Safety Embedded Differential Dynamic Programming Using Discrete Barrier States

Hassan Almubarak, *Graduate Student Member, IEEE*, Kyle Stachowicz, Nader Sadegh, *Member, IEEE*, and Evangelos A. Theodorou, *Member, IEEE*

***Abstract*—Certified safe control is a growing challenge in robotics, especially when performance and safety objectives must be concurrently achieved. In this work, we extend the barrier state (BaS) concept, recently proposed for safe stabilization of continuous time systems, to safety embedded trajectory optimization for discrete time systems using discrete barrier states (DBaS). The constructed DBaS is embedded into the discrete model of the safety-critical system integrating safety objectives into the system's dynamics and performance objectives. Thereby, the control policy is directly supplied by safety-critical information through the b rrier state. This allows us to employ the DBaS with differential dynamic programming (DDP) to plan and execute safe optimal trajectories. The proposed algorithm is leveraged on various safety-critical control and planning problems including a differential wheeled robot safe navigation in randomized and complex environments and on a quadrotor to safely perform reaching and tracking tasks. The DBaS-based DDP (DBaS-DDP) is shown to consistently outperform penalty methods commonly used to approximate constrained DDP problems as well as CBF based safety filters.**

***Index Terms*—Constrained motion planning, optimization and optimal control, robot safety.**

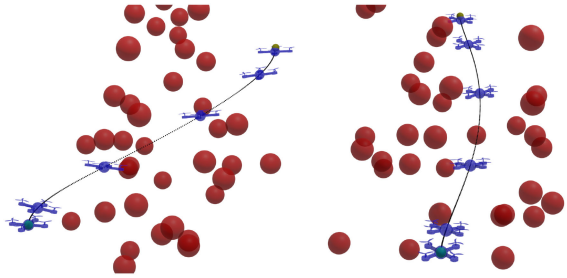

Fig. 1. Two views of the quadrotor reaching problem with many spherical obstacles in the space. The proposed DBaS-DDP safely performs the reaching task starting from the initial position (green) to the final position (yellow).

I. Introduction

SAFETY in robotics, in its various forms - including collision avoidance, safe collaboration, etc. - is crucial to expanding the applicability of autonomous robots. With increasing demand for autonomy in various industries, this task is increasingly daunting even for known environments. Therefore, there is a clear need for provably safe controls. Yet, the difficulty in unifying safety and performance objectives usually calls for the trade-off between the objectives. To confront such a trade-off, this letter develops a technique to enforce safety in optimization-based controllers for discrete time nonlinear systems that guarantees safety as long as a solution exists. The letter builds on a recently proposed safety integrating technique for stabilization of continuous time systems [1], which enforces safety through embedding barrier states (BaS) into the model of the dynamical system. We extend the idea to trajectory optimization for discrete time nonlinear systems by developing a novel extension we term discrete barrier states (DBaS).

Safety, which can be verified through set invariance [2], is an increasingly important property of dynamical systems. The development of barrier certificates [3], [4] formed an early approach to verification. Later, inspired by control Lyapunov functions and barrier certificates, Wieland and Allgöwer [5] introduced control barrier functions (CBFs) to propose a feedback method of enforcing safety in continuous time systems. In an attempt to develop safe stabilization, Ames *et al.* [6] and Romdlony and Jayawardhana [7] proposed spiritually similar, albeit distinct, CLF-CBF unification techniques. Ames *et al.* [6] pioneered the CLF-CBF quadratic program (QP) paradigm which was further developed in [8]. The CLF-CBF QP and the developed CBF have attracted researchers attention to be adopted in various control frameworks and robotic applications [9]–[12]. For discrete time systems, Agrawal and Sreenath [9] extended the notion of continuous time CBFs and CLF-CBF QPs to problems in discrete time. Nonetheless, discrete CBFs, which use reciprocal barrier functions, tend to be more restrictive than their continuous counterparts as the optimization problem

Manuscript received August 8, 2021; accepted December 27, 2021. Date of publication January 14, 2022; date of current version February 1, 2022. The work of E. A. Theodorou was supported by the National Science Foundation, CPS under Grant 1932288. This letter was recommended for publication by Associate Editor Alan Kuntz and Editor Hanna Kurniawati upon evaluation of the reviewers' comments. *(Corresponding author: Hassan Almubarak.)*

Hassan Almubarak is with the School of Electrical and Computer Engineering, Georgia Institute of Technology, Atlanta, GA 30332 USA, and also with the Control and Instrumentation Engineering Department, King Fahd University of Petroleum and Minerals, Dhahran 31261, Saudi Arabia (e-mail: halmubarak@gatech.edu).

Kyle Stachowicz is with the School of Computer Science, Georgia Institute of Technology, Atlanta, GA 30332 USA (e-mail: kwstach@gatech.edu).

Nader Sadegh is with the George W. Woodruff School of Mechanical Engineering, Georgia Institute of Technology, Atlanta, GA USA (e-mail: sadegh@gatech.edu).

Evangelos A. Theodorou is with the Department of Aerospace Engineering, Georgia Institute of Technology, Atlanta, GA 30332 USA (e-mail: evangelos.theodorou@gatech.edu).

This letter has supplementary downloadable material available at https://doi.org/10.1109/LRA.2022.3143301, provided by the authors.

Digital Object Identifier 10.1109/LRA.2022.3143301

may not be quadratic and is non-convex, which limits its applicability [9]. Therefore, they proposed discrete time exponential barrier function (DECBFs) that solves the problem, which was generalized by Ahmadi *et al.* [13] and called discrete zeroing CBFs (DZCBFs), in analogy to its continuous time counterpart ZCBFs in [8].

To use CBFs in multi-objective controls, one must trade off between safety and performance objectives [8]. Moreover, in the case of high relative-degree constraints, the problem becomes more challenging. Several methods have been developed to form higher-order CBFs, the most popular of these involving defining a new invariant safe set as the intersection of several invariant sets [14], [15]. However, these approaches tend to unnecessarily restrict the allowable state space and are difficult to implement and tune. To avoid finding a high-order CBF altogether, some practical implementations [16]–[18] instead apply an ad-hoc solution by modifying the CBF or the safety constraint to ensure a relative degree of 1. However, this is likely to affect performance or safety of the resulting system with respect to the original objective and safety constraint.

Typically, when multiple safety constraints must be satisfied, multiple corresponding CBF inequality constraints are used [16]–[18] and thus further relaxations could be needed. Wang *et al.* [19] proposed compositional barrier functions which can combine multiple CBFs into a single barrier function. However, this technique does not easily generalize, for example to learned or robust CBFs [17], and may create a CBF of ill-conditioned or high relative degree in some cases.

