Received 19 September 2022, accepted 20 October 2022, date of publication 26 October 2022, date of current version 2 November 2022.

*Digital Object Identifier 10.1109/ACCESS.2022.3217239*

RESEARCH ARTICLE

# Discrete-Time Differential Dynamic Programming on SO(3) With Pose Constraints

**SHU LIU AND DONGPO LIU**
China Academy of Information and Communications Technology, Beijing 100191, China
Corresponding author: Shu Liu (liushu.caict@gmail.com)

**ABSTRACT** The motion of many robotics systems, such as the rotation of unmanned aerial vehicles, can be modeled by SO(3). However, the difficulties in the parameterization of the SO(3) makes it hard to implement the state-of-the-art collision avoidance algorithm. In this paper, we present a new method for control on SO(3) via the combination of the nonlinear state constraints and geometric control technique. We first define the nonlinear constraints for trajectory optimization on SO(3). Then we solve the trajectory optimization problem by constrained Differential Dynamic Programming (DDP) in a Riemannian geometry framework. The first and second-order expansion of the cost function and dynamics are derived in a coordinate-free manner. The safety condition represented by nonlinear constraints is incorporated into the DDP via an active set method, where we find the active set in the backward path to obtain the optimal control policy. We validated our methods on motion planning of rigid model on SO(3) manifold where pose constraints are imposed. Our methods outperform the baseline methods in terms of convergence speed and numerical robustness to disturbance. The numerical robustness is essential when the system is initialized far from the local optimum.

**INDEX TERMS** Geometric control, motion planning, differential dynamic programming, nonlinear programming.

## I. INTRODUCTION

The motions of many robotics systems are evolving on Lie groups [1], [2]. For example, the rotational motion of unmanned aerial vehicle [3], [4], [5], [6] and torso of humanoid robot [7], [8] are evolving on the special orthogonal group SO(3) [2]. Each element of the SO(3) is the three dimensional rotational matrix that can satisfy the following condition $\{R|R^TR = I, R \in \mathbb{R}^3, \det R = 1\}$. Unlike $\mathbb{R}^n$ Euclidean space where most numerical optimization [9], [10] is applied, the systems on SO(3) are evolving on manifolds that are hard to parameterize. Therefore, designing the controller on these systems is not trivial. Thus it poses a question to us how we can design the collision avoidance algorithm on such manifolds as SO(3).

Conventional methods for systems on SO(3) need to use local coordinates like Euler angles, which, however, is known for singularity or the gimbal lock [3], [11]. Though the quaternion is a global representation, the ambiguity issue also restricts its applications, such that one configuration may have multiple different representations [3], [11]. To tackle these problems, the development of geometric control has introduced the Riemannian geometry to the control problems that solve the control on manifolds in a coordinate-free manner [12], [13], [14]. Reference [12] models the mechanical system on a general Riemannian manifold and applies the PD control with proof of convergence property. The Riemmanian geometry also enables the expression of the coordinate-dependent control theory in a coordinate-free manner. For example, the contraction theory [15] can be expressed in a Riemmanian geometry language using the internal quantity without specifying the coordinates [14]. The recent development of geometric control on the Lie group has applied the variational-based linearization [16] techniques that approximate the error dynamics around the reference trajectory. Compared to conventional linearization in Euclidean space, the variational-based linearization will result in state independent matrix that has superior numerical properties

The associate editor coordinating the review of this manuscript and approving it for publication was Min Wang.

in optimization. The variational-based linearization has been applied to robotics systems such as unmanned aerial vehicles [17], and legged robots [18].

In recent years, the optimization-based collision avoidance algorithm has been proposed and applied to robotics systems, such as mobile manipulators [19], unmanned aerial vehicles (UAV) [20], [21] and autonomous driving [22], [23]. These methods treat the safety condition via nonlinear constraints that restrict the robots' motion in the safe regions. Due to the non-convexity of the general nonlinear safe region, the convex decomposition methods have been proposed and applied to decompose the original optimization problem to a sequence of convex optimizations [24]. Given some assumptions on the convexity of the objective function, convergence and optimality could be guaranteed. However, these methods are mainly direct methods that do not assume optimality conditions; therefore, they may result in low accuracy in real-time application. Among the application, the control of the rotational motion of UAV is on SO(3) that is not properly solved using conventional parameterization method. The existing methods widely utilize the Euler angles to parameterize the trajectories [3], [25], [26], [27], thus may face issues in the singularities [11]. In order to solve the singularities problem, the existing methods need to manually adjust the yaw angle to the range $[-\pi, \pi]$ and avoid the possible discontinuity in this range adjustment. If the continuity is not dealt with correctly, the motion would be corrupted with more snap that is not desired [5], [28].

Other than the direct method, the indirect method that imposes optimality first condition and then executes the optimization has been widely applied. One of the indirect methods is the Differential Dynamic Programming (DDP) algorithm [29]. The DDP is composed of the forward pass for trajectory integral and the backward pass that compute the optimal control input. By imposing the optimality condition in the backward recursion, the optimal feedback gain could be obtained and applied in the forward path to integrate the system trajectory. Through DDP, the feedback controller and optimal trajectory can be obtained simultaneously. The recent application has also considered the input saturation [30], general nonlinear constraints [31], and global solution search by random search [32]. In the work of [30], the box bounds induced from input limits are considered. In the backward pass, this work does not directly apply the search direction from DDP but projects it to the feasible direction considering the box constraints. Compared to simply truncating the input in the forward pass, the proposed method has a faster convergence rate. To solve the more general nonlinear constraints, the work of [31] incorporates the barrier function [33] to the cost function to ensure the system is within the safe set. However, the barrier function methods may have a slow convergence rate due to the choice of barrier function shape and parameters [9]. Other than using barrier function, the active set methods in optimization theory have also been very successful in recent years [34], [35], [36], [37]. Though the worst-case performance of the active set method is not good due to the combinatorial nature of active set selection [9], however, in control design and trajectory optimization, a rough estimation of active set is accessible, and a warm start is possible to ensure the feasibility and fast convergence. More recent work on Model Predictive Control (MPC) has demonstrated the performance of the active set method [38], [39], [40]. The MPC executes the trajectory optimization at every time step and the state at consecutive time steps is quite close to the former one, which makes the warm start reliable.

Though optimization-based trajectory optimization and geometric-based control methods are widely applied, there is no unification method that can handle collision avoidance and on-manifold system dynamics at the same time. In order to apply the collision avoidance algorithms to the system on the Lie group, several challenges need to be resolved:

- The parameterization of the dynamics and the construction of a safe set on SO(3) manifold that is totally different from the Euclidean space.
- The incorporation of inequality constraints on SO(3) manifold in the DDP settings.
- The possible infeasibility issues in the DDP method with inequality constraints.

In order to solve the aforementioned issues, this paper incorporates the state-of-the-art geometric DDP algorithm [41] with safety encoded as general nonlinear constraints in a coordinate-free manner and applies it to safe trajectory generation on SO(3). The key point of this work is that we incorporate the active set method [9] and slack variable into the DDP to tackle the inequality constraints in trajectory optimization in the geometric control settings. The incorporation of active set update would enable us to solve constrained trajectory optimization using DDP that is designed for unconstrained problem. The utilization of geometric control formulation enables us to use the exponential coordinates that do not suffer from singularities in conventional methods. The main contribution of this paper could be summarized as follows:

- We proposed the DDP algorithm on SO(3) under the Riemmanian geometry settings with nonlinear constraints. The active set method is incorporated in the DDP backward path to handle the nonlinear constraints iteratively. A slack variable method is incorporated into the forward path to integrate the dynamics while avoiding infeasibility.
- We construct the unsafe pose avoidance condition as a nonlinear constraint and incorporate it into the proposed constrained DDP. The first-order expansion of the pose constraints is derived. The proposed construction is parameterized by the exponential coordinate that is global and do not suffer from the singularities in conventional methods.
- Numerical analysis has shown that the proposed algorithm could successfully plan a trajectory on SO(3) that avoid the unsafe pose and outperform the baseline

methods. Our method has a faster convergence rate and higher robustness to numerical disturbances than the state-of-the-art methods.

The remainder of this paper is organized as follows. Section II provides the math preliminaries. Section III introduces the unconstrained DDP algorithm on Lie groups. The DDP with nonlinear constraints is discussed in Section IV. Experiments of trajectory optimization on SO(3) are presented in Section V. Discussion and conclusion are presented in section VI and VII, respectively.

## II. PRELIMINARY

In this section, we introduce the preliminary knowledge of the proposed algorithm. We first discuss the notation in differential geometry. The main math notations are referred to [41] for consistency.

### A. DIFFERENTIAL GEOMETRY AND LIE GROUP

#### 1) LIE GROUPS

We denote the $n$ dimensional Lie group as $G$. The identity element of the $G$ is defined as $e$. The left translation map $L_h : G \to G$ is defined as:

$$L_h g = hg, \quad \forall h,\ g \in G. \tag{1}$$

Equivalently, the right translation map $R_h : G \to G$ is defined as

$$R_h g = gh, \quad \forall h,\ g \in G. \tag{2}$$

The tangent and cotangent bundles of the $G$ are denoted by $TG$ and $TG^*$, respectively. We define the Lie algebra as $\mathfrak{g} := T_eG$ and $\mathfrak{g}^* := T_e^*G$ is its dual. The left tangent map of $L_h$ at $g$ denoted by $T_gL_h = TG \to TG$. Equivalently, we denote the right tangent map of $R_h$ at $g$ as $T_gR_h = TG \to TG$. As the group action can be denoted by the left or right multiplication, for simplification, we have:

$$T_eL_g = g\xi, \quad \forall \xi \in \mathfrak{g},\ g \in G. \tag{3}$$

$$T_eR_g = \xi g, \quad \forall \xi \in \mathfrak{g},\ g \in G. \tag{4}$$

Let $\mathcal{X}$ denote the set of smooth vector fields on $G$. For any smooth function $f : G \to G$, we could define the Lie bracket $[\cdot,\cdot] : \mathcal{X} \times \mathcal{X} \to \mathcal{X}$, such that

$$[X, Y](f) := X(Y(f)) - Y(X(f)).$$

We define the adjoint representation of $G$ $\mathrm{Ad}_g : \mathfrak{g} \in \mathfrak{g}$ as

$$\mathrm{Ad}_g(\phi)\eta = g\eta g^{-1}.$$

The adjoint representation could be considered as a change of basis transformation. Furthermore, the adjoint representation of $\mathfrak{g}$ is denoted as $\mathrm{ad}_\eta \zeta := [\eta, \zeta]$.

