq 1$.

At each iteration of kinodynamic RRT*, a uniform sample $\boldsymbol{q}_s \in \mathcal{Q}_{safe}$ is generated; the configuration of its nearest vertex is used as an initial condition in steering the system to a configuration in the direction of $\boldsymbol{q}_s$. If such configuration is not reachable, or the trajectory to reach it is in collision with an obstacle, the sampling iteration is rejected. In [10], [17], the reachable set of each vertex is approximated and is used to guide the expansion of the tree. In this work, however, by using a variant of the synthesis (4), all exploration samples are accepted, see §III-A.2. The asymptotic optimality (AO) of RRT* is ensured by rewiring the vertices of the exploratory tree. For this phase, we propose to use the synthesis (4) to generate control inputs that certify steering to exact desired configurations while certifying the safety of the system trajectory, see §III-A.1.

1) Exact Local Motion Planning: We set $\boldsymbol{u}_{ref} = 0$ in (4), and given $\boldsymbol{q}_{init}$ and $\boldsymbol{q}_f$, let $V(\boldsymbol{q})$ be a CLF with $\boldsymbol{q}_{eq} = \boldsymbol{q}_f$. For $\mathcal{Q}_{safe}$, assume that we are given a HOCBF $h(\boldsymbol{q})$. Using the this setting of (4), discrete control inputs are generated to steer from $\boldsymbol{q}_{init}$ to $\boldsymbol{q}_f$. In the planner implementation, we assign the QP (4) to be solved with at most $\mathtt{T}$ times to generate control inputs to steer (1) to $\boldsymbol{q}_f$. Given that the CLF constraint is relaxed with the slack variable δ, the planner might fail to steer to $\boldsymbol{q}_f$ and will stuck in local solution. In such scenario, the local motion plan will be disregarded. The time horizon of the produced trajectory is determined by the number of instances the QP is solved times Δt.

2) Exploratory Local Motion Planning: In this setting the goal is to steer form $\boldsymbol{q}_{init} \in \mathcal{Q}_{safe}$ to an exploratory configuration $\boldsymbol{q}_f \in \mathcal{Q}$. We use a relaxed variant of (4), denoted as CBF-QP, in which HOCBF are the only constraints. As it will become clear in shortly, the computed control inputs will generate safe trajectory to approach $\boldsymbol{q}_f$. Such steering helps in exploring $\mathcal{Q}$ while using RRT^*, see §IV.

Assumption 3.1: In the absence of obstacles (i.e., $\mathcal{Q}_{safe} = \mathcal{Q}$), assume that for any $\boldsymbol{q}_f \in \mathcal{Q}$ that is reachable from any configuration $\boldsymbol{q}_{init} \in \mathcal{Q}$, there is an avilable sequence of control inputs $\boldsymbol{u}_{OL}(t),\ t \in [0, T_{OL}]$ to steer system (1) from $\boldsymbol{q}(0) = \boldsymbol{q}_{init}$ to $\boldsymbol{q}(T_{OL}) = \boldsymbol{q}_f$.

In the following, we detail the setting of (4) to implement the CBF-QP. Based on Assumption 3.1, let the control inputs $\boldsymbol{u}_{OL}(t),\ t \in [0, T_{OL}]$ be computed offline in the absence of obstacles to steer to $\boldsymbol{q}_f$. The quadratic cost is set as $||\boldsymbol{u}(t) - \boldsymbol{u}_{OL}(t)||^2$. For $t \in [0, T_{OL}]$, $\boldsymbol{u}_{LP}(t)$ is computed by solving the aforementioned settings of QP (4) and is applied for Δt time duration to evolve system (1).

The utilities of using such exploratory and exact control inputs are: first, mitigate the conventional collision-checking procedure, which is computationally expensive, and second, any sample in $\mathcal{Q}$ is accepted for exploration, where the synthesis certify that the produced trajectory is in $\mathcal{Q}_{safe}$, thus, the number of samples that are used to yield an acceptable solution is reduced (see Fig. 2).