Constrained trajectory optimization is a challenging problem that has been revisited repeatedly in the literature. The differential dynamic programming (DDP) method can efficiently find optimal trajectories, but it is not straightforward to implement constraints. Contrarily, direct methods based on general nonlinear solvers can directly incorporate constraints, but at the expense of computational efficiency. Murray and Yakowitz [20] describes an early approach to incorporating constraints into DDP, but only considers control constraints. This was later improved using active-set QP methods [21], but the state-constrained case has remained a difficult open problem. One common approach is an application of the Augmented Lagrangian [22] [23] technique, which iteratively finds Lagrange multipliers for the constraint with first-order convergence. Xie *et al.* [24] proposed an active-set approach to the constrained DDP problem, which calculates active-set conditions in the backwards pass and solves a QP at each stage of the forwards pass. Aoyama *et al.* [25] presented a related algorithm that switches online between an Augmented Lagrangian and an active-set method for faster global convergence. Finally, interior-point methods have been applied in Pavlov *et al.* [26] to achieve local second-order convergence in the presence of nonlinear constraints. However, these algorithms often have difficulty with highly locally-nonlinear constraints and require substantial tuning and good-quality warm-starts to achieve satisfactory results.

A. Contributions and Organization

In this letter, we state the safety constraint formulation in Section II. After that, we develop discrete barrier states (DBaS) to enforce safety for nonlinear discrete time systems in Section III. Thereafter, a DBaS is embedded in the system's model forcing the control search to take place in the set of safe controls, which avoids compromising the performance or safety objectives. In addition, we show how to represent multiple constraints using a single DBaS. Section IV states the constrained optimal control problem statement. Subsequently, we leverage the safety embedding technique with differential dynamic programming (DDP) to develop safe trajectory optimization. We show that the generated trajectories are guaranteed to be safe as long as the standard DDP convergence conditions are satisfied. We show the generality of our proposed framework in Section V by applying DBaS-DDP to several systems including collision-avoidance problems for omnidirectional and differential wheeled robots, a cart-pole problem where motion is bounded by a fixed-length rail, and a variety of quadrotor tasks including safe trajectory tracking and reaching as in Fig. 1. We compare DBaS-DDP with the penalty method DDP and with CBF when possible, and demonstrate that it exhibits improved performance and robustness characteristics in multiple extensive randomized experiments. Finally, concluding remarks and future directions are provided in Section VI.

II. Safety Constraint Formulation

Consider the discrete time nonlinear safety critical control system

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

where $k \in \mathbb{Z}_0^+$ is the time step, $x(k) \in \mathcal{D} \subset \mathbb{R}^n$, $u(k) \in \mathcal{U} \subset \mathbb{R}^m$ and $f : \mathcal{D} \times \mathcal{U} \to \mathcal{D}$ is continuous. Throughout the work, we will use the subscript formulation to indicate the time step. For this system, consider the set $\mathcal{S}$ defined as the superlevel set of a smooth function $h : \mathcal{D} \to \mathbb{R}$ such that

$$\begin{aligned} \mathcal{S} &:= \{x_k \in \mathcal{D} \mid h(x_k) \geq 0\} \\ \mathcal{S}^\circ &:= \{x_k \in \mathcal{D} \mid h(x_k) > 0\} \\ \partial\mathcal{S} &:= \{x_k \in \mathcal{D} \mid h(x_k) = 0\} \end{aligned} \tag{2}$$

where $\mathcal{S}^\circ$ and $\partial\mathcal{S}$ are the interior and the boundary of the set $\mathcal{S}$, respectively. Let $\mathcal{S}^\circ$ be the safe set we desire the system's state to stay in. To enforce safety, one needs to satisfy the invariance property given by the following definitions.

Definition 1: The set $\mathcal{S}^\circ \subset \mathbb{R}^n$ is said to be forward invariant for the dynamical system $x(k+1) = f(x(k))$ if $\forall x(0) \in \mathcal{S}^\circ, x(k) \in \mathcal{S}^\circ \; \forall k \in \mathbb{Z}^+$. Equivalently,

$$h(x_k) > 0 \; \forall k \geq 0; \; x(0) \in \mathcal{S}^\circ \tag{3}$$

We refer to this as the safety condition.

Definition 2: The set $\mathcal{S}^\circ \subset \mathbb{R}^n$ is said to be *controlled invariant* for the system in (1) if a continuous feedback controller $u_k = K(x_k)$ exists such that for the closed-loop system $x_{k+1} = f(x_k, K(x_k))$, the set $\mathcal{S}^\circ$ is forward invariant. Accordingly, the controller $u_k = K(x_k)$ is said to be safe.

To render $\mathcal{S}^\circ$ controlled invariant for the discrete control system (1), we define the barrier function $B : \mathcal{S}^\circ \to \mathbb{R}$ [27], to be a smooth function on the interior of $\mathcal{S}$ that goes to infinity as $x_k \in \mathcal{S}^\circ$ approaches a point of $\partial\mathcal{S}$. Mathematically,

$$x_k \in \mathcal{S}^\circ, \tilde{x} \equiv \lim_{k\to\infty} x_k \in \partial\mathcal{S} \Rightarrow B(x_k) \to \infty, k \to \infty$$

Examples of favored barrier functions with such properties over the set $\mathcal{S}$ defined by $h(x_k)$ include logarithmic barriers such as $B(h(x_k)) = -\log(h(x_k))$ and $B(h(x_k)) = -\log(\frac{h(x_k)}{1+h(x_k)})$ and the Carroll barrier, also called the inverse barrier, $B(h(x_k)) = \frac{1}{h(x_k)}$. Clearly, it is sufficient to force B to be bounded to guarantee safety, i.e. keeping $x_k \in \mathcal{S}^\circ \ \forall k$. In light of this, Definition 1, Definition 2 and the properties of the barrier function B, the following proposition follows.

Proposition 1: A continuous feedback controller $u_k = K(x_k)$ is safe, that is it renders $\mathcal{S}^\circ$ controlled invariant, if and only if $B(h(x_0)) < \infty \Rightarrow B(h(x_k)) < \infty \ \forall k \in \mathbb{Z}^+$.

Proof: The proof follows directly from the definitions above.

$\Rightarrow$. Suppose there exists a continuous control law $u_k = K(x_k)$ such that $\mathcal{S}^\circ$ is controlled invariant for w.r.t (1). Then, by Definition 2, $\mathcal{S}^\circ$ is forward invariant w.r.t. the closed loop system $x_{k+1} = f(x_k, K(x_k))$ and consequently, by Definition 1 and the definition of $\mathcal{S}^\circ$, $h(x_k) > 0 \ \forall k \geq 0$ implying $B(h(x_k)) < \infty \ \forall k \geq 0$.

$\Leftarrow$. Assume $B(h(x_0)) < \infty \Rightarrow B(h(x_k)) < \infty \ \ \forall k \in \mathbb{Z}^+$ under the continuous control action $u_k = K(x_k)$. By the properties of the barrier functions, $h(x_k) > 0 \ \ \forall k \geq 0$. Thus, by Definition 1, $\mathcal{S}^\circ$ is forward invariant w.r.t. the closed loop system and hence u_k is said to be safe by Definition 2.

A main objective of this letter is to design a safety enforcing tool that allows us to avoid possible conflicts between control objectives and safety constraints without any relaxation. To achieve this goal, the safety constraint is embedded into the system's model used to achieve control performance objectives by means of discrete barrier states (DBaS).