#### 2) NATURAL PAIRING AND DUAL MAPS

For a vector space $V$ and its dual $V^*$, we define the bilinear map $\langle\cdot,\cdot\rangle : V^* \times V \to \mathbb{R}$ as their natural pairing, such that

$$\langle \phi, x\rangle := \phi(x), \quad \forall x \in V,\ \forall \phi \in V^*.$$

For any linear map $h : V \to W$ between vector spaces, its dual map $h : W^* \to V^*$ is restricted to have the following property $\langle \phi, h(x)\rangle = \langle h^* \circ \phi, x\rangle, \forall \phi \in W^*$.

The natural pairing enables us to define the inner product that is essential to compute the distance in the Riemannian manifold settings.

#### 3) DERIVATIVES

Let $X, Y \in \mathcal{X}$ be two vector fields on $G$. Given an affine connection $\nabla : \mathcal{X} \times \mathcal{X} \to \mathcal{X}$, the covariant derivative of $Y$ with respect to $X$ is denoted by $\nabla_X Y$. The covariant derivative could be considered as a generalized directional derivative for vector fields.

Consider scalar value function $f : G \to \mathbb{R}$ that is twice differentiable. Let $\xi \in T_gG$, $Y \in \mathrm{X}$ and $Y : G \to TG$. The differential of $f$ at $g \in G$ is denoted by $\mathrm{D}f(g) : T_gG \to \mathbb{R}$ and satisfy $\xi(f(g)) = \mathrm{D}f(g)(\xi)$. For simplification, we use $\mathrm{D}_h f$ to denote the differential of $f$ with respect to $h$.

The Hessian operator is denoted as $\mathrm{Hess}f(g) : T_gG \to T_g^*G$ and satisfy the identity: $\mathrm{D}(\mathrm{D}f(Y))(g)(\xi) = \mathrm{D}f(g)(\xi)(Y(g)) + \mathrm{D}f(g)(\nabla_\xi Y)$. The Hessian can also be considered as the second-order covariant derivative.

Finally, given any two vector spaces $K$ and $V$, let $\mathfrak{L}(K, V)$ denote the set of linear maps from $K$ to $V$. We define the exponential functor, $(\cdot)^W$, acting on a linear map $g : K \to V$, such that $\mathrm{g}^W : \mathfrak{L}(W, K) \to \mathfrak{L}(W, V)$, with $\mathrm{g}^W(\theta) := \mathrm{g} \circ \theta$, for any vector space $W$ and linear map $\theta \in \mathfrak{L}(W, K)$.

## III. DDP ON LIE GROUPS

In this section, we introduce the unconstrained DDP. The DDP with nonlinear constraints also requires the same procedure to expand the cost function and system dynamics and update the value function. The main theory is based on [41]. The idea of DDP is to decrease the cost function by iteratively solving local optimal control problems and update the trajectories. The DDP consists of two parts, namely, the backward pass and the forward pass. In the backward pass, the DDP computes a locally optimal policy around the current trajectory that can decrease the cost function. As this policy is induced from dynamic programming that starts the recursion from the terminal states thus, we call it backward pass. In the forward pass, the system executes the optimal policy to integrate a new trajectory from the initial states. The main process of DDP is illustrated in Fig. 1.

The main structure of this section is as follows. III-A presents the formulation of the optimal control problem. III-B to III-E introduced the main procedures in the backward pass. III-B and III-C introduced the expansion of the state trajectory and value function. III-D and III-E introduced how to obtain the local linear optimal policy and the update of the value function by the dynamic programming from the backward. III-F introduces the procedures in the forward pass for trajectory integration. Here, we review the expansion in a coordinate-free manner, and the specific form of the expansion will be determined when a certain basis is chosen.

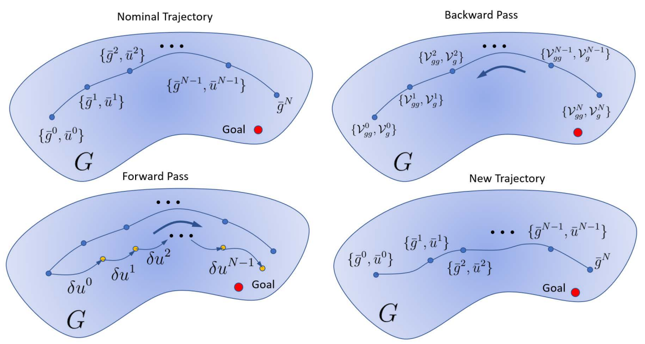
Nominal Trajectory
$\{\bar{g}^0, \bar{u}^0\}$
$\{\bar{g}^1, \bar{u}^1\}$
$\{\bar{g}^2, \bar{u}^2\}$
$\{\bar{g}^{N-1}, \bar{u}^{N-1}\}$
$\bar{g}^N$
Goal
$G$
Backward Pass
$\{\mathcal{V}^0_{gg}, \mathcal{V}^0_g\}$
$\{\mathcal{V}^1_{gg}, \mathcal{V}^1_g\}$
$\{\mathcal{V}^2_{gg}, \mathcal{V}^2_g\}$
$\{\mathcal{V}^{N-1}_{gg}, \mathcal{V}^{N-1}_g\}$
$\{\mathcal{V}^N_{gg}, \mathcal{V}^N_g\}$
$G$
Forward Pass
$\delta u^0$
$\delta u^1$
$\delta u^2$
$\delta u^{N-1}$
Goal
$G$
New Trajectory
$\{\bar{g}^0, \bar{u}^0\}$
$\{\bar{g}^1, \bar{u}^1\}$
$\{\bar{g}^2, \bar{u}^2\}$
$\{\bar{g}^{N-1}, \bar{u}^{N-1}\}$
$\bar{g}^N$
Goal
$G$

**FIGURE 1. Unconstrained Differential Dynamic Programming on Lie group. The DDP starts from an initialized nominal trajectory and repeats the backward pass and forward pass until convergence. In the backward pass, the DDP computes the expansion of the value function and derives the optimal control law. In the forward pass, the DDP integrates the system trajectory based on the optimal control law. The trajectory is supposed to converge iteratively to the optimal local trajectory.**

## A. PROBLEM FORMULATION

Consider the system dynamics on Lie group $G$ as

$$g^{k+1} = f(g^k, u^k), \tag{5}$$

where $g^k \in G$ is the state on the Lie group, $u^k \in \mathbb{R}^m$ denotes the input. Thus, we have the discrete-time constrained optimal control problem of the following form:

$$\begin{aligned} \min_{\{u^k\}_0^{N-1}} \; & F(g^N) + \sum_{k=0}^{N-1} \Lambda(g^k, u^k) \\ \text{s.t. } & g^{k+1} = f(g^k, u^k) \\ & g^0 = g(0) \\ & k = 0, 1, \ldots, N-1, \end{aligned} \tag{6}$$

where $\Lambda$ is the running cost and $F$ is the terminal cost. We consider the boundary condition as the initial state $g^0 = g(0)$ and the planning horizon being $N > 0$.

This optimization problem is generally hard to optimize due to its nonlinearity. However, we can use the DDP algorithm to obtain a local minimum. The basic philosophy is that if the nominal trajectory is not optimal, we could find a perturbation around the trajectory that decrease the cost function. Then, the main procedure of the DDP is to expand the cost function and the dynamics around a local nominal trajectory, apply the dynamic programming to decrease the cost function locally, and repeat the procedure until convergence.

## B. PERTURBED STATE TRAJECTORY

We consider the nominal system trajectory $\{\bar{u}^k\}_0^{N-1}$ and $\{\bar{g}^k\}_0^N$. We also consider the perturbation of the trajectory given by the perturbed input $\{\bar{u}^k_\epsilon\}_0^{N-1} := \{\bar{u}^k + \delta u^k\}_0^{N-1}$. Note that we assume the deviation $\{\delta u^k\}_0^{N-1}$ is small enough to make the perturbed state trajectory $\{\bar{g}^k_\epsilon\}_0^N$ remain in the neighborhood of the nominal one. Therefore, we could utilize the local coordinates of the exponential map to denote the perturbed states:

$$\{\bar{g}^k_\epsilon\}_0^N := \{\bar{g}^k \exp(\zeta^k)\}_0^N, \quad \zeta^k \in \mathfrak{g}. \tag{7}$$

The linearized perturbation $\zeta^k \in \mathfrak{g}$ of the current trajectory is incremental in the tangent space at the current configuration. To indicate the perturbations on the group $G$, we mapped it to the $G$ by the exponential map and multiplied it at the right-hand side of the current configuration $g^k$ [2]. As we consider the initial value is fixed, such that $g_0 = g(0)$, we set $\zeta_0 = \zeta(0)$ to impose this constraints. For simplification, we denote the nominal trajectory by the concatenation of nominal inputs and system trajectory:

$$\bar{\tau}_0^N := \{\{\bar{u}^k\}_0^{N-1}, \{\bar{g}^k\}_0^N\} \tag{8}$$

The DDP requires second-order expansion of the dynamics of the perturbed state $\{\zeta^k\}$ to compute the dynamic programming solution. As is introduced in [41], such a linearization scheme exists, and now we give its form as follows:

$$\begin{aligned} \zeta^{k+1} \approx \; & \Phi^k\left(\bar{\tau}_0^N\right)\left(\zeta^k\right) + \mathrm{B}^k\left(\bar{\tau}_0^N\right)\left(\delta u^k\right) \\ & + \frac{1}{2}\Big(\Theta^k\left(\bar{\tau}_0^N\right)\left(\zeta^k\right)\left(\zeta^k\right) \\ & + \Gamma^k\left(\bar{\tau}_0^N\right)\left(\zeta^k\right)\left(\delta u^k\right) + \Delta^k\left(\bar{\tau}_0^N\right)\left(\delta u^k\right)\left(\zeta^k\right) \\ & + \Xi^k\left(\bar{\tau}_0^N\right)\left(\delta u^k\right)\left(\delta u^k\right)\Big) \end{aligned} \tag{9}$$

where $\Phi^k\left(\bar{\tau}_0^N\right) : \mathfrak{g} \to \mathfrak{g}$, $\mathrm{B}^k\left(\bar{\tau}_0^N\right) : \mathbb{R}^m \to \mathfrak{g}$, $\Theta^k\left(\bar{\tau}_0^N\right) : \mathfrak{g} \times \mathfrak{g} \to \mathfrak{g}$, $\Gamma^k\left(\bar{\tau}_0^N\right) : \mathfrak{g} \times \mathbb{R}^m \to \mathfrak{g}$, $\Delta^k\left(\bar{\tau}_0^N\right) : \mathbb{R}^m \times \mathfrak{g} \to \mathfrak{g}$, and $\Xi^k\left(\bar{\tau}_0^N\right) : \mathbb{R}^m \times \mathbb{R}^m \to \mathfrak{g}$ are all linear in their arguments. We also require that $\Theta^k_{(i)}\left(\bar{\tau}_0^N\right) = \left(\Theta^k_{(i)}\left(\bar{\tau}_0^N\right)\right)^*$, $\Xi^k_{(i)}\left(\bar{\tau}_0^N\right) = \left(\Xi^k_{(i)}\left(\bar{\tau}_0^N\right)\right)^*$ and $\Delta^k_{(i)}\left(\bar{\tau}_0^N\right) = \left(\Gamma^k_{(i)}\left(\bar{\tau}_0^N\right)\right)^*$, for all $k, i$. The subscript here corresponds to a particular component of an operator (i.e., given a basis $\{E_i\}$ in $\mathfrak{g}$, we take $\Theta^k\left(\bar{\tau}_0^N\right)\left(\zeta^k\right)\left(\zeta^k\right) = \sum_i \Theta^k_{(i)}\left(\bar{\tau}_0^N\right)\left(\zeta^k\right)\left(\zeta^k\right)E_i$, etc.).