Example 3.1: Consider a unicycle robot with configuration $\boldsymbol{q} = [x,\ y,\ \theta]^\top \in \mathbb{R}^2 \times [-\pi, \pi]$, where $(x, y) \in \mathbb{R}^2$ and $\theta \in [-\pi, \pi]$ are the robot position and heading, respectively, with respect to the fixed frame $O - x_0y_0$ which is fixed at the origin. The elements of $\boldsymbol{q}$ evolve with respect to the following dynamics: $\dot{x} = v\cos(\theta),\ \dot{y} = v\sin(\theta),\ \dot{\theta} = \omega$, where $\omega \in [\underline{\omega}, \overline{\omega}],\ \underline{\omega}, \overline{\omega} \in \mathbb{R}$, and $v \in [\underline{v}, \overline{v}],\ \underline{v}, \overline{v} \in \mathbb{R}$ are the angular and translational velocity inputs with their corresponding upper and lower bounds, respectively. We assume the obstacles are modeled as circles or ellipsoids.

The HOCBF of ellipsoid i is defined using the following:

$$h_i(\boldsymbol{q}(t)) = [x(t) - x_i,\ y(t) - y_i]\, E \begin{bmatrix} x(t) - x_i \\ y(t) - y_i, \end{bmatrix} - 1 \quad (5)$$

where $(x_i, y_i) \in \mathbb{R}^2$ is the center of the obstacle with respect to $O - x_0y_0$; and the matrix E is given by

$$E = \begin{bmatrix} (\frac{\cos(\phi)}{\tilde{a}})^2 + (\frac{\sin(\phi)}{\tilde{b}})^2 & -\sin(\phi)\cos(\phi)\big((\frac{1}{\tilde{b}})^2 - (\frac{1}{\tilde{a}})^2\big) \\ -\sin(\phi)\cos(\phi)\big((\frac{1}{\tilde{b}})^2 - (\frac{1}{\tilde{a}})^2\big) & (\frac{\sin(\phi)}{\tilde{a}})^2 + (\frac{\cos(\phi)}{\tilde{b}})^2 \end{bmatrix}$$

with $\tilde{a} = a + r_r$ and $b = \tilde{b} + r_r$ being safety distances of the center of the robot along the major and minor axes, respectively; $a, b, r_r \in \mathbb{R}$ are the lengths of the major and minor axes of the ellipsoid, and the radius of the robot, respectively, and $\phi \in [-\pi, \pi]$ is the orientation of the obstacle with respect to $O - x_0y_0$. If $a = b$, then Eq. (5) degenerates to a circle.

Given an initial configuration (x_0, y_0, θ_0), we want to generate motion plans for the following two cases: **(i)** steering the robot to (x_d, y_d, θ_d) using the exact local motion planner (§III-A.1), and **(ii)** steering towards (x_d, y_d, θ_d) using the exploratory motion planner (§III-A.2).

Exact local motion planner formulation. Following the approach in [18], we consider controlling a look-ahead point that is d distance from the center of the wheels axis and along the sagittal axis of unicycle robot. The dynamics of a look-ahead point, $(x_l, y_l) \in \mathbb{R}^2$ is given by the integrator dynamics,

$$\begin{bmatrix} \dot{x}_l \\ \dot{y}_l \end{bmatrix} = \begin{bmatrix} u_1 \\ u_2 \end{bmatrix} = \begin{bmatrix} \cos\theta & -d\sin\theta \\ \sin\theta & d\cos\theta \end{bmatrix} \begin{bmatrix} v \\ \omega \end{bmatrix}. \quad (6)$$

where $u_1, u_2 \in \mathbb{R}$. Let $V := ||[x_l - (x_d + d\cos\theta),\ y_l - (y_d + d\sin\theta)]^\top||^2$ be a CLF with $\boldsymbol{q}_{eq} = (x_d + d\cos\theta), y_d + d\sin\theta))$. For each obstacle we define a HOCBF (5) while substituting the look-ahead state variables $x_l(t)$ and $y_l(t)$ instead of $x(t)$ and $y(t)$, respectively. The CLF and HOCBF are both with relative degree $\rho = 1$ with respect to the control $\boldsymbol{u} = [u_1, u_2]^\top$. We compute the HOCBF constraint using inequality (3) where $\psi_0(\boldsymbol{q}) = h(\boldsymbol{q})$ and we assign $\alpha_1(\psi_0(\boldsymbol{q})) = h(\boldsymbol{q})$; in the CLF constraint in (4) c_3 is set to 1. The control inputs are computed by solving the QP (4) at each time step, then they could be mapped to the linear and angular velocities v, u via the static map

$$\begin{bmatrix} v \\ \omega \end{bmatrix} = \begin{bmatrix} \cos\theta & -d\sin\theta \\ \sin\theta & d\cos\theta \end{bmatrix}^{-1} \begin{bmatrix} u_1 \\ u_2 \end{bmatrix}, \quad (7)$$

where the matrix in (7) is always invertible unless $d = 0$. In Fig. 1.**c** we show the generated trajectory of the look-ahead state using the exact local motion planner control inputs.