III. Discrete Time Barrier States (DBaS)

Let us define the barrier function to be $\beta(x_k) := B(h(x_k))$, that is for example for the inverse barrier, $\beta(x_k) = B(h(x_k)) = \frac{1}{h(x_k)}$. Let x^d be the desired state to be tracked. Define $w_k := \beta(x_k) - \beta^d$, where $\beta^d = \beta(x^d)$. Without loss of generality, in the case of stabilization, x^d will be a fixed point, e.g. the origin of the system. Consequently, we derive the discrete barrier state (DBaS) as

$$w_{k+1} = B(h(f(x_k, u_k)) - \beta^d \tag{4}$$

In some robotic applications, e.g. in obstacle avoidance problems, it is more suitable to represent the safe region by a set of functions. In such a problem, increasing the dimension of the system by including too many barrier states may inflate the problem size and complexity. Multiple safety constraints can be represented with only one DBaS by combining the barrier functions. There are some drawbacks to combining barrier states in this way: the process may reduce the amount of information available to the controller as discussed in Section IV, and we may lose access to some explicit information on the safety of the system with respect to certain constraints or obstacles which would be available with multiple barrier states. Therefore, representing multiple constraints with one barrier state introduces a trade-off between state dimension and information available to resulting feedback policies.

In the discrete setting, combining the barrier functions to create a barrier state for the discrete case is simpler than the continuous case [1]. For q constraints, the barrier function can be chosen to be $\beta(x) = \sum_{i=1}^{q} B(h^{(i)}(x_k))$, where $h^{(i)}(x_k)$ describes the i^{th} region of interest. Consequently, a single DBaS can be constructed as

$$w_{k+1} = \sum_{i=1}^{q} B \circ h^{(i)} \left(f(x_k, u_k)\right) - \beta^d \tag{5}$$

where $\beta^d = \sum_{i=1}^{q} B \circ h^{(i)}(x^d)$. It must be noted that shifting the barrier state by β^d is not necessary but ensures that the minimum lies at the desired set point which may be needed for some applications [28]. Now, we append a vector of p barrier states $w \in \mathcal{W} \subset \mathbb{R}^p$ to the model of the safety critical system (1) giving the safety embedded model

$$\hat{x}_{k+1} = \hat{f}(\hat{x}_k, u_k) \tag{6}$$

where $\hat{x}_k = [x_k \quad w_k]^{\mathrm{T}} \in \hat{\mathcal{D}} \subset \mathcal{D} \times \mathcal{W}$ and $\hat{f} : \hat{\mathcal{D}} \times \mathcal{U} \to \hat{\mathcal{D}}$ is a vector field representing the system's dynamics (1) and the barrier states' dynamics (5). It must be noted that $\hat{f}$ is continuous due to continuity of f and smoothness of h and B. As a consequence of the development above, the forward-invariance of $\mathcal{S}^\circ$, i.e. safety of the safety-critical system, can be tied to the performance objectives of the safety embedded system (6). In other words, boundedness of the DBaS implies the generation of safe trajectories. Next, we use a finite horizon trajectory optimization technique, namely differential dynamic programming (DDP), to generate safely optimized trajectories.

IV. Safety Embedded DDP

In this section, we apply the proposed DBaS methodology to safe trajectory optimization by applying DDP [29]–[31] to the safety embedded dynamics (6).

A. Problem Statement

We consider the finite horizon optimal control problem

$$V_k(x) = \min_{U_k} \sum_{i=k}^{N-1} l(x_i, u_i) + \Phi(x_N) \tag{7}$$

subject to the dynamical system (1) and the safety condition (3), where $U_k = \{u_k, u_{k+1} + \cdots + u_{N-1}\}$, l and Φ are the the running cost and terminal cost respectively.

B. Differential Dynamic Programming

The well-known Bellman equation yields the following recurrence relation, with boundary condition $V_N = \Phi$:

$$V_k(x_k) = \min_{u_k} [l(x_k, u_k) + V_{k+1}(f(x_k, u_k))] \tag{8}$$

The DDP algorithm iteratively solves the optimal control problem starting by expanding the value function around a nominal trajectory $(\bar{x}, \bar{u})$ and solving (8) to find a local feedback policy. Then, a new nominal trajectory for the system is computed forwards. The process is repeated until convergence.

Using the proposed DBaS technique to render $\mathcal{S}^\circ$ forward invariant, the constrained finite horizon optimal control problem reduces to an unconstrained optimal control problem that minimizes (7) subject to the safety embedded dynamics (6):

$$V_k(\hat{x}) = \min_{U_k} \sum_{i=k}^{N-1} l(\hat{x}_i, u_i) + \Phi(\hat{x}_N) \tag{9}$$

subject to $\hat{x}_{k+1} = \hat{f}(\hat{x}_k, u_k)$. Therefore, the associated Bellman equation can be given by $V_k(\hat{x}_k) = \min_{u_k}[l(\hat{x}_k, u_k) + V_{k+1}(\hat{f}(\hat{x}_k, u_k))]$. For the DDP equations used in the algorithm, define

$$\begin{aligned}
H_{\hat{x}_k} &= l_{\hat{x}_k} + V_{\hat{x}_{k+1}}^{\mathrm{T}} \hat{f}_{\hat{x}_k},\ H_{u_k} = l_{u_k} + V_{\hat{x}_{k+1}}^{\mathrm{T}} \hat{f}_{u_k} \\
H_{\hat{x}\hat{x}_k} &= l_{\hat{x}\hat{x}_k} + \hat{f}_{\hat{x}_k}^{\mathrm{T}} V_{\hat{x}\hat{x}_{k+1}} \hat{f}_{\hat{x}_k} + V_{\hat{x}_{k+1}} \hat{f}_{\hat{x}\hat{x}_k} \\
H_{uu_k} &= l_{uu_k} + \hat{f}_{u_k}^{\mathrm{T}} V_{\hat{x}\hat{x}_{k+1}} \hat{f}_{u_k} + V_{\hat{x}_{k+1}} \hat{f}_{uu_k} \\
H_{\hat{x}u_k} &= l_{\hat{x}u_k} + \hat{f}_{\hat{x}_k}^{\mathrm{T}} V_{\hat{x}\hat{x}_{k+1}} \hat{f}_{u_k} + V_{\hat{x}_{k+1}} f_{\hat{x}u_k}
\end{aligned} \tag{10}$$