The dynamics (9) is an approximation of the perturbed state trajectory in the Lie algebra of the group.

## C. VALUE FUNCTION EXPANSION

We define the value function $V^k : G \to \mathrm{R}$ at time step $k$ as:

$$V^k(g^k) := \min_{\{u^j\}_k^{N-1}} F(g^N) + \sum_{j=k}^{N-1} \Lambda(g^j, u^j) \tag{10}$$

The above value function indicates the optimal cost at timestamp $k$ and state $g^k$. The value function, in general, can be highly nonlinear that is intractable. To handle this issue, we expand the value function to second-order so we could apply the dynamic programming algorithm with the linearized perturbed state dynamics.

Bellman's principle indicates the following recursion:

$$V^k(g^k) = \min_{\{u^k\}}\left[L(g^k, u^k) + V^{k+1}(g^{k+1})\right], \tag{11}$$

such that the tail of the optimal trajectory is optimal. Thus the optimal cost to go could be computed backward by the above recursion. The key idea of DDP is that if the nominal trajectory is not optimal, the value function could be decreased by applying a small perturbation around the nominal trajectory. As linear quadratic optimal control is well studied and a second-order approximation of the value function is accessible via Taylor expansion, we expand the value function to its second order around the optimal trajectory by:

$$\begin{aligned} V^k\left(g^k_\epsilon\right) &= V^k\left(\bar{g}^k\right) + \mathrm{D}V^k\left(\bar{g}^k\right)\left(T_eL_{\bar{g}^k}\zeta^k\right) \\ &\quad + \frac{1}{2}\mathrm{Hess}^{(0)}V^k\left(\bar{g}^k\right)\left(T_eL_{\bar{g}^k}\zeta^k\right)\left(T_eL_{\bar{g}^k}\zeta^k\right) \\ &\quad + O\left(\left\|\zeta^k\right\|^3\right) \end{aligned} \tag{12}$$

where the term with an order higher than two is discarded. With this expansion, we could obtain a quadratic cost function that enables us to decrease the cost function locally via linear optimal control. For simplification, we transform the above equation to

$$\begin{aligned} V^k\left(g^k_\epsilon\right) &= V^k\left(\bar{g}^k\right) + \mathcal{V}^k_g\left(\bar{g}^k\right)\left(\zeta^k\right) \\ &\quad + \frac{1}{2}\mathcal{V}^k_{gg}\left(\bar{g}^k\right)\left(\zeta^k\right)\left(\zeta^k\right) + O\left(\left\|\zeta^k\right\|^3\right) \end{aligned} \tag{13}$$

with

$$\begin{aligned} \mathcal{V}^k_g\left(g^k\right) &:= T_eL^*_{g^k} \circ DV^k\left(g^k\right) \\ \mathcal{V}^k_{gg}\left(g^k\right) &:= T_eL^*_{g^k} \circ \mathrm{Hess}^{(0)}V^k\left(g^k\right) \circ T_eL_{g^k} \end{aligned}$$

Similarly, the second-order expansion of the stage cost could be expressed as:

$$\begin{aligned} &\Lambda^k\left(g^k_\epsilon, u^k_\epsilon\right) \approx \Lambda^k\left(\bar{g}^k, \bar{u}^k\right) \\ &+ \left\langle \ell^k_g\left(\bar{g}^k, \bar{u}^k\right), \zeta^k\right\rangle + \left\langle \ell^k_u\left(\bar{g}^k, \bar{u}^k\right), \delta u^k\right\rangle \\ &+ \frac{1}{2}\left(\left\langle \ell^k_{gg}\left(\bar{g}^k, \bar{u}^k\right)\left(\zeta^k\right), \zeta^k\right\rangle + \left\langle \ell^k_{gu}\left(\bar{g}^k, \bar{u}^k\right)\left(\zeta^k\right), \delta u^k\right\rangle \right.\\ &+ \left\langle \ell^k_{ug}\left(\bar{g}^k, \bar{u}^k\right)\left(\delta u^k\right), \zeta^k\right\rangle \\ &+ \left.\left\langle \ell^k_{uu}\left(\bar{g}^k, \bar{u}^k\right)\left(\delta u^k\right), \delta u^k\right\rangle\right) (9) \end{aligned} \tag{14}$$

with

$$\begin{aligned} \ell^k_g\left(g^k, u^k\right) &:= T_eL^*_{g^k} \circ \mathrm{D}_g\Lambda^k\left(g^k, u^k\right) \\ \ell^k_u\left(g^k, u^k\right) &:= \mathrm{D}_u\Lambda^k\left(g^k, u^k\right) \\ \ell^k_{uu}\left(g^k, u^k\right) &:= \mathrm{D}^2_u\Lambda^k\left(g^k, u^k\right), \\ \ell^k_{gu}\left(g^k, u^k\right) &:= \mathrm{D}_g\mathrm{D}_u\Lambda^k\left(g^k, u^k\right) \circ T_eL^k_g \\ \ell^k_{ug}\left(g^k, u^k\right) &:= T_eL^*_{g^k} \circ \mathrm{D}_u\mathrm{D}_g\Lambda^k\left(g^k, u^k\right) \end{aligned} \tag{15}$$

Then, we incorporate the perturbed state dynamics (9) to (13) and we have the following form:

$$\begin{aligned} &V^{k+1}\left(g^{k+1}_\epsilon\right) \\ &\approx V^{k+1} + \left\langle\left(\Phi^k\right)^* \circ \mathcal{V}^{k+1}_g, \zeta^k\right\rangle \\ &+ \left\langle\left(\mathrm{B}^k\right)^* \circ \mathcal{V}^{k+1}_g, \delta u^k\right\rangle \\ &+ \frac{1}{2}\left(\left\langle\left(\Phi^k\right)^* \circ \mathcal{V}^{k+1}_{gg} \circ \Phi^k\left(\zeta^k\right) + \mathcal{V}^{k+1}_g \circ \Theta^k\left(\zeta^k\right), \zeta^k\right\rangle\right. \\ &+ \left\langle\left[\left(\Phi^k\right)^* \circ \mathcal{V}^{k+1}_{gg} \circ \mathrm{B}^k\left(\delta u^k\right) + \mathcal{V}^{k+1}_g \circ \Delta^k\left(\delta u^k\right), \zeta^k\right\rangle \\ &+ \left\langle\left[\left(\mathrm{B}^k\right)^* \circ \mathcal{V}^{k+1}_{gg} \circ \Phi^k\left(\zeta^k\right) + \mathcal{V}^{k+1}_g \circ \Gamma^k\left(\zeta^k\right), \delta u^k\right\rangle \\ &+ \left.\left\langle\left[\left(\mathrm{B}^k\right)^* \circ \mathcal{V}^{k+1}_{gg} \circ \mathrm{B}^k\left(\delta u^k\right) + \mathcal{V}^{k+1}_g \circ \Xi^k\left(\delta u^k\right), \delta u^k\right\rangle\right) \end{aligned} \tag{16}$$

Now we define the state-action function as:

$$Q^k\left(g^k, u^k\right) := \Lambda^k\left(g^k, u^k\right) + V^{k+1}\left(g^{k+1}\right). \tag{17}$$

Then we could use the perturbed function (14) and (16) to approximate the $Q$ function as a quadratic function:

$$\begin{aligned} \tilde{Q}^k\left(g^k_\epsilon, u^k_\epsilon\right) &:= Q^k_0 + \left\langle Q^k_g, \zeta^k\right\rangle + \left\langle Q^k_u, \delta u^k\right\rangle \\ &\quad + \frac{1}{2}\left(\left\langle Q^k_{gg}\left(\zeta^k\right), \zeta^k\right\rangle\right. \\ &\quad + \left\langle Q^k_{ug}\left(\delta u^k\right), \zeta^k\right\rangle + \left\langle Q^k_{gu}\left(\zeta^k\right), \delta u^k\right\rangle \\ &\quad + \left.\left\langle Q^k_{uu}\left(\delta u^k\right), \delta u^k\right\rangle\right) \end{aligned} \tag{18}$$

so that $\tilde{Q}^k\left(g^k_\epsilon, u^k_\epsilon\right) \approx Q^k\left(g^k_\epsilon, u^k_\epsilon\right)$ and

$$\begin{aligned} Q^k_0 &:= \Lambda^k + V^{k+1} \\ Q^k_g &:= \ell^k_g + \left(\Phi^k\right)^* \circ \mathcal{V}^{k+1}_g, \quad Q^k_u := \ell^k_u + \left(\mathrm{B}^k\right)^* \circ \mathcal{V}^{k+1}_g \\ Q^k_{gg} &:= \ell^k_{gg} + \left(\Phi^k\right)^* \circ \mathcal{V}^{k+1}_{gg} \circ \Phi^k + \left(\mathcal{V}^{k+1}_g\right)^{\mathfrak{g}} \circ \Theta^k \\ Q^k_{gu} &:= \ell^k_{gu} + \left(\mathrm{B}^k\right)^* \circ \mathcal{V}^{k+1}_{gg} \circ \Phi^k + \left(\mathcal{V}^{k+1}_g\right)^m \circ \Gamma^k \\ Q^k_{ug} &:= \ell^k_{ug} + \left(\Phi^k\right)^* \circ \mathcal{V}^{k+1}_{gg} \circ B^k + \left(\mathcal{V}^{k+1}_g\right)^{\mathfrak{g}} \circ \Delta^k \\ Q^k_{uu} &:= \ell^k_{uu} + \left(B^k\right)^* \circ \mathcal{V}^{k+1}_{gg} \circ B^k + \left(\mathcal{V}^{k+1}_g\right)^m \circ \Xi^k \end{aligned} \tag{19}$$

Attributed to the use of the exponential functor, we can drop all arguments from $\Theta^k$, $\Gamma^k$, $\Delta^k$, and $\Xi^k$.