Exploratory local motion planner formulation. In the exploratory CBF-QP (see §III-A.2) we compute $\boldsymbol{u}_{OL}$ as follows. $(x_{l,0}, y_{l,0}) = (x_0 + d\cos\theta_0, y_0 + d\sin\theta_0)$ and $(x_{l,d}, y_{l,d}) = (x_d + d\cos\theta_d, y_d + d\sin\theta_d)$ are the initial and desired configurations of the look ahead point, respectively, given the integrator dynamics (6) we define $\boldsymbol{u}_{OL}$ as piecewise linear controls that represents the line between $(x_{l,0}, y_{l,0})$ and $(x_{l,d}, y_{l,d})$. In Fig. 1.**b** we show the produced trajectory of the look-ahead state using the exploratory local motion planner control input, where the trajectory is deviated from following $\boldsymbol{u}_{OL}$ due to the presence of obstacles.

IV. CBF-RRT*

In this section we detail the formulation of the proposed algorithm, CBF-RRT*, which approximates a solution of the OMPP 2.1. The exploratory and exact local motion planners (see §III-A.2, and §III-A.1) are used to expand the RRT tree and to rewire the tree, respectively. We show that, under some assumptions, the probabilistic completeness of RRT* is preserved using such local motion planning.

A. The Algorithm

Considering tree $\mathcal{T} = (\mathcal{V}, \mathcal{E})$ on $\mathcal{Q}_{safe}$, with vertices set $\mathcal{V} \subset \mathcal{Q}_{safe}$ and edges $\mathcal{E} = \mathcal{V} \times \mathcal{V}$, we define the following primitive functions that are used in the proposed work: **(i)** $\texttt{Sample}(\mathcal{G}, \texttt{adapFlag}) : \mathcal{G} \times \{\texttt{True}, \texttt{False}\} \to \mathcal{Q}$,, given a set of $\varphi \in \mathcal{G}$ and adapFlag, the function returns a sample in $\mathcal{Q}$. If $\texttt{adapFlag} = \texttt{False}$, the function returns a uniform sample from $\mathcal{Q}$, otherwise the SDF will be adapted (see §V) and will be used to generate a sample in $\mathcal{Q}$. **(ii)** $\texttt{Comp\_uOL}(\boldsymbol{q}_s, v) : \mathcal{Q}_{safe} \times \mathcal{V} \to \mathcal{U}$, given sample $\boldsymbol{q}_s$, vertex v and Assumption 3.1, the function computes the control inputs $\boldsymbol{u}_{OL}(t) \in \mathcal{U}, \ t \in [0, T_{OL}]$ and a time horizon T_{OL} to steer from vertex v towards $\boldsymbol{q}_s$. **(iii)** $\texttt{ExpLPlng}(v, \boldsymbol{u}_{OL}, T_{OL}) : \mathcal{V} \times \mathcal{U} \times \mathbb{R}_{>0} \to \mathcal{V}$, given vertex v and control inputs $\boldsymbol{u}_{OL}$, the function steers system (1) form v using the exploratory CBF-QP local motion planner (see §III-A.2) with $\boldsymbol{u}_{ref} = \boldsymbol{u}_{OL}$, and then establishes a vertex, v_{new}, at the last configuration of the produced trajectory, which, as detailed in §III, is certified to be in $\mathcal{Q}_{safe}$. **(iv)** $\texttt{ExtLPlng}(v_1, v_2) : \mathcal{V} \times \mathcal{V} \to \mathcal{V}$, given v_1 and v_2, the function steers from v_1 to v_2 using the exact CLF-CBF-QP local motion planner (see §III-A.1).

CBF-RRT* is initialized with a root vertex, v_{start}, at $\boldsymbol{q}_{start}$ (Line 2 in Alg. 1). Exploration is done by sampling $\boldsymbol{q}_s \in \mathcal{Q}$, which is used guide the expansion of its nearest vertex, $v_{nearest}$ (Line 6 - Line 8). We compute $\boldsymbol{u}_{OL}(t), t \in [0, T_{OL}]$ that, if $||\boldsymbol{q}_s - v_{nearest} < \eta||$, steers (1) from $v_{nearest}$ to $\boldsymbol{q}_s$, otherwise, steers (1) to $\boldsymbol{q}_{new}$ (which is toward $\boldsymbol{q}_s$) such that $||\boldsymbol{q}_{new} - v_{nearest} = \eta||$, where $\eta \in \mathbb{R}_{R>0}$(Line 7). $\eta := \frac{\varepsilon}{4} + \mu + 2\iota$, where ε is a parameter imposed by the robot environment, μ is a parameter measured by tuning the HOCBF, and $0 < \iota < \frac{\varepsilon}{4} - \mu$ (see the completeness analysis §IV-B in [19]). The computed $\boldsymbol{u}_{OL}$ is used with the exploratory local motion planner ExpLPlng to extend to v_{new} (Line 8).