Then, the optimal variation may be given by

$$\delta u_k^* = -H_{uu_k}^{-1} \left(H_{u_k}^{\mathrm{T}} + H_{u\hat{x}_k} \delta\hat{x}_k\right) = \mathbf{k}_k + \mathbf{K}_k \delta\hat{x} \tag{11}$$

where $\mathbf{k}_k = -H_{uu_k}^{-1} H_{u_k}^{\mathrm{T}}$, $\mathbf{K}_k = -H_{uu_k}^{-1} H_{u\hat{x}_k}$ and $\delta x_k = x_k - \bar{x}_k$, $\delta u_k = u_k - \bar{u}_k$ represent deviations from the nominal state and control sequences, respectively. Now, as we need to minimize the expanded Bellman equation, setting each power of approximation to zero leads to the Riccati equations for V_k, $V_{\hat{x}_k}$ and $V_{\hat{x}\hat{x}_k}$ that are solved to get

$$\begin{aligned}
V_k &= V_{k+1} - \frac{1}{2} H_{u_k} H_{uu_k}^{-1} H_{u_k}^{\mathrm{T}} \\
V_{\hat{x}_k} &= H_{\hat{x}_k} - H_{\hat{x}u_k} H_{uu_k}^{-1} H_{u_k} \\
V_{\hat{x}\hat{x}_k} &= \frac{1}{2}(H_{\hat{x}\hat{x}_k} - H_{\hat{x}u_k} H_{uu_k}^{-1} H_{u\hat{x}_k})
\end{aligned} \tag{12}$$

which are the equations used for the backward propagation. Consequently, one can compute V_k and its gradient and Hessian along the states' trajectory as well as the optimal variation δu backwards from $k = N-1$ to $k = 1$ with the initialization $V_N(\hat{x}_N) = l_f(\hat{x}_N)$. Next, the new trajectory is computed forwards and the process is repeated until convergence.

Note that for the DDP problem to be well-defined, we must have $H_{uu_k} \succ 0$ [31]. For this to be the case it is sufficient to have that $V_{\hat{x}\hat{x}_k} \succ 0$ for all k. However, in the general case there is a distinct possibility that l_{xx} is indefinite: it is perfectly reasonable for the cost function to be locally non-convex and in fact this is necessarily the case in the obstacle-avoidance problem. In contrast, the DBaS cost remains a convex function of the states, meaning that for a cost function $l(\hat{x}, u) = l_x(x, u) + l_w(w)$ its hessian $l_{\hat{x}\hat{x}}$ remains positive.

Theorem 2: Assume the state is of the form $\hat{x} = [x\, w]^{\mathrm{T}}$ where x is the real state of the system and w is the barrier state. Further, let $\ell(\hat{x}, u)$ be of the form $\ell(\hat{x}, u) = \ell^x(x, u) + \ell^w(w)$, where $\ell_{ww} \succ 0$ and $\begin{pmatrix} \ell_{xx} & \ell_{xu} \\ \ell_{ux} & \ell_{uu} \end{pmatrix} \succ 0$, and assume second-order dynamics terms f_{xx} are ignored. Then, $V_{\hat{x}\hat{x}_k} \succ 0\ \forall k$.

Proof: By induction: for the base case, we have by assumption that $V_{\hat{x}\hat{x}_N}$ is positive-definite, as it is simply $\Phi_{\hat{x}\hat{x}}$.

Then, we want to show that if $V_{\hat{x}\hat{x}_{k+1}}$ is positive definite we also have that $V_{\hat{x}\hat{x}_k}$ is positive definite. Examine the second-order expansion of the Bellman equation:

$$\begin{aligned}
\delta\hat{x}_k^{\mathrm{T}} V_{\hat{x}\hat{x}_k} \delta\hat{x_k} &= \min_u \begin{pmatrix} \delta x_k \\ \delta u_k \end{pmatrix}^{\mathrm{T}} \begin{pmatrix} \ell_{xx_k} & \ell_{xu_k} \\ \ell_{ux_k} & \ell_{uu_k} \end{pmatrix} \begin{pmatrix} \delta x_k \\ \delta u_k \end{pmatrix} \\
&\quad + \delta w_k^{\mathrm{T}} \ell_{ww_k} \delta w_k + \delta\hat{x}_{k+1}^{\mathrm{T}} \bar{V}_{\hat{x}\hat{x}_{k+1}} \delta\hat{x}_{k+1} \\
&\geq \delta\hat{x}_{k+1}^{\mathrm{T}} V_{\hat{x}\hat{x}_{k+1}} \delta\hat{x}_{k+1} > 0 \\
\delta u_k^{\mathrm{T}} H_{uu_k} \delta u_k &= \delta u_k^{\mathrm{T}} \left(l_{uu} + f_u^{\mathrm{T}} V_{\hat{x}\hat{x}_{k+1}} f_u\right) \delta u \\
&\geq (f_u \delta u)^{\mathrm{T}} V_{\hat{x}\hat{x}_{k+1}} (f_u \delta u_k) \geq 0
\end{aligned}$$

So $V_{\hat{x}\hat{x}_k}$ is positive definite, and furthermore H_{uu} is also positive definite. By induction, this holds for all k.

A natural question is whether it is advantageous to include the barrier state in the model of the system's dynamics instead of simply adding the barrier to the cost function in the optimization problem as in some constrained DDP approaches, known as penalty methods. In those methods, the modified optimization problem given by the cost function $l(x, u) = \beta^2(x) + l'(x, u)$, where β is a barrier function, appears at the surface level to be equivalent to the proposed safety embedded DDP formulation, which we term DBaS-DDP. While any local optimum for the DBaS based optimal control problem is also an optimum for the penalty method, the two mechanisms differ substantially in their interaction with the optimizer. Firstly, in many robotic applications, this new cost function is highly locally non-convex. Theorem 2 explains part of the practical improvement seen when using barrier states over simple penalty methods: it moves some nonlinear terms from the cost function to the dynamics, removing local non-convexity from the problem. In this sense, the DBaS-DDP has a regularizing effect on the algorithm when applied to highly non-convex cost functions. It is worth mentioning that for the experiments presented in Section V, the DBaS-DDP did not need any explicit regularization (i.e. adding a multiple of identity to H_{uu_k} so it is positive definite), unlike the penalty method. Secondly, in the case of DBaS-DDP the optimizer has richer information and can better anticipate the progression of the cost in the forward pass: by embedding the barrier state in the dynamics, the feedback mechanism present in the forwards pass has access to the exact value of the barrier state according to the nonlinear dynamics rather than a quadratic approximation. In this sense, the optimization process can be considered a joint optimization over the real state, barrier state, and controls as decision variables, yielding a smoother cost landscape.

Under certain conditions, with the incorporation of line search, standard DDP is able to guarantee that a single iteration of DDP will improve the trajectory's cost. Similarly, we show that in our formulation, a single iteration of DDP-DBaS is able to find a *safe* trajectory with improved cost.

Theorem 3 (Improvement of Safe Trajectory): Let $(\bar{x}, \bar{u})$ be a safe nominal trajectory, and let $\delta u = \epsilon\mathbf{k} + \mathbf{K}\delta\hat{x}$. If δu_k is nonzero for some k, then there exists some $0 < \epsilon \leq 1$ such that for the objective function $J = \sum_{k=1}^{N-1} l(\hat{x}_k, u_k) + \Phi(\hat{x}_N)$, $J(\bar{x} + \delta x, \bar{u} + \delta u) < J(\bar{x}, \bar{u})$ and $\bar{x} + \delta x, \bar{u} + \delta u$ is a safe trajectory.