### D. LOCAL OPTIMAL CONTROL POLICY

Based on the $Q$ function, we can apply Bellman's principle to derive the optimal local policy via solving the following quadratic programming (QP ) [9]:

$$V^k\left(g^k_\epsilon\right) = \min_{\delta u^k}\left[\tilde{Q}^k\left(g^k_\epsilon, u^k_\epsilon\right)\right] \tag{20}$$

By the quadratic expansion, we have:

$$\begin{aligned} \delta u^k_\star &= -\left(Q^k_{uu}\right)^{-1} \circ Q^k_u - \left(Q^k_{uu}\right)^{-1} \circ Q^k_{gu}\left(\zeta^k\right) \\ &= \delta u^k_f + K^k(\zeta^k) \end{aligned} \tag{21}$$

and the optimal policy becomes $u^k = \bar{u}^k + \delta u^k$.

### E. VALUE FUNCTION UPDATE

When the optimal control policy is accessible, we could iteratively update the value function as:

$$
\begin{aligned}
& V^k + \left\langle \mathcal{V}_g^k, \zeta^k \right\rangle + \frac{1}{2}\left\langle \mathcal{V}_{gg}^k\left(\zeta^k\right), \zeta^k \right\rangle + O\left(\left\|\zeta^k\right\|^3\right) \\
&= Q_0^k + \left\langle Q_g^k, \zeta^k \right\rangle + \left\langle Q_u^k, -\left(Q_{uu}^k\right)^{-1} \circ Q_u^k \right. \\
&\quad \left. - \left(Q_{uu}^k\right)^{-1} \circ Q_{gu}^k\left(\zeta^k\right)\right\rangle \\
&\quad + \left\langle Q_{gu}^k\left(\zeta^k\right), -\left(Q_{uu}^k\right)^{-1} \circ Q_u^k - \left(Q_{uu}^k\right)^{-1} \circ Q_{gu}^k\left(\zeta^k\right)\right\rangle \\
&\quad + \frac{1}{2}\left\langle Q_{gg}^k\left(\zeta^k\right), \zeta^k \right\rangle \\
&\quad + \frac{1}{2}\left\langle Q_{uu}^k \circ \left(Q_{uu}^k\right)^{-1} \circ Q_u^k, \left(Q_{uu}^k\right)^{-1} \circ Q_u^k \right\rangle \\
&\quad + \left\langle Q_{uu}^k \circ \left(Q_{uu}^k\right)^{-1} \circ Q_u^k, \left(Q_{uu}^k\right)^{-1} \circ Q_{gu}^k\left(\zeta^k\right)\right\rangle \\
&\quad + \frac{1}{2}\left\langle Q_{uu}^k \circ \left(Q_{uu}^k\right)^{-1} \circ Q_{gu}^k\left(\zeta^k\right), \left(Q_{uu}^k\right)^{-1} \circ Q_{gu}^k\left(\zeta^k\right)\right\rangle
\end{aligned} \tag{22}
$$

Since the above result holds for arbitrary $\zeta^k$, we can match the first and second-order terms. After a simple manipulation, we obtain the following expressions:

$$
\begin{aligned}
\mathcal{V}_g^k &= Q_g^k - Q_{ug}^k \circ \left(Q_{uu}^k\right)^{-1} \circ Q_u^k \\
\mathcal{V}_{gg}^k &= Q_{gg}^k - Q_{ug}^k \circ \left(Q_{uu}^k\right)^{-1} \circ Q_{gu}^k
\end{aligned}
$$

Recall that all quantities above are evaluated at $\bar{\tau}_0^N$, with the right-hand sides depending on $\mathcal{V}_g^{k+1}$ and $\mathcal{V}_{gg}^{k+1}$. The final condition for this backpropagation scheme is given by

$$
\begin{aligned}
\mathcal{V}_g^N\left(\bar{g}^N\right) &= T_e L_{\bar{g}^N}^* \circ \mathrm{D}F\left(\bar{g}^N\right) \\
\mathcal{V}_{gg}^N\left(\bar{g}^N\right) &= T_e L_{\bar{g}^N}^* \circ \mathrm{Hess}^{(0)} F\left(\bar{g}^N\right) \circ T_e L_{\bar{g}^N}
\end{aligned}
$$

### F. FORWARD PASS

In the backward pass, we have obtained the local policy by computing the matrix inversion induced from Bellman's principle and updating the value function iteratively. In the forward pass, we apply the optimal policy and integrate a new trajectory:

$$
\begin{aligned}
\zeta^k &= \log((g^k)^{-1}\bar{g}^k) \\
g^{k+1} &= f(g^k, \bar{u}^k + \delta u_f^k + K^k(\zeta^k))
\end{aligned} \tag{23}
$$

with the initial condition $\zeta^0 = 0$.

With the above analysis, the unconstrained DDP algorithm could be summarized as follows: When the difference between the consecutive trajectory or inputs is below a threshold, we say that the algorithms converge.

## IV. CONSTRAINED DDP ON LIE GROUP

In this section, we consider the DDP problems with general nonlinear constraints that can handle the safety conditions.

**Algorithm 1** DDP on Lie Group

```
1:  Initialize trajectory $\tau_0^N$
2:  while not converge do
3:      Initialize the value function via (13) at time step $N$.
4:      for $k \in (N-1, \dots, 0)$ do                ▷ Backward pass
5:          Compute the $Q$ function by (19).
6:          Compute the optimal policy by (21).
7:          Update the value function by (22).
8:      end for
9:      Store the policy as $\mathcal{P}_0^{N-1}$.
10:     for $k \in (0, \dots, N-1)$ do                ▷ Forward pass
11:         Integrate the trajectory by dynamics using policy $\mathcal{P}$.
12:     end for
13:     Store the trajectory as $\tau_0^N$.
14: end while
15: return $\tau_0^N$ and $\mathcal{P}_0^{N-1}$.
```

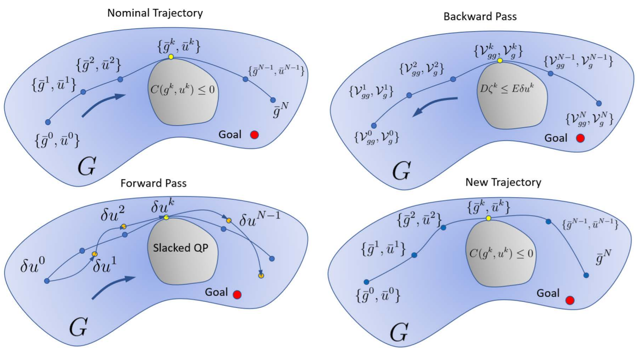
Nominal Trajectory
$\{\bar{g}^k, \bar{u}^k\}$
$\{\bar{g}^2, \bar{u}^2\}$
$\{\bar{g}^1, \bar{u}^1\}$
$\{\bar{g}^{N-1}, \bar{u}^{N-1}\}$
$C(g^k, u^k) \leq 0$
$\bar{g}^N$
$\{\bar{g}^0, \bar{u}^0\}$
Goal
$G$
Backward Pass
$\{\mathcal{V}_{gg}^k, \mathcal{V}_g^k\}$
$\{\mathcal{V}_{gg}^2, \mathcal{V}_g^2\}$
$\{\mathcal{V}_{gg}^{N-1}, \mathcal{V}_g^{N-1}\}$
$\{\mathcal{V}_{gg}^1, \mathcal{V}_g^1\}$
$D\zeta^k \leq E\delta u^k$
$\{\mathcal{V}_{gg}^N, \mathcal{V}_g^N\}$
$\{\mathcal{V}_{gg}^0, \mathcal{V}_g^0\}$
Goal
$G$
Forward Pass
$\delta u^2$
$\delta u^k$
$\delta u^{N-1}$
$\delta u^0$
$\delta u^1$
Slacked QP
Goal
$G$
New Trajectory
$\{\bar{g}^k, \bar{u}^k\}$
$\{\bar{g}^2, \bar{u}^2\}$
$\{\bar{g}^{N-1}, \bar{u}^{N-1}\}$
$\{\bar{g}^1, \bar{u}^1\}$
$C(g^k, u^k) \leq 0$
$\bar{g}^N$
$\{\bar{g}^0, \bar{u}^0\}$
Goal
$G$

**FIGURE 2. Constrained Differential Dynamic Programming on Lie group. When the nonlinear constraint is considered, the DDP backward pass incorporates the active set to determine the optimal control law. By considering the active set, the control law would not violate the inequality constraints. In the forward pass, the DDP solves a slacked QP to integrate the trajectory that can avoid infeasibility due to constraints.**

The main process of DDP with general nonlinear constraints is illustrated in Fig. 2.

### A. PROBLEM FORMULATION

Consider the system dynamics on Lie group $G$ in (5). We consider the input saturation and define the safe set as

$$h(g^k) \geq 0.$$

Thus, we have the discrete-time, finite-horizon constrained optimal control problem of the following form:

$$
\begin{aligned}
\min_{\{u^k\}_0^{N-1}} \quad & F(g_N) + \sum_{k=0}^{N-1} \Lambda(g^k, u^k) \\
\text{s.t. } & g^{k+1} = f(g^k, u^k) \\
& h(g^k) \geq 0 \\
& u^k \in \mathcal{U}^k, \quad g^0 = g(0) \\
& k = 0, 1, \dots, N-1.
\end{aligned} \tag{24}
$$

where $\mathcal{U}^k$ is the input constraints at time step $k$. Compared to the unconstrained case, the input saturation and the nonlinear constraints are considered. Thus we could not directly apply the matrix inversion to compute the optimal inputs as we do in the unconstrained case. To be compatible with the notations in the nonlinear optimization, we use $C(\cdot) \leq 0$ to denote any nonlinear constraints.

The local optimal policy for unconstrained case are similar to optimizing (20). However, additional constraints are needed to take into account the nonlinear condition. Thus, the value function for the constrained case can be expressed as:

$$\begin{aligned} &\min_{\delta u^k} \tilde{Q}^k\left(g_\epsilon^k, u_\epsilon^k\right) \\ &\text{s.t. } C(g_\epsilon^k, u_\epsilon^k) \leq 0. \end{aligned} \tag{25}$$

As the inequality constraints are taken into account, the optimal local policy can no longer be expressed by a simple matrix inversion as we do in the unconstrained case.

To solve this problem, we apply the active set method introduced in [42]. The idea is that we determine the active constraints each time we compute (26) in the backward pass. As the constraints are generally nonlinear, a linear expansion of the active constraints is required:

$$\begin{aligned} &\min_{\delta u^k} \tilde{Q}^k\left(g_\epsilon^k, u_\epsilon^k\right) \\ &\text{s.t. } D\zeta^k = E\delta u^k \end{aligned} \tag{26}$$

where the $D$ and $E$ matrices are selected from the active constraints, such that $C(g^k, u^k) \leq -\epsilon$. We choose a small positive constant $\epsilon$ to avoid the numerical issues. Note that for the general case, the constraints are assumed to be inequality constraints:

$$\begin{aligned} &\min_{\delta u^k} \tilde{Q}^k\left(g_\epsilon^k, u_\epsilon^k\right) \\ &\text{s.t. } D\zeta^k \leq E\delta u^k. \end{aligned} \tag{27}$$

However, in the active set method, we have set the linear inequality constraints strictly satisfied as the active set is determined. The problem (27) will assist us in determining the active set in the following section.