Algorithm 1: Adap-CBF-RRT*

1 **Input:** $\boldsymbol{q}_{start}$; $\mathcal{Q}_{goal}$; $\mathcal{Q}_{safe}$; N, e_l, **and** Δt
2 **Initialization:** $v_{start} = (\boldsymbol{q}_{init}, \texttt{index} = 0)$, $i = 1$, $\mathcal{V} = \{v_{init}\}$, $\mathcal{E} = \emptyset$, $\mathcal{G} = \emptyset$, $\texttt{GoalReached} = \texttt{False}$, $\texttt{adapFlag} = \texttt{True}$, $\texttt{optSDFflag} = \texttt{False}$, **and** $r = \eta$
3 $\mathcal{T} \leftarrow (\mathcal{V}, \mathcal{E})$
4 **while** $i < N$ **do**
5 $\quad \boldsymbol{q}_s \leftarrow \texttt{Sample}(\mathcal{G}, \texttt{adapFlag})$
6 $\quad v_{nearest} \leftarrow \texttt{Nearest}(\boldsymbol{q}_s)$
7 $\quad \boldsymbol{u}_{OL} = \texttt{Comp\_uOL}(\boldsymbol{q}_s, v_{nearest})$
8 $\quad \mathcal{V} \leftarrow \mathcal{V} \cup \{v_{new} \leftarrow \texttt{ExpLPlng}(v_{nearest}, \boldsymbol{u}_{OL})\}$
9 $\quad r = \min\{\lambda(\log(|\mathcal{V}|)/|\mathcal{V}|)^{1/(d+1)}, \eta\}$
10 $\quad \mathcal{V}_{near} \leftarrow \texttt{Near}(\mathcal{T}, r, v_{new})$
11 $\quad$ **foreach** $v_{near} \in \mathcal{V}_{near}$ **do**
12 $\quad\quad v' \leftarrow \texttt{ExtLPlng}(v_{near}, v_{new})$
13 $\quad\quad c' = v_{near}.\texttt{cToCome} + \texttt{Cost}(v', v_{near})$
14 $\quad\quad$ **if** $c' < c_{min}$ **then**
15 $\quad\quad\quad v'_{new} \leftarrow v'$; $v_{min} \leftarrow v_{near}$; $c_{min} \leftarrow c'$
16 $\quad \mathcal{T} \leftarrow \texttt{AddChild}(\mathcal{T}, v_{min}, v_{new})$
17 $\quad$ **foreach** $v_{near} \in \mathcal{V}_{near}$ **do**
18 $\quad\quad v' \leftarrow \texttt{ExtLPlng}(v_{new}, v_{near})$
19 $\quad\quad$ **if** $(v_{new}.cToCome + Cost(v_{new}, v') < v_{near}.cToCome)$ **then**
20 $\quad\quad\quad \mathcal{T} \leftarrow \texttt{Reconnect}(v_{new}, v_{near}, \mathcal{T})$
21 $\quad\quad\quad \texttt{UpcToCome}(v_{near}, \texttt{cToCome}(v_{new} + Cost(v')))$
22 $\quad \mathcal{T}, \mathcal{G} \leftarrow \texttt{extToGoal}(\mathcal{T}, v_{new}, \texttt{adapFlag})$; $i \leftarrow i + 1$
23 **return** $\mathcal{T}$

The ideal case of the exploration phase is to steer to $\boldsymbol{q}_{new}$ such that $||v_{nearest} - \boldsymbol{q}_{new}|| = \eta$ and in the direction of sample $\boldsymbol{q}_s$, however, if $\boldsymbol{q}_{new}$ lies within or close to an obstacle, the produced trajectory will deviate from reaching the desired configuration. Such deviation, however, is acceptable under some assumptions to preserve the completeness of the algorithm, see Theorem 4.1. Moreover, since the trajectories are guaranteed to be safe, no collision check is needed, which reduces the computational burden of the algorithm.