TABLE I
COMPARISON OF DBAS-DDP, THE PENALTY METHOD, AND CBF-QP SAFETY FILTERING

Experiment	DBaS		Penalty		CBF	
	%	Cost	%	Cost	%	Cost
Point Robot	**95**	**1.00x**	77	1.17x	79	2.54x
Diff. Wheeled	**82**	**1.00x**	21.7	4.69x	-	-
Cart-pole	-	**1.00x**	-	1.26x	-	1.74x
Quad (1 obstacle)	**100**	**1.00x**	81	1.57x	-	-
Quad (3 obstacles)	**90**	**1.00x**	37	3.84x	-	-
Quad (Random)	**96**	**1.00x**	59	1.87x	-	-

TABLE II
TIMING AND CONVERGENCE INFORMATION. M.I. IS FOR MINIMUM ITERATIONS NEEDED TO REACH THE GOAL AND C.I. IS THE ITERATIONS NEEDED TO ACHIEVE CONVERGENCE

Experiment	Unconstrained		Penalty		DBaS	
	M.I.	C.I.	M.I.	C.I.	M.I.	C.I.
Point Robot[1]	1	1	1.39	3.87	2.82	10.47
Diff. Wheeled	10.48	19.48	11.19	24.40	13.96	25.08
Cart-pole[2]	2	17	284	408	37	63
Quadrotor	1	12	1.24	16.25	1.93	31.82

[1]Unconstrained point robot is an LQ problem and completes in one step.
[2]Cart-pole is a qualitative example, i.e. no randomized experiments.

Proof: Because $\bar{x}, \bar{u}$ is safe and the safe set is open, there exists some neighborhood U of safe trajectories near $\bar{x}, \bar{u}$. Define $J(\pi)$ as the objective function: $J(\pi) = \sum_{k=1}^{N-1} l(\hat{x}_k, u_k) + \Phi(\hat{x}_N)$ with the update rule $\delta u_k = u_k - \bar{u}_k = \mathbf{K}_k \delta \hat{x}_k + \epsilon \mathbf{k}_k$. Find partial derivatives with respect to u_k using Bellman's principle $\Delta J = [l(\hat{x}_k, u_k) + V(f(\hat{x}_k, u_k))] - [l(\bar{x}_k, \bar{u}_k) + V(f(\bar{x}_k, \bar{u}_k))]$ to get $\frac{\partial}{\partial u_k}\Delta J = H_{u_k}$, $\frac{\partial^2}{\partial u_k^2}\Delta J = H_{uu_k}$.

Rewrite the objective function ΔJ as a function of the parameter ϵ, with Taylor expansion around zero:

$$\Delta J(\epsilon) = \sum_{k=1}^{N-1} \left[H_{u_k} \frac{\partial u_k}{\partial \epsilon}\epsilon + \frac{1}{2} \frac{\partial u_k}{\partial \epsilon}^{\mathrm{T}} H_{uu_k} \frac{\partial u_k}{\partial \epsilon}\epsilon^2 \right] + \mathcal{O}(\epsilon^3)$$

Finally, because $u_k = \bar{u}_k + \mathbf{K}_k \delta \hat{x}_k + \epsilon \mathbf{k}_k$, we have $\frac{\partial u_k}{\partial \epsilon} = k_k = -Q_{uu}^{-1}$:

$$\Delta J(\epsilon) = \sum_{k=1}^{N-1} \left[\frac{\epsilon^2}{2} H_{u_k} H_{uu_k}^{-1} H_{u_k}^{\mathrm{T}} - \epsilon H_{u_k} H_{uu_k}^{-1} H_{u_k}^{\mathrm{T}} \right] + \mathcal{O}(\epsilon^3)$$

$$= \left(\frac{1}{2}\epsilon^2 - \epsilon \right) \sum_{k=1}^{N-1} \left[H_{u_k} H_{uu_k}^{-1} H_{u_k}^{\mathrm{T}} \right] + \mathcal{O}(\epsilon^3)$$

By Theorem 2, we know that H_{uu_k} is positive definite. Then, the summation is positive and so for $\epsilon \leq 1$ the lower-order terms are negative. In addition, by making ϵ small we can reduce the higher-order terms to be arbitrarily small compared to the reduction in cost.

Finally, because there is an open neighborhood U of safe trajectories around the nominal trajectory, by making ϵ small we will find a trajectory within the safe set. In particular, because a trajectory is safe if and only if it has a bounded cost by design, any ϵ that leads to a reduction in cost yields a safe trajectory.

V. SAFETY EMBEDDED DDP APPLICATION EXAMPLES

In this section, we conduct qualitative and quantitative comparisons between our proposed algorithm and some commonly-used methods of enforcing safety. Namely, we compare against the penalty-DDP method, in which a barrier cost is added to the state cost function (similar to the DBaS-DDP cost) as described earlier, and the traditional CBF-based method by wrapping DDP with a CBF filter.

Results from all experiments, including success rates and mean costs, are recorded in Table I. In this table costs are normalized to allow for comparison between experiments, by expressing each cost as a fraction of the cost achieved using DBaS-DDP. In each case, DBaS-DDP achieves the lowest cost and the highest success rate.

Convergence speed is detailed in Table II. M.I. indicates the average number of iterations needed to solve a problem (i.e. until the trajectory reaches a threshold of the goal) and C.I. indicates the number of iterations before the algorithm has converged (the difference in cost between successive iterations drops below 10^{-3}). Only successfully completed problems are included in this timing table, as including failures as the maximum iteration count would artificially inflate the number of iterations for the penalty-DDP method (which often fails to reach the goal). It can be seen that DBaS-DDP takes a few more iterations on average but brings substantially higher success rates and lower costs as shown in Table I.

In all experiments in this section, we pick a quadratic cost function $J = \sum_{i=k}^{N-1} x_k^{\mathrm{T}} Q x_k + w_k^{\mathrm{T}} q_w w_k + u_k^{\mathrm{T}} R u_k + x_N^{\mathrm{T}} S x_N + w_N^{\mathrm{T}} s_w w_N$ with $Q = \mathbf{0}_{n\times n}, R = 0.005I$. It is worth mentioning that tuning for DDP parameters specific to the safety embedded case is not required; the parameters from the unconstrained case works well with $q_w > 0$ to ensure safety. The continuous time systems were discretized using Euler methods with $\Delta t = 0.02$. All problems are initialized with a steady-state nominal trajectory (hovering for quadrotor and zero input for other examples). Note that in all experiments, the relative degree of the safety constraint is higher than 1 and is in some cases ill-conditioned, meaning that CBFs are either have limited behavior or impossible to apply. Because of this, we only consider CBFs in the planar double-integrator and cart-pole examples. To address the possibility of stepping across a barrier in a single discrete step, the interior of a barrier is assumed to have infinite cost, ensuring that such trajectories are rejected by line search. This effect can also be addressed with a finer discretization.