### B. EXPANSION OF NONLINEAR CONSTRAINTS

As the dynamics need to incorporate into the cost function, a second-order expansion is necessary. However, only the first-order approximation is needed in the nonlinear constraints expansion. Thus, we have:

$$\tilde{h}^k\left(g_\epsilon^k, u_\epsilon^k\right) := h_0^k + \left\langle h_g^k, \zeta^k\right\rangle + \left\langle h_u^k, \delta u^k\right\rangle \tag{28}$$

where

$$\begin{aligned} h_g^k\left(g^k, u^k\right) &:= T_e L_{g^k}^* \circ \mathrm{D}_g h^k\left(g^k, u^k\right) \\ h_u^k\left(g^k, u^k\right) &:= \mathrm{D}_u h^k\left(g^k, u^k\right) \\ h_0^k &= h^k(g^k, u^k). \end{aligned} \tag{29}$$

Note that it is also possible to apply a second-order approximation of the nonlinear function. But there may not be a guarantee that the Hessian matrix remains positive definite, which will result in a nonlinear and nonconvex optimization that is hard to solve.

### C. BACKWARD PASS

In the backward pass, the optimal policy needs to take into account the additional inequality constraints [42]. For optimization problem (27) that includes additional inequality constraints, by KKT condition [9], we could introduce additional multiplier $\lambda$ and convert the (27) to its dual problem:

$$\begin{aligned} &\min_{\delta u^k} \tilde{Q}^k\left(g_\epsilon^k, u_\epsilon^k\right) + \lambda^T(D\zeta^k - E\delta u^k) \\ &s.t\ \lambda \geq 0 \quad \text{(dual feasibility)}, \\ &\quad \lambda^T(D\zeta^k - E\delta u^k) = 0 \quad \text{(complementary condition)}, \\ &\quad D\zeta^k - E\delta u^k \leq 0 \quad \text{(primal feasibility)}, \end{aligned} \tag{30}$$

where we have the dual feasibility, such that the multiplier $\lambda$ is non-negative, the complementary condition, and the primal feasibility. For the active set method, as we have already determined the active equality constraints, we have that $D\zeta^k - E\delta u^k = 0$. Thus, one straight forward approach is to reduce the KKT condition (30) to the following form:

$$\begin{bmatrix} Q_{uu}^k & D^T \\ D & 0 \end{bmatrix} \begin{bmatrix} \delta u^k \\ \lambda \end{bmatrix} = -\begin{bmatrix} Q_{ug}^k \\ E \end{bmatrix} \zeta_k - \begin{bmatrix} Q_u^k \\ 0 \end{bmatrix} \tag{31}$$

However, simply applying the (31) can not update the active set because the solution itself assumes the satisfaction of the predefined active set.

Therefore, we do not apply the (31) to compute the multiplier $\lambda$ and control input $\delta u^k$ simultaneously, but compute the $\delta u^k$ around the trajectory and examine the multiplier to check if it satisfies the active set constraints, which will enable us to update the active sets in the backward path. As is in the backward path, the nominal trajectory is selected. Thus we assume all the $\zeta^k$ remain unchanged to compute the $\delta u^k$. Note that $\tilde{Q}^k\left(g_\epsilon^k, u_\epsilon^k\right)$ is a quadratic function, we convert (26) to the following form with fixed $\zeta^k$:

$$\begin{aligned} &\min_{\delta u^k} \frac{1}{2}\delta u^{k,T} Q_{uu}^k \delta u^k + Q_u^k \delta u^k \\ &\text{s.t. } E\delta u^k = 0. \end{aligned} \tag{32}$$

Remember that for the inequality constraints case, only active constraints will lead to positive multipliers because of the complementary condition [9]. Therefore, we solve (32) to obtain the $\delta u^k$ and then compute the multipliers by substituting the solution of $\delta u^k$ to (30). Then we remove the constraints with negative multipliers from the active set. We denote the updated constraints matrix as $\hat{D}$ and $\hat{E}$.

Now we could obtain the optimal policy using the KKT condition with updated constraints $\hat{D}$ and $\hat{E}$:

$$\delta u^k = K^k \xi^k + v^k. \tag{33}$$

where

$$\begin{aligned} K_k &= -H \circ Q^k_{ug} + W^* \circ \hat{D}^k, \\ v_k &= -H \circ Q_{u,k}, \\ W &= \left(\hat{C}^k \circ (Q^k_{uu})^{-1} \circ (\hat{C}^k)^T\right)^{-1} \circ \hat{C}^k \circ (Q^k_{uu})^{-1}, \\ H &= Q^{-1}_{uu,k} \circ \left(I - (\hat{C}^k)^T \circ W\right) \end{aligned} \tag{34}$$

By iteratively updating the active set, we could obtain the optimal control policy for the constrained problem. The backward path could be summarized in Algorithm 2.

**Algorithm 2** Constraint DDP Backward Pass

```
1: for $k \in (N-1, \dots, 0)$ do
2:     Compute the second-order expansion of the $Q$ function by (19).
3:     Determine the active set ($D$ and $E$ matrix) by checking $C(g^k, u^k) \geq -\epsilon$.
4:     Solve the dual variables via Problem (31).
5:     Update the active set ($\hat{D}$ and $\hat{E}$ matrix).
6:     Compute the optimal policy by (33).
7:     Update the value function by (22).
8: end for
9: return the optimal policy.
```

### D. FORWARD PASS

In the forward pass, we have to ensure that the updated nominal trajectory remains feasible and the action is consistently reducing the cost. As the $\zeta$ is only accessible in the forward pass but remains unknown in the backward path, there is no guarantee that the optimal policy obtained in the backward path makes the forward path feasible and optimal. Therefore, we apply a QP solver in the forward path to integrate the solution:

$$\begin{aligned} &\min_{\delta u^k} \frac{1}{2}\delta u^{k,T} Q^k_{uu} \delta u^k + \delta u^T Q^k_{ux} \zeta^k + Q^{k,T}_k \delta u^k \\ &\text{s.t } C^k\left(g^k, u^k\right) + C^k_g\left(g^k, u^k\right)\zeta^k + C^k_u\left(g^k, u^k\right)\delta u^k \leq 0 \end{aligned} \tag{35}$$

However, if it is possible that the QP may not be feasible, thus we introduce a slack variable $\eta \geq 0$ and form the following problem:

$$\begin{aligned} &\min_{\delta u^k} \frac{1}{2}\delta u^{k,T} Q^k_{uu} \delta u^k + \delta u^{k,T} Q^k_{ux} \zeta^k + Q^{k,T}_k \delta u^k + w\|\eta\|^2 \\ &\text{s.t } C^k\left(g^k, u^k\right) + C^k_g\left(g^k, u^k\right)\zeta^k + C^k_u\left(g^k, u^k\right)\delta u^k \leq \eta \\ &\quad \eta \geq 0 \end{aligned} \tag{36}$$

where $w$ is a large number that penalizes the slack variables. By introducing the slack variable, there is no need to iteratively shrink the feasible region as is introduced in [42] to ensure feasibility in the forward pass.

**Algorithm 3** Constraint DDP Forward Pass

```
1: Given nominal trajectory $\tau_0^N$
2: for $k \in (0, \dots, N-1)$ do
3:     Compute $\zeta^k = \log(g^{-1}\bar{g}^k)$.
4:     Compute the optimal control problem (36) to obtain $\delta u$.
5:     Integrate the trajectory by the dynamics.
6: end for
7: return the optimal trajectory $\tau_0^N$
```

## V. EXPERIMENTS

In this section, we aim to plan a trajectory on SO(3) with pose constraints. We consider the motion on SO(3) and wish to avoid certain configurations that are thought unsafe. one practical usage of this task is to prevent the camera on a drone from pointing to the sun, which might make it lose track of landmarks.

### A. DYNAMICS ON SO(3)

In this section, we introduce the dynamics on SO(3). We consider the rotational motion of a rigid body. The continuous equation of motion is given by:

$$\begin{aligned} \dot{R} &= R\omega^\times \\ \dot{\omega} &= \mathbb{I}^{-1}\left((\mathbb{I}\omega) \times \omega + u\right) \end{aligned} \tag{37}$$

where $R \in SO(3)$ is the rotation matrix, $\omega \in \mathbb{R}^3$ is the velocity in the body frame, and the $u \in \mathbb{R}^3$ is the input, i.e., generalized force acted on the body frame. We use the $\mathbb{I}$ to represent the inertial matrix in the body frame.

To apply the DDP algorithm in discrete time, we apply the Euler first-order method to obtain the discrete dynamics as follows:

$$\begin{aligned} R^{k+1} &= R^k \exp\left(\hat{\omega}^k \Delta t\right) \\ \omega^{k+1} &= \omega^k + \Delta t \mathbb{I}^{-1}\left((\mathbb{I}\omega^k) \times \omega^k + u^k\right) \end{aligned} \tag{38}$$

where $\hat{(\cdot)}$ is to map the element in $\mathbb{R}^n$ space to the corresponding Lie algebra. To map the element from Lie algebra to vector space, we use the notation $(\cdot)^\vee$. Note that the equation of motion of the $\omega$ is described in a vector space. Thus we could linearize the equation of motion as we do in conventional DDP.

The expansion of the dynamics of the pose, which is denoted as $f$, can be expressed as follows:

$$\begin{aligned} T_e L^*_{R^k} \circ \mathrm{D}_R\left(f^k_\omega\right)_i &= 0_{1\times 3}, \quad \mathrm{D}_u\left(f^k_\omega\right) = \Delta t \mathbb{I}^{-1} \\ \mathrm{D}_\omega\left(f^k_\omega\right) &= I_3 + \Delta \mathbb{I}^{-1}(-\hat{\omega}\mathbb{I} + \widehat{\mathbb{I}\omega}) \\ \mathrm{D}_{\omega_j}\mathrm{D}_{\omega_i}\left(f^k_\omega\right) &= \Delta t \mathbb{I}^{-1}\left(\left(\widehat{\mathbb{I}e_i}\right)e_j + \left(\widehat{\mathbb{I}e_j}\right)e_i\right) \\ \Theta^k_{RR(i)} &= 0_{6\times 6}, \quad \Theta^k_{R\omega(i)} = 0_{6\times 6}, \quad \text{for all } i > 3 \\ \Gamma^k_{(i)} &= 0_{6\times m}, \quad \Delta^k_{(i)} = 0_{m\times 6}, \\ \Xi^k_{(i)} &= 0_{m\times m}, \quad \text{for all } i \end{aligned} \tag{39}$$

**TABLE 1. Simulation parameters.**

| | $S_R$ | $S_\omega$ | $S_v$ | $S_b$ | $b$ |
|---|---|---|---|---|---|
| Value | $1000I_3$ | $I_3$ | $0.01I_3$ | $I_3$ | Adaptive |

where the subscripts $i, k$ denote the components of the quantities. The detailed procedure of computing these terms is referred to [41]. The dynamics of $\omega$ are in vector space, where we could apply the conventional Jacobian linearization as in conventional nonlinear controller design.