The rewiring procedure (Line 11 - Line 21) is similar to RRT* [4]. Rewiring $v_1 \in \mathcal{V}$ to a vertex that is reachable from, $v_2 \in \mathcal{V}$, is accomplished through the exact local motion planner (§III-A.1), see Line 12 and Line 18.

Remark 4.1: Theorem 1 in [5] concludes that the AO of a path generated using Kinodynamic RRT*, is guaranteed if, for $v \in \mathcal{V}$, the vertices that lie within a d-dimensional hypersphere of radius $\lambda(\log(|\mathcal{V}|)/|\mathcal{V}|)^{1/(d+1)}$ are considered for searching for better parent vertex for v, where $\lambda \in \mathbb{R}_{>0}$ and $|\mathcal{V}|$ is the number of vertices of tree $\mathcal{T}$. The rewiring procedure of CBF-RRT* is assigned as such (Line 9 in Alg. 1), however, having used local motion planners based on CBF-based synthesis, the AO of CBF-RRT* still requires further investigation.

B. Probabilistic Completeness of the Algorithm

In the following, the probabilistic completeness of CBF-RRT* is established.

First, We formulate some definitions that are needed for the analysis. For any $\varphi \in \mathcal{G}$, let $\varphi_{\boldsymbol{q}} := \{\mathbf{proj}_{\mathcal{Q}}(\varphi(t,T)) | \varphi \in \mathcal{G}, \ t \in [0,T]\}$ and $\varphi_{\boldsymbol{u}} := \{\mathbf{proj}_{\mathcal{U}}(\varphi(t,T)) | \varphi \in \mathcal{G}, \ t \in [0,T]\}$, where $\mathbf{proj}_{\mathcal{U}} : \mathcal{U} \times \mathcal{Q} \to \mathcal{U}$ and $\mathbf{proj}_{\mathcal{Q}} : \mathcal{U} \times \mathcal{Q} \to \mathcal{Q}$ are the projection of the control inputs and the produced

Algorithm 2: $\boldsymbol{q}_s \leftarrow \texttt{Sample}(\mathcal{G}, \mathcal{T}, m)$

```
1  u ~ Uniform(0, 1)
2  if u ≤ 0.5 ∧ G ≠ ∅ then
3      if optSDFflag then
4          X ~ ĝ*(q) return (q)
5      else
6          if mod(|G|, n_v) = 0 then
7              E ← Quantile(G, ϱ)                    ▷ Assign the elite set
8              ĝ(q) ← CE_Estimate (E, m)             ▷ Compute PDF of E
9              return q ~ ĝ(q)
10         else
11             return q ~ Uniform(Q)
12 else
13     return q ~ Uniform(Q)
```

trajectory of φ, respectively, i.e., $\mathbf{proj}_{\mathcal{U}}(\varphi(t,T)) = \boldsymbol{u}(t)$ and $\mathbf{proj}_{\mathcal{Q}}(\varphi(t,T)) = \boldsymbol{q}(t)$.

Following [4], we say that OMPP 2.1 is robustly feasible with minimum clearance $\varepsilon > 0$, if there exist control inputs $\varphi_{\boldsymbol{u}}$ which produce trajectory $\varphi_{\boldsymbol{q}}$, and $\varphi \in \mathcal{G}$, such that the distance between any configuration $\boldsymbol{q} \in \varphi_{\boldsymbol{q}}$ and any obstacle configuration $\boldsymbol{q}_o \in \mathcal{Q} \setminus \mathcal{Q}_{safe}$ is at least $\varepsilon/2$.

Theorem 4.1: CBF-RRT* is probabilistically complete.

Proof: See [19], in which we detail the proof. ∎

V. Adaptive Sampling for CBF-RRT*

We leverage CBF-RRT* with an adaptive sampling procedure, in which we use CEM to focus sampling around the optimal trajectory $\varphi^*_{\boldsymbol{q}}$. The motivation behind this approach is to approximate the solution of the OMPP 2.1 with a fewer number of samples by focusing the sampling in promising regions of $\boldsymbol{Q}_{safe}$.

A. Adaptive Sampling using the cross-entropy Method

CEM [8] has been used to estimate the probability of rare events using IS. CEM is a multi-stage stochastic optimization algorithm that iterates upon two steps: first, it generates samples from a current SDF and computes the cost of each sample; second, it chooses an *elite subset* of the generated samples for which their cost is below some threshold; finally, the elite subset is used to estimate a probability density function (PDF) as if they were drown as i.i.d samples. The estimated PDF will be used as the SDF for the next iteration. The algorithm terminate when it converges to a limiting PDF. It has been proven in [20] that CEM with parametric SDF converges to a limiting distribution.