A. Planar Double-Integrator (Point Robot)

As a simple proof-of-concept we apply DBaS-DDP to the planar double-integrator problem. In this problem, an omnidirectional robot navigates from an initial position to a goal. The barrier state was defined, by (5), such that the safe set is the exterior of several circles each with safe set defined by $h_i(x) = \|x - o_i\|^2 - r_i^2$. We pick $S = \mathrm{diag}(4000, 4000, 400, 400)$ for terminal cost.

Fig. 2 shows a collision-free course planned by DBaS-DDP starting from the initial point (0,0) to the goal (3,3). The figure also shows solutions generated by the penalty method and a

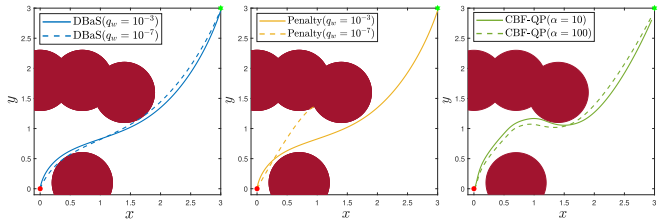

Fig. 2. Planar double-integrator navigation with DBaS (left), penalty (center) and a higher order CBF [14] (right) with different parameters.

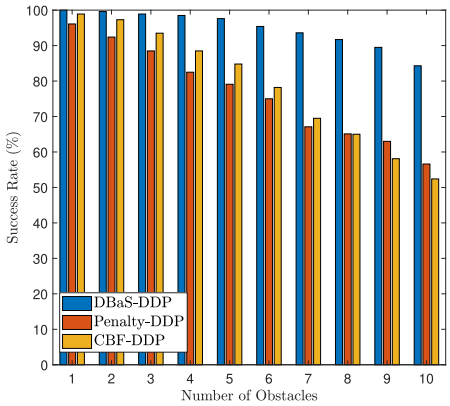

Fig. 3. Success rates of DBaS-DDP (blue), penalty method (red), and CBF (yellow) on planar double integrator example with varying obstacle counts.

higher order CBF [14] QP that is wrapped around the unconstrained solution. Two choices of parameter (q_w for DDP-based methods and α for CBF) are shown for each method. Small barrier penalization q_w causes tighter constraint tolerance near the obstacle, but can cause the penalty method to easily get stuck in local minima. Large values of α in CBFs cause smaller nominal deviation from the trajectory but require sharp inputs when near a barrier.

To empirically test the results of Theorem 2, we compare the eigenvalues of $H_{uu_k}^{-1}$ between the naive penalty method and DBaS-DDP. We find that for the penalty method, we have a minimum eigenvalue (across an entire run of DDP) of -0.173, meaning that although the penalty method is able to find some non-intersecting path in this case, it is despite numerical ill-conditioning. Comparatively, using DBaS-DDP the minimum eigenvalue is 0.1, meaning that H_{uu_k} is numerically well-conditioned across the entire sweep.

We also perform robustness testing with randomized obstacle configurations. In these tests the robot is to safely move from $(0, 0)$ to $(3, 3)$ while avoiding circular obstacles randomly generated in the box with corners at $(3, -2), (5, 0), (0, 5), (-2, 3)$. Success is defined as a trajectory that is able to reach within 0.3 units of the goal. Table I shows the results of penalty-DDP, DBaS-DDP, and CBF over a uniform distribution of obstacle count from 1 to 10 and Fig. 3 shows these results broken down by number of obstacles.

B. Cart-Pole Swing up

We demonstrate the applicability of DBaS-DDP to non-collision-avoidance problems using the cart-pole system, in

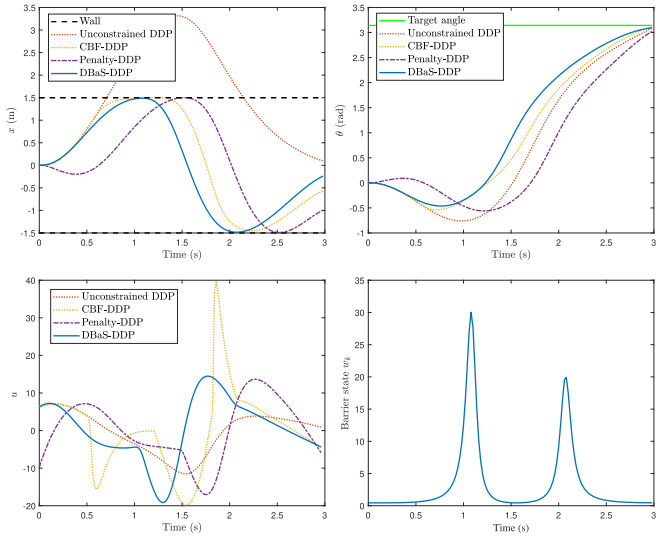

Fig. 4. Cart-pole swing up using unconstrained DDP (red), CBF-QP (yellow), penalty-DDP (purple) and DBaS-DDP (blue). DBaS-DDP respects the constraint on cart position (top left) while reaching the target angle (top right) with smooth control input (bottom left). The DBaS progression over time is shown in the bottom right sub-figure.

which the controller must swing up the pole in 3 seconds by moving the cart while adhering to the safety constraints on the cart's position. We use the system dynamics from [16], but with a tighter constraint in the cart's position and with no modification of the safety constraint to obtain a low relative degree. Namely, we define our safe set by $h(x) = x_{\text{lim}}^2 - x^2$ where $x_{\text{lim}} = 1.5$. We pick $R = 0.05, q_w = 10^{-3}$ and $S = \text{diag}(50, 800, 10, 10)$. For the high order CBF, the vector $\alpha = [100 \;\; 200]$ gave the best feasible results. Results are shown in Fig. 4. Comparatively, we conclude that:

- Unconstrained DDP violates the safety constraint.
- The CBF-based method maintains safety but needed a great deal of tuning to ensure feasibility and complete the task, and requires high control input.
- Penalty-DDP also yields a safe solution but required a lot more iterations to find a solution and converge.

Only DBaS-DDP is able to find the optimal solution for the problem. This is reflected in its comparatively lower cost as shown in Table I. It also converges in much fewer iterations than penalty-DDP as shown in Table II.

C. Differential Wheeled Robot Safe Navigation

The system dynamics are given by $\dot{x} = r\cos\theta(u_1 + u_2)/2, \dot{y} = r\sin\theta(u_1 + u_2)/2, \dot{\theta} = \frac{r}{2d}(u_1 - u_2)$, where x and y are the robot's coordinates, θ is its heading, $r = 0.2$ is the wheel radius, $d = 0.2$ is the wheelbase width, and u_1 and u_2 are the speeds of the right and left wheels respectively. The robot is to safely navigate different courses including randomly generated obstacle courses, a tight course and a course with different geometric shapes. We used the inverse barrier function and augmented the system's dynamics with a single DBaS with $q_w = 10^{-3}$ and $S = 100I_{3\times3}$.