To indicate the configuration difference on the manifold, we consider the weighted Frobenius norm defined as follows:

$$\|A\|_{m,S} := \sqrt{\operatorname{trace}(A^\top SA)}, \tag{40}$$

where $S$ is a positive-definite matrix and $S, A \in \mathbb{R}^{n\times n}$ The weighted Euclidean norm is denoted as:

$$\|x\|_{v,S} := \sqrt{x^\top Sx}, \tag{41}$$

where $S$ is a positive-definite matrix, we use the Frobenius norm to indicate the distance between two configurations on SO(3).

## B. TRAJECTORY OPTIMIZATION WITH POSE CONSTRAINTS

We consider a trajectory optimization problem with pose constraints. We show that the proposed controller could plan a trajectory on SO(3) that can avoid certain unsafe poses. For the problem (24), we consider the terminal cost $F$, stage cost $\Lambda$ and safe set $h$ of the following form:

$$\begin{aligned} F(R^N, \omega^N) &:= \frac{1}{2}\|I_3 - R_d^\top R^N\|^2_{m,S_R} + \frac{1}{2}\|\omega^N\|^2_{v,S_\omega}, \\ \Lambda(R, \omega, u^k) &:= \frac{1}{2}\|u^k\|^2_{v,S_u}, \\ h(R^k) &:= \|I_3 - R_s^\top R^k\|^2_{m,S_b} - b^2. \end{aligned} \tag{42}$$

We do not add a penalty for the stage cost and only consider the terminal cost for configurations. The constraint is designed to ensure that the rigid body will never approach the neighborhood of the unsafe configuration $R_s$. The neighborhood is defined by a constant $b > 0$.

As we are solving the trajectory optimization problem on SO(3), the property of trace function and skew matrices are very useful in determining the local coordinates and the matrix expansions. We define the following notations before we give the specific form of the Jacobian and Hessian matrix of the value functions. $\operatorname{sym}(N) := \frac{1}{2}\left(N + N^\top\right)$, $\operatorname{skew}(N) := \frac{1}{2}(N - N^\top)$, $\zeta_i = (\eta_i, \rho_i) \in \mathfrak{so}(3) \times \mathbb{R}^3$, $\operatorname{trace}(\operatorname{sym}(N)\operatorname{skew}(M)) = 0$, $\operatorname{trace}\left(\hat{x}^\top S\right) = 2x^\top(\operatorname{skew}(S))^\vee$, $\operatorname{trace}(\hat{x}^\top S\hat{y}) = y^\top(\operatorname{trace}(S)I_3 - S)\,x$, for all $N, M \in \mathbb{R}^{n\times n}, x, y \in \mathbb{R}^3,\ S \in \mathbb{R}^{3\times 3}$.

Thus we have the first order term of the terminal cost as:

$$\begin{aligned} &\left\langle \mathrm{D}F\left(g^N\right), g^N\zeta\right\rangle \\ &\quad = \left(\left[2\operatorname{skew}\left(S_R R_d^\top R^N\right)\right]^\vee\right)^\top [\eta]^\vee \\ &\qquad + \left(S_\Omega\left(\Omega^N - \Omega_d\right)\right)^\top \rho, \end{aligned} \tag{43}$$

and the Hessian of the terminal cost can be expressed by:

$$\begin{aligned} &\left\langle \operatorname{Hess}^{(0)} F\left(g^N\right)\left(g^N\zeta_1\right), g^N\zeta_2\right\rangle \\ &\quad = \left(\eta_1^\vee\right)^\top\left(\operatorname{sym}\left(\operatorname{trace}\left[S_R R_d^\top R^N\right] I_3 - S_R R_d^\top R^N\right)\right)\eta_2^\vee \\ &\qquad + \rho_1^\top S_\Omega \rho_2 \end{aligned} \tag{44}$$

As the nonlinear constraints also consider the pose constraints as the terminal cost, the first-order expansion of the terminal cost can also be applied to the expansion of the nonlinear constraints.

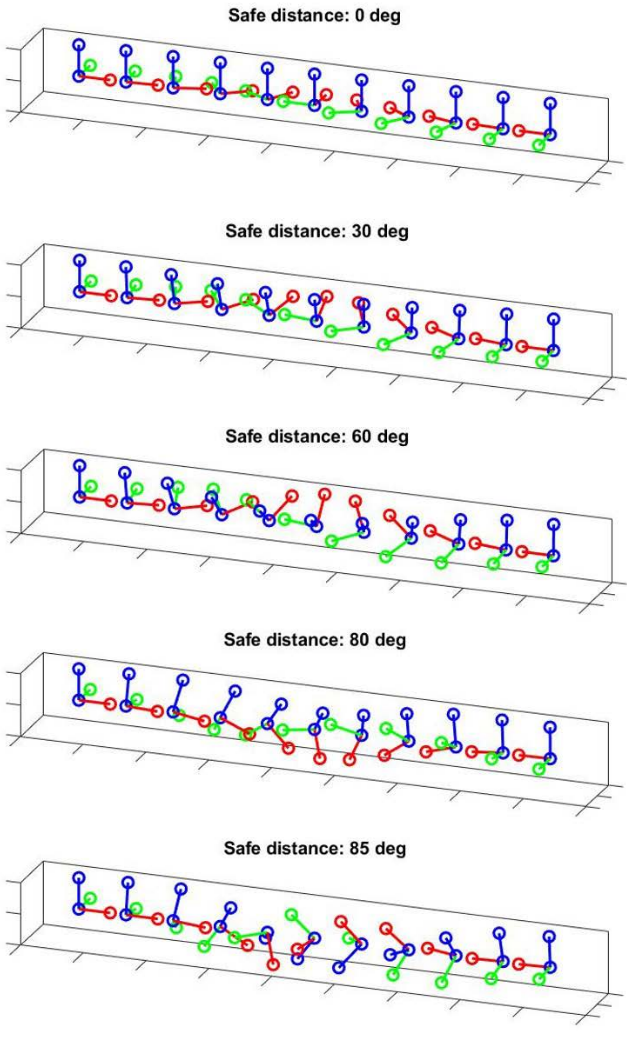

**FIGURE 3. Rigid body motion with a different safe distance. When the safe distance is set to 0, the rigid body directly rotates to the final pose and goes across the unsafe pose. When the nonlinear constraints are taken into consideration, the rigid body starts to move on the other axis and avoid the unsafe pose.**

## C. NUMERICAL RESULT

We now implement the proposed algorithm to plan a safe path for SO(3) rigid body. We use $R_x(t), R_y(t), R_z(t)$ to denote the

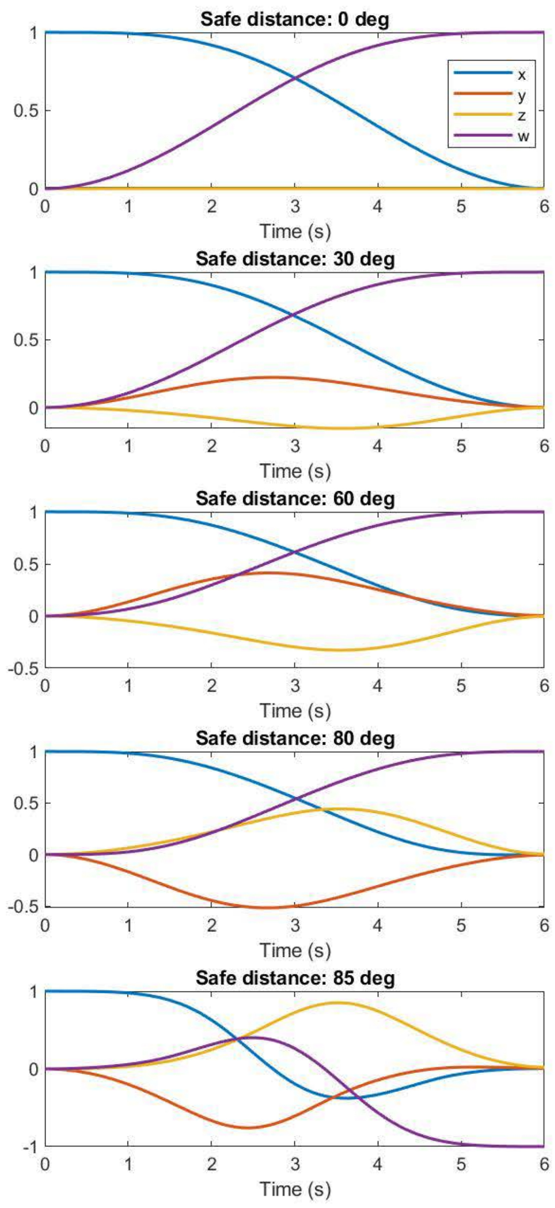

**FIGURE 4.** Trajectory represented by quaternions. When the safe distance is set to 0, only *z* motion is involved. When the safe distance is set, the motion in other axes is observed.

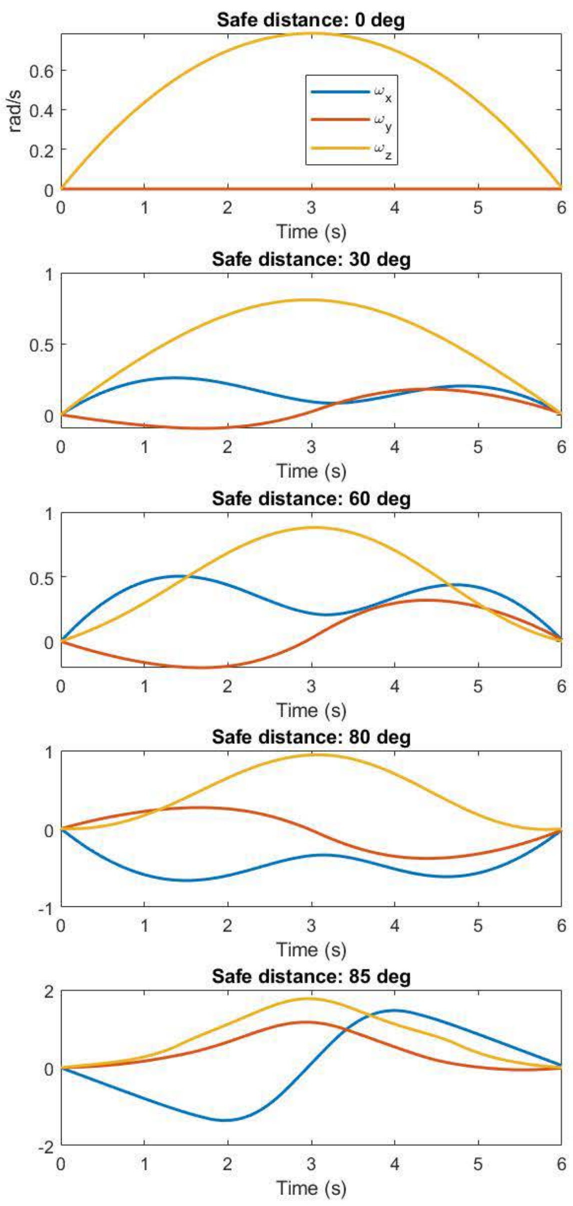

**FIGURE 5.** Angular velocity of the planned trajectory. When the safe distance is set to 0, only angular velocity in *z* axis is observed. When the safe distance is set, the angular velocity in the other axes gets large, which enables the robot to avoid the unsafe pose.

rotation around the $x$, $y$ and $z$ axis of body-fixed frame for $t$ degree.