Going into more technical details, let a random variable $Q : \Omega \to \mathcal{Q}$ be defined over the probability space $(\Omega, \mathcal{F}, P)$, where Ω is the sample space, $\mathcal{Q}$ is the range space, $\mathcal{F}$ denotes the σ-algebra subset of $\mathcal{Q}$, and P is the probability measure over $\mathcal{F}$. CEM aims to find rare events with probability $P(\mathcal{J}(\boldsymbol{q}) \leq \gamma)$, where $\gamma \in \mathbb{R}_{>0}$ is the optimal cost and $\mathcal{J} : \mathcal{Q} \to \mathbb{R}_{>0}$ is the cost of a sampled solution $\boldsymbol{q}$. Computing $P(\mathcal{J}(\boldsymbol{q}) \leq \gamma)$ is equivalent to find the expectation $E[I(\{\mathcal{J}(\boldsymbol{q}) < \gamma\})]$, where $I(.)$ is the indicator function.

The work in [8] proposes to evaluate the expectation $E[I(\{\mathcal{J}(\boldsymbol{q}) < \gamma\})]$ using the following estimator: $\hat{\ell} = \frac{1}{N}\sum_{i=1}^{N} I(\{\mathcal{J}(\boldsymbol{q}) < \gamma\})\frac{\mathfrak{f}(\boldsymbol{q}_i)}{\mathfrak{g}(\boldsymbol{q}_i)}$, where $\mathfrak{f}(\boldsymbol{q}_i)$ is the PDF of a sampled solution, $\boldsymbol{q}_i$, and $\mathfrak{g}(\boldsymbol{q}_i)$ is an underlying IS PDF. Choosing $\mathfrak{g}^*(\boldsymbol{q}) = I(\{\mathcal{J}(\boldsymbol{q}) < \gamma\})\mathfrak{f}(\boldsymbol{q})/\hat{\ell}$ will yield the best estimate of $\hat{\ell}$. However, this solution is hypothetical, since it involves $\hat{\ell}$. Instead, $\mathfrak{g}^*(\boldsymbol{q})$ is computed in a multi-stage manner, where at each stage the elite subset is used to estimate $\mathfrak{g}(\boldsymbol{q})$ until the CE between $\mathfrak{g}(\boldsymbol{q})^*$ and $\mathfrak{g}(\boldsymbol{q})$ is minimized. The CE is related to the Kullback-Leibler divergence, $D_{KL}(\mathfrak{g}^*(\boldsymbol{q})||\mathfrak{g}(\boldsymbol{q})) = \int_{\mathcal{Q}} \mathfrak{g}^*(\boldsymbol{q}) \ln(\mathfrak{g}^*(\boldsymbol{q})/\mathfrak{g}(\boldsymbol{q}))\ d\boldsymbol{q}$, and minimizing it implies minimizing the CE.

B. Adap-CBF-RRT*

In the context of CBF-RRT*, one could ask the following question: what is the probability of sampling configurations that lie on the OMPP solution $\varphi^*_{\boldsymbol{q}}$? It can be easily seen that it is an extremely small probability. Kobalirov [7] proposes to use CEM to estimate the probability of generating samples that lie on $\varphi^*_{\boldsymbol{q}}$ using a mixture of Gaussian models (GMM) for the proposal distribution $\mathfrak{g}(\boldsymbol{q})$. However, it is hard to know, prior to planning, how promising regions of $\mathcal{Q}$ are distributed in order to choose a suitable number of GMM.

The challenge above has motivated us to use a non-parametric density estimate, namely the weighted *Gaussian Kernel Density Estimate* (WGKDE), instead of a GMM.

To improve the SDF of CBF-RRT* using CEM, we need to generate a population of approximated solutions of the OMPP 2.1. `extToGoal` procedure (Line 22 in Alg. 1) attempts to steer system (1) from v_{new} to $\boldsymbol{q}_g \in \mathcal{Q}_{goal}$ using `ExpLPlng`. If the final configuration of the produced trajectory lies in $\mathcal{Q}_{goal}$, a vertex, v_g, at that configuration, is created and added to $\mathcal{V}$. Accordingly, the generated control inputs and system trajectory, φ, is added to $\mathcal{G}$.