Fig. 5 shows that DBaS-DDP can handle both simple and complex obstacles with barriers as defined in Table III, while the

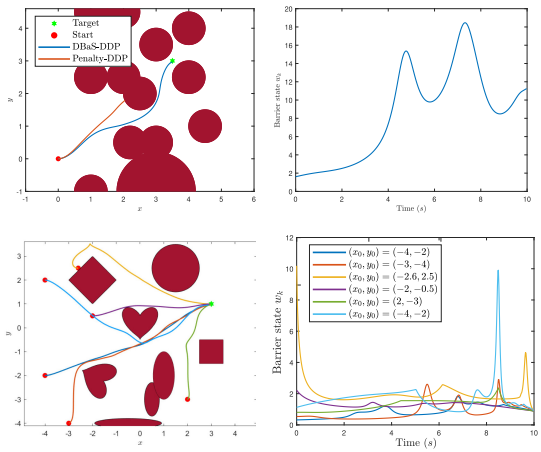

Fig. 5. Left: traces of DBaS-DDP and penalty-DDP (top) on differential wheeled robot, and several traces of DBaS-DDP with complex obstacles (bottom). Robots move from start (red) to goal (green). Right: progression of the associated barrier state over time in which larger values indicate that the robot is close to some obstacles.

TABLE III
EQUATIONS USED FOR COMPLEX OBSTACLE SHAPES

Shape	Function
Ellipse	$a_x x^2 + a_y y^2 - r^2$
Cardioid	$(a_x x^2 + a_y y^2 - 1)^3 - a(a_x x)^2 (a_y y)^3$
Diamond	$\|x\| + \|y\|$
Square	$\|x + y\| + \|x - y\| - 1$

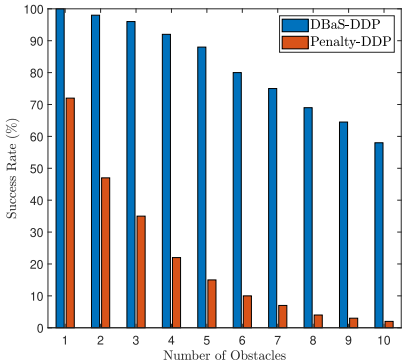

Fig. 6. Success rates of DBaS-DDP (blue) and the penalty method (red) on the differential wheeled robot example in randomized obstacle courses with varying obstacle counts.

naive penalty approach often fails. We also consider randomized courses. The start and target points are drawn from a uniform distribution such that the positions are within a 0.5 unit square around (3,0) and $(-3, 0)$, and with angles up to ± 0.5 rad. Anywhere from 1 to 10 obstacles are created with locations drawn from a normal distribution $\mathcal{N}(0, 1)$ and radii drawn from $\mathbf{U}[0, 1]$. Success is achieved if the generated trajectory reaches within 0.1 units of the target. We performed 1000 trials for each number of obstacles, with results listed by obstacle count in Fig. 6 and summarized in Table I which show the DBaS approach consistently and significantly outperforming the penalty method. Note that because the relative degree is ill-conditioned for this system, we cannot compare results with standard CBF-based methods.

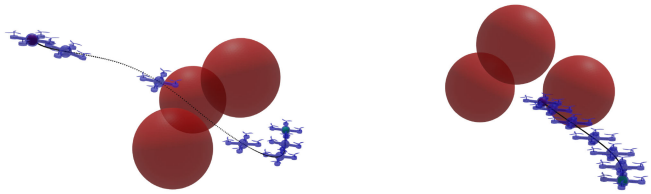

Fig. 7. Quadrotor reaching task with tight squeeze using DBaS-DDP (left) compared to the penalty method (right). Using the DBaS-DDP solver, the quadrotor was able to reach the goal (dark ball) safely while the penalty based DDP solver failed to navigate through the obstacles.

D. Quadrotor Safe Reaching and Tracking

1) Reaching Task: We applied the discrete barrier state based DDP (DBaS-DDP) to a quadrotor model as described in Sabatino [32] with unity parameters (1 kg, 1kg m^2, etc.). The quadrotor was to perform a *reaching* problem safely, i.e. to fly from some initial state to some arbitrary final state in the presence of some obstacles without collision. The safe set is again defined as the complement of a set of spherical obstacles. A solution to the quadrotor reaching problem found by DBaS-DDP with randomly-generated obstacles is shown in Fig. 1.

DBaS-DDP was compared against penalty-DDP in two fixed environments: a single-obstacle case where the quadrotor must navigate around a single large spherical obstacle and a three-obstacle case (shown in Fig. 7) designed to add a local minimum and a narrow passage between the obstacles. When testing in these fixed environments with random initial conditions, we see that DBaS-DDP successfully finds a path to the goal much more frequently than penalty-DDP as shown in Table I. Note that *failure* in this case indicates failure to reach the goal, rather than a failure to maintain safety: as described in Proposition 1, the trajectory and controller found by DBaS-DDP is guaranteed to be safe in all cases.

DBaS-DDP was also tested in the presence of 40 randomized obstacles in an environment similar to Fig. 1. In this case, DBaS-DDP reached the goal 96% of the time while the penalty-based method only succeeded 59% of the time and found substantially higher-cost trajectories.

While there exist CBF-based obstacle avoidance methods for quadrotors, they are tedious to construct for the full 3D model and comparison is out of the scope of this analysis.

2) Tracking Task: The technique of barrier states can also generalize to the *tracking* problem, in which we want to safely track some (possibly unsafe) reference trajectory. To put the DBaS-DDP to the test, we attempted to track the trajectory defined by the parametric equations for a figure eight:

$$\mathrm{x(s)} = \sin(2\mathrm{s}), \mathrm{y(s)} = \cos(\mathrm{s}), \mathrm{z(s)} = 0, \mathrm{s(t)} = \frac{(\pi \mathrm{t}/25)^2}{\pi \mathrm{t}/25 + 1}$$

Then, the squared deviation of the quadrotor's trajectory from this path was penalized in the cost function. In addition, we placed an obstacle at the origin forcing the quadrotor to navigate around the obstacle to remain safe. Fig. 8 shows an execution trace from this experiment. The quadrotor was able to successfully track the trajectory in a very aggressive maneuver without losing safety.

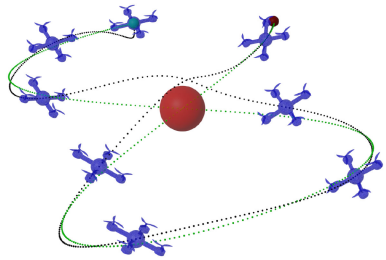

Fig. 8. Quadrotor tracking a predetermined trajectory (green) while avoiding a spherical obstacle. Starting from the green ball, the safe trajectory from DBaS-DDP (black) successfully avoids the obstacle to reach the red ball.

VI. Conclusion

In this work, the newly proposed barrier state method for stabilization of continuous time systems was extended to trajectory optimization of discrete time systems. This extension, named the discrete barrier state (DBaS) method, provides provable safety guarantees when combined with DDP for safe trajectory optimization problems. To show the efficacy of the proposed safety embedded DDP, we presented several comparisons with other commonly used methods and successful simulation examples for a constrained cart-pole swing up, safe holonomic and non-holonomic robot navigations, and a quadrotor performing safety-critical planning and execution, demonstrating improvements in comparison to the other methods on each problem.