We consider to rotate the rigid body to configuration $R_z(180)$ from the identity and avoid the unsafe configuration $R_s = R_z(90)$. We select the constant $b$ such that the rigid body will keep a distance with $R_s$ for 30, 60, 80, and 85 degrees. We also consider the case without the safe set and use it as the initialization for the other cases. The specific parameters are listed in Table. 1.

The planned trajectory represented by the body-fixed axes is presented in time series as in Fig. 3. The trajectory depicted in quaternion is presented in Fig. 4. The corresponding angular velocities are presented in Fig. 5. The case with safe distances being 30, 60, 80, and 85 degrees and without safe sets are presented.

In Fig. 3, the rigid body configuration starts at the identity from the left-hand side and moves to the $R_z(180)$ at the right-hand side while trying to avoid $R_s$ with different buffers. Without the nonlinear safe constraints, the rigid body only rotates about its $z$ axis. Thus it will certainly go across the unsafe configuration $R_s$. When we add safe distance by incorporating the nonlinear constraints, we can see the involvement of rotation in $x$ and $y$ axis. With the motion in $x$ and $y$ axis, the robot could avoid the unsafe motions $R_s$. When the safe distance becomes larger, we can see that the magnitude of motion in $x$ and $y$ axes become larger in order

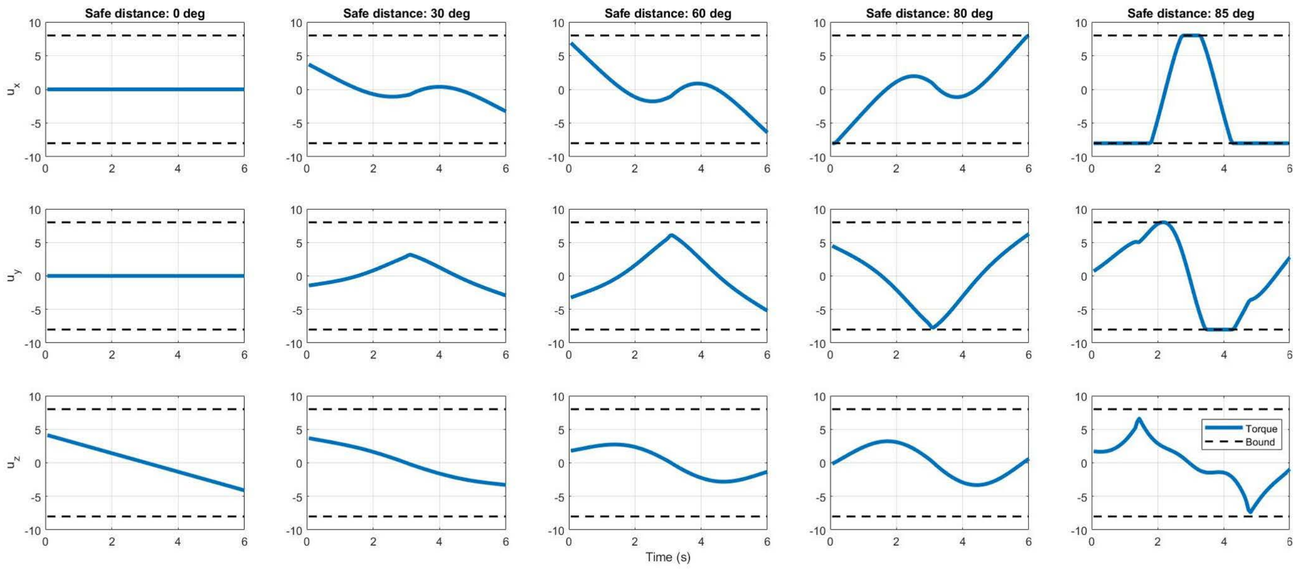

**FIGURE 6.** Control input of planned trajectories. For the case without safe distance, the system acts like a second-order integrator as it is rotating around a single-body principle axis. For the 85-degree safe distance case, the input saturates but does not violate the bounds.

to avoid hitting the boundary of the safe sets. The change in the angular velocity can be seen in Fig. 5.

The inputs are presented in Fig. 6. We can see that the planned inputs do not violate the boundary in the whole planning horizon. With the buffer becoming 85 degrees, more control efforts are needed to move the rigid body. However, the inputs are on the boundary but never violate the constraints.

The value of the safe set function $h$ is presented in Fig. 7. It can be seen that the $h > 0$ for the entire trajectory. When approaching the goal from the identity, the pose goes close to the $R_s$, i.e., $h$ approaches 0 but never hits this pose.

### D. CONVERGENCE RATE AND ROBUSTNESS

We compare the proposed method with the constrained DDP on the manifold in terms of convergence rate and robustness to numerical disturbances.

We consider the DDP on Lie group that adopts the same active set method as [42] as our baseline. We test the case at a safe distance being 60 degrees. To test the robustness of the method, we apply random noise to the feedforward force in each iteration. We launched 200 simulations to obtain the convergence rate, and the result is shown in Fig. 8.

We find that the proposed method has fewer non-convergent cases and a faster convergence rate than the baseline. In the 200 randomly perturbed cases, the proposed method has 6 cases that do not converge, while the non-convergent case of the baseline is 37. The convergence rate of the proposed algorithm is also faster than the baseline method. All the case converges within 20 iterations, while the baseline fails to converge in 20 iterations. The random noise is introduced to model the poor initialization that is widely seen in trajectory optimization. Our result suggests that even though the system trajectory deviates a lot from an initial guess due to the disturbances, the optimization still maintains a high success rate than the baseline methods. Such robustness is essential to trajectory optimization.

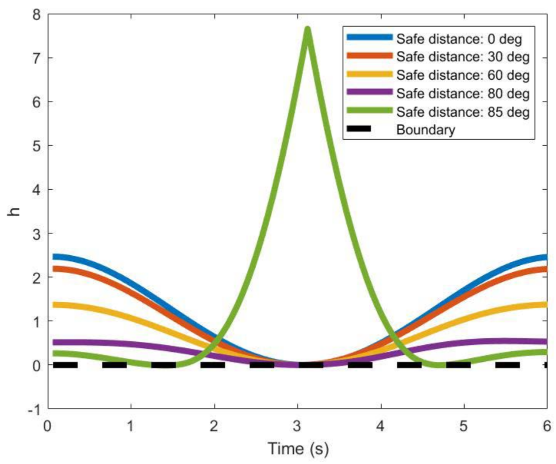

**FIGURE 7.** Value of safe constraints. We can see that the safe constraints always remain feasible. In the case that the safe distance is 30, 60, and 80 degrees, the rigid body reaches the safe distance at the middle of the profile but still remains feasible. For the 85-degree case, there are two configurations that reach the boundary. For the case with a 0-degree safe distance, it certainly approaches 0 at the middle of the profile.

## VI. DISCUSSIONS

In this paper, we have applied the DDP algorithm with nonlinear constraints that address the pose-constrained problems on SO(3) in a coordinate-free manner. Compared to the state-of-the-art method with the active set method, our method has

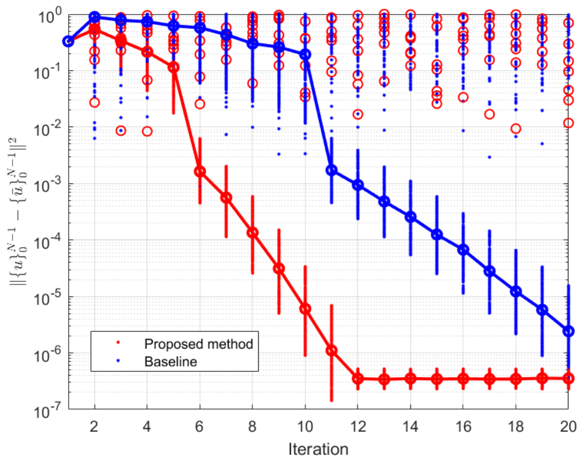

**FIGURE 8. Convergence of proposed method and the DDP method on Lie group with active set in [42]. The solid line indicates the average convergence rate for the converged case. The red dot indicates the non-convergent case of the proposed method. We find that the proposed method has a faster convergence rate and fewer non-convergent cases than the baseline. We select the case with a safe distance being 60 degrees. In the 200 perturbed cases, only 6 cases for the proposed methods do not converge, while the number for the baseline is 37. The results suggest that our methods are more numerically stable and robust than the conventional methods in the presence of perturbations, such as poor initialization.**

faster convergence and higher robustness when the system is perturbed.

The proposed active set-based method has been successfully applied to a discrete-time system. One reason for its success is that it is feasible to determine the active constraints at the backward pass by thresholding the nonlinear constraints. However, the active-set-based method may be hard to apply to continuous-time systems as determining such an active set in the continuous-time domain is not feasible. One possible solution to the continuous-time problem is to apply the log barrier function that is widely adopted in the interior-point methods.

One drawback of the proposed method is that the introduction of the slack variable does not completely solve the infeasibility problem in the forward path. Thus it may still take a few iterations for the trajectory to converge to a feasible solution. This problem has been an open problem for the DDP-based method, which is an interesting direction for future work.

## VII. CONCLUSION

In this paper, we presented a discrete-time Differential Dynamics Programming algorithm on SO(3) for trajectory optimization with pose constraints. We apply the algorithm with Riemannian geometry techniques to handle the parameterization problem on manifold and active set methods for constraints. The distance function is imposed on the manifold as the safety constraint. The active safe set is determined in the backward path to deriving the optimal policy. New slack variables are introduced in the forward pass to tackle the infeasibility problem. Numerical simulation has been conducted to show that the proposed algorithm could guarantee a safe and dynamically feasible trajectory that could avoid unsafe poses for systems on SO(3).