Adapting the SDF of CBF-RRT* is detailed in Alg. 2. Consider an iteration of CBF-RRT* (Alg. 1) with $\mathcal{G} \neq \emptyset$, the elite set, $\mathfrak{E}$, is assigned by choosing the trajectories of all $\varphi \in \mathcal{G}$ with $J(\varphi) \leq \gamma$, i.e $\mathfrak{E} = \{\varphi_{\boldsymbol{q}} \,|\, (\varphi_{\boldsymbol{u}}, \varphi_{\boldsymbol{q}}) = \varphi \in \mathcal{G};\ J(\varphi) \leq \gamma\}$. We pick γ as the ϱ^{th} percentile cost of $\varphi \in \mathcal{G}$; it is advised to assign $\varrho \in [0.01, 0.1]$ [8]. Since the SDF samples in $\mathcal{Q}$, we will use a sparse set of configurations of the elite trajectories ($\mathfrak{E}$) to estimate a PDF that will be used as an SDF for the next iteration. Let `d_elite` be a set of pairs of e configurations of each $\varphi_{\boldsymbol{q}} \in \mathfrak{E}$ with cost of the corresponding elite trajectory, i.e., $\texttt{d\_elite}(\mathfrak{E}, e) = \{(\boldsymbol{q}_i, \mathcal{J}(\boldsymbol{q}_i))|i \in \{1, \ldots, m\}, \boldsymbol{q}_i \in \varphi_{\boldsymbol{q}},\ \forall \varphi_{\boldsymbol{q}} \in \mathfrak{E}, \mathcal{J}(\boldsymbol{q}_i) = J(\varphi)\}$. The WGKDE of the discretized elite trajectories is computed by: $\hat{\mathfrak{g}}(\boldsymbol{q}) = \sum_{(\boldsymbol{q}_i, \mathcal{J}(\boldsymbol{q}_i)) \in \texttt{d\_elite}(\mathfrak{E}, m)} \tilde{w}_i\ K(\boldsymbol{q})$, where the normalized weight $\tilde{w}_i$ and the GK function $K(\boldsymbol{q})$ are computed, respectively, by: $\tilde{w}_i = 1 - \frac{\mathcal{J}(\boldsymbol{q}_i)}{\sum_{(\boldsymbol{q}_j, \mathcal{J}(\boldsymbol{q}_j)) \in \texttt{d\_elite}(\mathfrak{E}, m)} (\mathcal{J}(\boldsymbol{q}_j))}$ and $K_i(\boldsymbol{q}) = \frac{1}{\sqrt{2\,\pi}\sigma}\, exp\Big(\frac{-||\boldsymbol{q}-\boldsymbol{q}_i||^2}{2\sigma^2}\Big)$. The procedure `CE_Estimate` $(\mathfrak{E}, m)$ (Line 8 in Alg. 2) performs the WGKDE from the elite trajectories and checks if the D_{KL} between the current estimate and the previous estimate is bellow a certain threshold and update `optSDFflag` accordingly. Finally, the algorithm converges to a limiting PDF (where in this case `optSDFflag` is set to `True`).

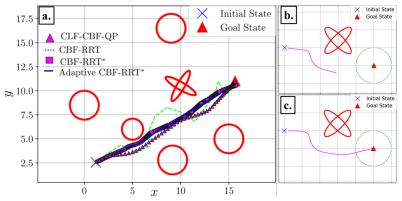

Fig. 1. **(a)** Trajectories generated using CBF-RRT (green path), CBF-RRT* (magenta path), Adap-CBF-RRT* (blue path), and CLF-CBF-QP (magenta triangles path); **(b)** trajectories generated using CBF-QP exploratory motion planner, and **(c)** trajectories CBF-CLF-QP exact motion planner.

VI. Simulation Example

We consider generating motion plans using CBF-RRT, RRT*, CBF-RRT*, Adap-CBF-RRT*, and the CLF-CBF-QP-based exact motion planner for the unicycle drive robot of Example 3.1. The generated paths are depicted in Fig. 1, where the Adap-CBF-RRT* (shown in solid blue path) appears to be the smoothest path because the algorithm keeps the extensions of the vertices to the goal as part of the tree. Even though keeping such extensions requires additional memory, they help to produce acceptable paths with fewer vertices, see Fig. 2.

Fig. 2 shows the evolution of the path length with respect to the number of vertices. For 20 runs of Adap-CBF-RRT* the algorithm needed, on average, 392 vertices to converge to a limiting SDF, which leads to more efficient refinement of the path, see Fig. 2. On the other hand, the other algorithms were able to find a path after the 200^{th} vertex.