Our work requires perfect knowledge of the system's dynamics and, similarly, assumes full knowledge of state and safety constraints, which may not hold true in real-world applications where the former may require model identification or leaning and the latter are recovered from sensor measurements. Incorporating dynamics, state, and safety constraint uncertainty, for example using Gaussian Process regression, into the DBaS framework represents a promising direction for future research. Furthermore, future work will include improving robustness by extending the DBaS-DDP to min-max and risk-sensitive optimal control problems. Additionally, we are currently developing real-time implementations of DBaS-DDP in a lower-level language and plan to conduct physical experiments using a receding-horizon formulation.

References

[1] H. Almubarak, N. Sadegh, and E. A. Theodorou, "Safety embedded control of nonlinear systems via barrier states," *IEEE Control Syst. Lett.*, vol. 6, pp. 1328–1333, 2022 doi: 10.1109/LCSYS.2021. 3093255.

[2] F. Blanchini, "Set invariance in control," *Automatica*, vol. 35, no. 11, pp. 1747–1767, 1999.

[3] S. Prajna, "Barrier certificates for nonlinear model validation," in *Proc. 42nd IEEE Int. Conf. Decis. Control (IEEE Cat. No 03CH37475)*, vol. 3, 2003, pp. 2884–2889.

[4] S. Prajna and A. Jadbabaie, "Safety verification of hybrid systems using barrier certificates," in *Proc. Int. Workshop Hybrid Syst.: Comput. Control*, 2004, pp. 477–492.

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

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

[7] M. Z. Romdlony and B. Jayawardhana, "Uniting control Lyapunov and control barrier functions," in *Proc. 53rd IEEE Conf. Decis. Control*, 2014, pp. 2293–2298.

[8] A. D. Ames, X. Xu, J. W. Grizzle, and P. Tabuada, "Control barrier function based quadratic programs for safety critical systems," *IEEE Trans. Autom. Control*, vol. 62, no. 8, pp. 3861–3876, Aug. 2017.

[9] A. Agrawal and K. Sreenath, "Discrete control barrier functions for safety-critical control of discrete systems with application to bipedal robot navigation," *Robot: Sci. Syst.*, vol. 13, 2017.

[10] J. Choi, F. Castaneda, C. J. Tomlin, and K.Sreenath, "Reinforcement learning for safety-critical control under model uncertainty, using control Lyapunov functions and control barrier functions," *Robotics: Sci. Syst.*, vol. 16, 2020.

[11] A. J. Taylor and A. D. Ames, "Adaptive safety with control barrier functions," in *Proc. Amer. Control Conf.*, 2020, pp. 1399–1405.

[12] L. Wang, E. A. Theodorou, and M. Egerstedt, "Safe learning of quadrotor dynamics using barrier certificates," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2018, pp. 2460–2465.

[13] M. Ahmadi, A. Singletary, J. W. Burdick, and A. D. Ames, "Safe policy synthesis in multi-agent pomdps via discrete-time barrier functions," in *Proc. IEEE 58th Conf. Decis. Control*, 2019, pp. 4797–4803.

[14] Q. Nguyen and K.Sreenath, "Exponential control barrier functions for enforcing high relative-degree safety-critical constraints," in *Proc. Amer. Control Conf.*, 2016, pp. 322–328.

[15] W. Xiao and C. Belta, "Control barrier functions for systems with high relative degree," in *Proc. IEEE 58th Conf. Decis. Control*, 2019, pp. 474–479.

[16] M. Pereira, Z. Wang, I. Exarchos, and E. Theodorou, "Safe optimal control using stochastic barrier functions and deep forward-backward SDES," in *Proc. Conf. Robot Learn.*, Nov. 2020, vol. 155, pp. 1783–1801.

[17] K. Long, C. Qian, J. Cortés, and N. Atanasov, "Learning barrier functions with memory for robust safe navigation," *IEEE Robot. Automat. Lett.*, vol. 6, no. 3, pp. 4931–4938, Jul. 2021.

[18] W. Xiao, C. A. Belta, and C. G. Cassandras, "Feasibility-guided learning for constrained optimal control problems," in *Proc. 59th IEEE Conf. Decis. Control*, 2020, pp. 1896–1901.

[19] L. Wang, A. D. Ames, and M. Egerstedt, "Multi-objective compositions for collision-free connectivity maintenance in teams of mobile robots," in *Proc. IEEE 55th Conf. Decis. Control*, 2016, pp. 2659–2664.

[20] D. M. Murray and S. J. Yakowitz, "Constrained differential dynamic programming and its application to multireservoir control," *Water Resour. Res.*, vol. 15, no. 5, pp. 1017–1027, 1979.

[21] Y. Tassa, N. Mansard, and E. Todorov, "Control-limited differential dynamic programming," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2014, pp. 1168–1175.

[22] B. Plancher, Z. Manchester, and S. Kuindersma, "Constrained unscented dynamic programming," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2017, pp. 5674–5680.

[23] T. A. Howell, B. E. Jackson, and Z. Manchester, "ALTRO: A. fast solver for constrained trajectory optimization," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2019, pp. 7674–7679.

[24] Z. Xie, C. K. Liu, and K. Hauser, "Differential dynamic programming with nonlinear constraints," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2017, pp. 695–702.

[25] Y. Aoyama, G. Boutselis, A. Patel, and E. A. Theodorou, "Constrained differential dynamic programming revisited," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2021, pp. 9738–9744. doi: 10.1109/ICRA48506.2021. 9561530.

[26] A. Pavlov, I. Shames, and C. Manzie, "Interior point differential dynamic programming," *IEEE Trans. Control Syst. Technol.*, vol. 29, no. 6, pp. 2720–2727, Nov. 2021.

[27] A. Ben-Tal and A. Nemirovski, *Lectures on Modern Convex Optimization*. Society for Industrial, 2001. doi: 10.1137/1.9780898718829.

[28] A. G. Wills and W. P. Heath, "Barrier function based model predictive control," *Automatica*, vol. 40, no. 8, pp. 1415–1422, 2004.

[29] D. Mayne, "A second-order gradient method for determining optimal trajectories of non-linear discrete-time systems," *Int. J. Control*, vol. 3, no. 1, pp. 85–95, 1966.

[30] D. H. Jacobson, "Differential dynamic programming methods for determining optimal control of non-linear systems," Ph.D. dissertation, Elect. Eng., Centre for Comput. and Automat., Imperial College of Sci. and Technol., Univ. of London, 1967.

[31] D. H. Jacobson and D. Q. Mayne, *Differential Dynamic Programming*. American Elsevier Publishing Company North-Holland, 1970. [Online]. Available: https://books.google.com/books?id=tA-oAAAAIAAJ

[32] F. Sabatino, "Quadrotor control: Modeling, nonlinearcontrol design, and simulation," Master Thesis, KTH, School Elect. Eng., Automat. Control., Sweden2015.