The main advantage of this proposed method is that it solves the constrained optimization problems on the SO(3) manifold in a coordinate-free manner using geometric control techniques. The proposed algorithm is not only capable of handling inequality constraints on the manifolds but also more numerically robust to handle noise and disturbance than the baseline methods.

Based on the aforementioned contributions, certain future topics may need further investigation. It would be interesting to see how to incorporate more complicated constraints, for example, the position constraints related to the rigid body orientation. Incorporating more complicated dynamics into Lie groups, such as the special Euclidean group that also models the translation of the rigid body, can also be considered in future research. On the numerical side, other than a quadratic expansion, a more general convex set expansion may further reduce the burden of removing the active linear constraints. For example, we could determine the convex set as a quadratic constraint set that has more numerical advantages.

## REFERENCES

[1] F. C. Park, J. E. Bobrow, and S. R. Ploen, "A lie group formulation of robot dynamics," *Int. J. Robot. Res.*, vol. 14, no. 6, pp. 609–618, Dec. 1995.

[2] A. Iserles, H. Z. Munthe-Kaas, S. P. Nørsett, and A. Zanna, "Lie-group methods," *Acta Numerica*, vol. 9, pp. 215–365, Jan. 2000.

[3] F. Gao, W. Wu, Y. Lin, and S. Shen, "Online safe trajectory generation for quadrotors using fast marching method and Bernstein basis polynomial," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2018, pp. 344–351.

[4] R. Mahony, V. Kumar, and P. Corke, "Multirotor aerial vehicles: Modeling, estimation, and control of quadrotor," *IEEE Robot. Autom. Mag.*, vol. 19, no. 3, pp. 20–32, Sep. 2012.

[5] D. Mellinger and V. Kumar, "Minimum snap trajectory generation and control for quadrotors," in *Proc. IEEE Int. Conf. Robot. Autom.*, May 2011, pp. 2520–2525.

[6] T. Lee, M. Leok, and N. H. McClamroch, "Geometric tracking control of a quadrotor UAV on SE(3)," in *Proc. 49th IEEE Conf. Decis. Control (CDC)*, Dec. 2010, pp. 5420–5425.

[7] S. Kuindersma, R. Deits, M. Fallon, A. Valenzuela, H. Dai, F. Permenter, T. Koolen, P. Marion, and R. Tedrake, "Optimization-based locomotion planning, estimation, and control design for the atlas humanoid robot," *Auton. Robots*, vol. 40, no. 3, pp. 429–455, Mar. 2016.

[8] H. Dai, A. Valenzuela, and R. Tedrake, "Whole-body motion planning with centroidal dynamics and full kinematics," in *Proc. IEEE-RAS Int. Conf. Humanoid Robots*, Nov. 2014, pp. 295–302.

[9] S. Boyd, S. P. Boyd, and L. Vandenberghe, *Convex Optimization*. Cambridge, U.K.: Cambridge Univ. Press, 2004.

[10] M. S. Bazaraa, H. D. Sherali, and C. M. Shetty, *Nonlinear Programming: Theory and Algorithms*. Hoboken, NJ, USA: Wiley, 2013.

[11] E. G. Hemingway and O. M. O'Reilly, "Perspectives on Euler angle singularities, gimbal lock, and the orthogonality of applied forces and applied moments," *Multibody Syst. Dyn.*, vol. 44, no. 1, pp. 31–56, Sep. 2018.

[12] F. Bullo and R. M. Murray, "Tracking for fully actuated mechanical systems: A geometric framework," *Automatica*, vol. 35, no. 1, pp. 17–34, Jan. 1999.

[13] F. Bullo and A. D. Lewis, *Geometric Control of Mechanical Systems: Modeling, Analysis, and Design for Simple Mechanical Control Systems*, vol. 49. Cham, Switzerland: Springer, 2019.

[14] J. W. Simpson-Porco and F. Bullo, "Contraction theory on Riemannian manifolds," *Syst. Control Lett.*, vol. 65, pp. 74–80, Mar. 2014.

[15] W. Lohmiller and J.-J. E. Slotine, "On contraction analysis for non-linear systems," *Automatica*, vol. 34, no. 6, pp. 683–696, 1998.

[16] G. Wu and K. Sreenath, ''Variation-based linearization of nonlinear systems evolving on *SO*(3) and $\mathbb{S}^2$,'' *IEEE Access*, vol. 3, pp. 1592–1604, 2015.

[17] H. Wang, Z. Li, H. Xiong, and X. Nian, ''Robust $H_\infty$ attitude tracking control of a quadrotor UAV on SO(3) via variation-based linearization and interval matrix approach,'' *ISA Trans.*, vol. 87, pp. 10–16, Apr. 2019.

[18] M. Chignoli and P. M. Wensing, ''Variational-based optimal control of underactuated balancing for dynamic quadrupeds,'' *IEEE Access*, vol. 8, pp. 49785–49797, 2020.

[19] J. Nubert, J. Kohler, V. Berenz, F. Allgöwer, and S. Trimpe, ''Safe and fast tracking on a robot manipulator: Robust MPC and neural network control,'' *IEEE Robot. Autom. Lett.*, vol. 5, no. 2, pp. 3050–3057, Apr. 2020.

[20] Z. Wang, X. Zhou, C. Xu, and F. Gao, ''Geometrically constrained trajectory optimization for multicopters,'' *IEEE Trans. Robot.*, vol. 38, no. 5, pp. 3259–3278, Oct. 2022.

[21] F. Gao, Y. Lin, and S. Shen, ''Gradient-based online safe trajectory generation for quadrotor flight in complex environments,'' in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Sep. 2017, pp. 3681–3688.

[22] X. Zhang, A. Liniger, and F. Borrelli, ''Optimization-based collision avoidance,'' *IEEE Trans. Control Syst. Technol.*, vol. 29, no. 3, pp. 972–983, May 2021.

[23] X. Zhang, A. Liniger, A. Sakai, and F. Borrelli, ''Autonomous parking using optimization-based collision avoidance,'' in *Proc. IEEE Conf. Decis. Control (CDC)*, Dec. 2018, pp. 4327–4332.

[24] I. Griva, S. G. Nash, and A. Sofer, *Linear and Nonlinear Optimization*, vol. 108. Philadelphia, PA, USA: SIAM, 2009.

[25] B. Zhou, F. Gao, L. Wang, C. Liu, and S. Shen, ''Robust and efficient quadrotor trajectory generation for fast autonomous flight,'' *IEEE Robot. Autom. Lett.*, vol. 4, no. 4, pp. 3529–3536, Oct. 2019.

[26] J. Park, J. Kim, I. Jang, and H. J. Kim, ''Efficient multi-agent trajectory planning with feasibility guarantee using relative Bernstein polynomial,'' in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2020, pp. 434–440.

[27] W. Ding, L. Zhang, J. Chen, and S. Shen, ''Safe trajectory generation for complex urban environments using spatio-temporal semantic corridor,'' *IEEE Robot. Autom. Lett.*, vol. 4, no. 3, pp. 2997–3004, Jul. 2019.

[28] D. Mellinger, N. Michael, and V. Kumar, ''Trajectory generation and control for precise aggressive maneuvers with quadrotors,'' *Int. J. Robot. Res.*, vol. 31, no. 5, pp. 664–674, 2012.

[29] D. H. Jacobson and D. Q. Mayne, *Differential Dynamic Programming*, no. 24. Amsterdam, The Netherlands: Elsevier, 1970.

[30] Y. Tassa, N. Mansard, and E. Todorov, ''Control-limited differential dynamic programming,'' in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2014, pp. 1168–1175.

[31] H. Almubarak, K. Stachowicz, N. Sadegh, and E. A. Theodorou, ''Safety embedded differential dynamic programming using discrete barrier states,'' *IEEE Robot. Autom. Lett.*, vol. 7, no. 2, pp. 2755–2762, Apr. 2022.

[32] E. Theodorou, Y. Tassa, and E. Todorov, ''Stochastic differential dynamic programming,'' in *Proc. Amer. Control Conf.*, Jun. 2010, pp. 1125–1132.

[33] 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. 2016.

[34] W. W. Hager and H. Zhang, ''A new active set algorithm for box constrained optimization,'' *SIAM J. Optim.*, vol. 17, no. 2, pp. 526–557, 2006.

[35] H. J. Ferreau, C. Kirches, A. Potschka, H. G. Bock, and M. Diehl, ''qpOASES: A parametric active-set algorithm for quadratic programming,'' *Math. Program. Comput.*, vol. 6, no. 4, pp. 327–363, 2014.

[36] M. Hintermüller, K. Ito, and K. Kunisch, ''The primal-dual active set strategy as a semismooth Newton method,'' *SIAM J. Optim.*, vol. 13, no. 3, pp. 865–888, 2002.

[37] H. J. Ferreau, H. G. Bock, and M. Diehl, ''An online active set strategy to overcome the limitations of explicit MPC,'' *Int. J. Robust Nonlinear Control*, vol. 18, no. 8, pp. 816–830, Jul. 2007.

[38] R. Milman and E. J. Davison, ''A fast MPC algorithm using nonfeasible active set methods,'' *J. Optim. Theory Appl.*, vol. 139, no. 3, pp. 591–616, Dec. 2008.

[39] H. D. Quang, T. L. Tran, T. N. Manh, C. N. Manh, T. N. Nhu, and N. B. Duy, ''Design a nonlinear MPC controller for autonomous mobile robot navigation system based on ROS,'' *Int. J. Mech. Eng. Robot. Res.*, vol. 11, no. 6, pp. 379–388, 2022.

[40] D. Arnström, A. Bemporad, and D. Axehill, ''A dual active-set solver for embedded quadratic programming using recursive LDL$^T$ updates,'' *IEEE Trans. Autom. Control*, vol. 67, no. 8, pp. 4362–4369, Aug. 2022.

[41] G. I. Boutselis and E. Theodorou, ''Discrete-time differential dynamic programming on lie groups: Derivation, convergence analysis, and numerical results,'' *IEEE Trans. Autom. Control*, vol. 66, no. 10, pp. 4636–4651, Oct. 2021.

[42] Z. Xie, C. K. Liu, and K. Hauser, ''Differential dynamic programming with nonlinear constraints,'' in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2017, pp. 695–702.

**SHU LIU** received the B.S. degree from Beijing Jiaotong University, Beijing, China, in 2005.

He is currently a Senior Engineer with the China Academy of Information and Communications Technology. He has 16 years' working experience on the internet, the Internet of Things, and control theory.

**DONGPO LIU** received the B.S. degree in information and computer science and the M.S. degree in control theory and engineering from Shandong University, Shandong, China, in 2007 and 2010, respectively.

He was a Research Engineer at the Institute of Automation of Chinese Academy of Science, from 2010 to 2018. Since 2018, he has been the Director of the Center of the Business Management Center, Institute of Industrial Internet and Internet of Things, China Academy of Information and Communications Technology.