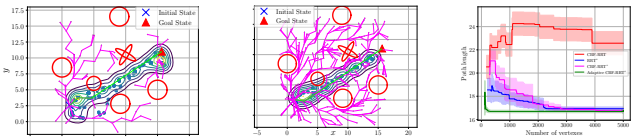

Fig. 2. Adap-CBF-RRT* tree and the corresponding $\varphi_{\boldsymbol{q}}$ (shown in green), at the 1^{st} (left fig.) and 4^{th} (middle fig.) iterations of adapting the SDF (depicted using level sets), respectively. The average path length (right fig.) of 20 runs of RRT*, CBF-RRT, CBF-RRT*, and Adap-CBF-RRT*.

VII. Conclusion and Future Work

Two variants of RRT*, (Adaptive) CBF-RRT*, are introduced to approximate a solution for the optimal motion planning problem. Inspired by CBF-RRT [12], we utilized the recent advances in controlling safety-critical systems via CBFs to generate feasible motion plans that are guaranteed to be collision-free. We prove, that CBF-RRT* is probabilistically complete. Furthermore, and for efficient exploration, we equip CBF-RRT* with an IS procedure, which is inspired by CE-RRT* [7], and uses CEM algorithm with WGKDE to estimate IS density functions. The procedure adapts the SDF of CBF-RRT* to focus the sampling around the optimal path. The proposed variants are demonstrated through numerical simulation, and they have been shown to outperform analogous algorithms in terms of number of iterations. In future work, we consider using more efficient ways of utilizing CBF-based local planner, and we consider planning for robots other than unicycles.

References

[1] S. M. LaValle, "Rapidly-exploring random trees: A new tool for path planning," in *Ames, IA, USA*, 1998.

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

[3] 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.

[4] 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.

[5] 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)*, 2020, pp. 2189–2195.

[6] 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*, 2014, pp. 2997–3004.

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

[8] R. Rubinstein, "The Cross-Entropy Method for Combinatorial and Continuous Optimization," *Methodology And Computing In Applied Probability*, vol. 1, no. 2, pp. 127–190, 1999.

[9] S. Karaman and E. Frazzoli, "Optimal kinodynamic motion planning using incremental sampling-based methods," *Proceedings of the IEEE Conference on Decision and Control*, pp. 7681–7687, 2010.

[10] A. Wu, S. Sadraddini, and R. Tedrake, "R3T: Rapidly-exploring Random Reachable Set Tree for Optimal Kinodynamic Planning of Nonlinear Hybrid Systems," *IEEE International Conference on Robotics and Automation (ICRA)*, pp. 4245–4251, 2020.

[11] A. D. Ames, S. Coogan, M. Egerstedt, G. Notomista, K. Sreenath, and P. Tabuada, "Control barrier functions: Theory and applications," *arXiv*, pp. 3420–3431, 2019.

[12] 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.

[13] K. Majd, S. Yaghoubi, T. Yamaguchi, B. Hoxha, D. Prokhorov, and G. Fainekos, "Safe navigation in human occupied environments using sampling and control barrier functions," in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2021, pp. 5794–5800.

[14] J. H. Reif, "Complexity of the mover's problem and generalizations," *20th Annual Symposium on Foundations of Computer Science (sfcs 1979)*, pp. 421–427, 1979.

[15] P. Wieland and F. Allgöwer, "Constructive safety using control barrier functions," *IFAC Proceedings Volumes (IFAC-PapersOnline)*, vol. 7, no. PART 1, pp. 462–467, 2007.

[16] W. Xiao and C. Belta, "Control Barrier Functions for Systems with High Relative Degree," *Proceedings of the IEEE Conference on Decision and Control*, pp. 474–479, 2019.

[17] 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)*, 2017, pp. 1625–1630.

[18] S. Wilson, P. Glotfelter, L. Wang, S. Mayya, G. Notomista, M. Mote, and M. Egerstedt, "The robotarium: Globally impactful opportunities, challenges, and lessons learned in remote-access, distributed control of multirobot systems," *IEEE Control Systems Magazine*, vol. 40, no. 1, pp. 26–44, 2020.

[19] A. Ahmad, C. Belta, and R. Tron, "Adaptive sampling-based motion planning with control barrier functions," 2022. [Online]. Available: https://arxiv.org/abs/2206.00795

[20] J. Hu, M. C. Fu, and S. I. Marcus, "A model reference adaptive search method for global optimization," *Operations Research*, vol. 55, no. 3, pp. 549–568, 2007.