On-Manifold Model Predictive Control for Trajectory Tracking on Robotic Systems

Guozheng Lu, Wei Xu, and Fu Zhang, *Member, IEEE*

***Abstract*—Robotic systems usually evolve on manifolds, which are often overparameterized or minimally parameterized (but with singularities) in model predictive control (MPC). How to naturally incorporate the system manifold constraints without overparameterization or singularity is a fundamental problem when deploying MPC on robotic systems. In this article, we propose a unified on-manifold MPC framework to address this issue. This framework first formulates the MPC based on a canonical representation of on-manifold systems. Then, the on-manifold MPC is solved by linearizing the system at each point along the trajectory under tracking. There are two main advantages of the proposed scheme. First, the linearized system leads to an equivalent error system represented by local coordinates without singularities. Second, the process of system modeling, error-system derivation, linearization, and control has the manifold constraints completely decoupled from the system descriptions, enabling us to develop a symbolic MPC framework naturally encapsulating the manifold constraints. In this framework, one needs only to supply system-specific descriptions without dealing with the manifold constraints. To validate the generality of the proposed framework, we implement it on two different robotic platforms, a quadrotor unmanned aerial vehicle (UAV) evolving on a Lie group, and an unmanned ground vehicle moving on curved surface with a non-Lie group structure. To validate the nonsingularity and tracking performance, we test it in tracking aggressive UAV trajectories. Experimental results show that with a single, global on-manifold MPC, the quadrotor tracks highly aggressive trajectories with large actuator efforts and attitude variation (360°) in both roll and pitch directions.**

***Index Terms*—Manifolds, mobile robots, predictive control.**

I. Introduction

MODEL predictive control (MPC) has brought significant benefits to robotic research and applications in recent years. The main advantage of MPC is the fact that it computes the current optimal input to produce the best possible behavior in the future based on the system dynamic model. The predictive nature makes MPC a powerful tool for high-performance trajectory tracking on robotic systems, such as autonomous driving [1], unmanned aerial vehicles (UAVs) [2], mobile manipulators [3], and legged robots [4]. In particular, MPC plays an essential role in challenging dynamic control problems, like the locomotion control of the Atlas humanoid robot by Boston Dynamics [5]. Moreover, augmented objective functions and constraints also make MPC easily extendable to multiple-input–multiple-output systems and able to cater for constraints on state and input, which are necessary for high-level robot autonomy, such as obstacle avoiding [6], perception awareness [7], and swarm collaboration [8].

Although MPC has already yielded various exciting results in robotics, a fundamental problem is how to incorporate the manifold constraints of the robot state into an MPC. MPC is commonly designed for a system described by ordinary differential equation where the state is a flat vector in the Euclidean space $\mathbb{R}^n$, while the configuration space of robotic systems usually involves curved, non-vector-space manifolds. For instance, the following.

1) *Lie group:* The rotation workspace of satellites [9] is lying on the special orthogonal group $SO(3)$; the rigid rotation and translation workspace of manipulators [10] and UAVs [11] is the special Euclidean group $\mathrm{SE}(3)$; the movement of cars on a flat ground is considered as $\mathrm{SE}(2)$.
2) *2-sphere* $\mathbb{S}^2$*:* 3-D pendulum movement of pointing devices [9].
3) 2-D surface: Cars are always restricted to move on ground surfaces with only two degree of freedom [12].

In these cases, theories and experiences developed for MPC formed in $\mathbb{R}^n$ cannot be immediately extended to robotic systems evolving on manifolds.

To incorporate the manifold constraints, existing MPC methods applied on robotic systems often use minimal- or overparameterization as detailed in the following.

An intuitive way to eliminate manifold constraints is to parameterize a manifold directly by minimal parameters. The resultant system model based on the minimal parameterization is generally nonlinear and can be controlled using any nonlinear control approaches (e.g., [13] and [14]). While being straightforward, this method completely gives up the intrinsic geometry structure of manifolds. Due to the topological obstruction, a minimal parameterization always suffers from certain singularities that significantly restrict the system operation range. For example, $\mathrm{SO}(3)$ can be minimally represented by three Euler angles with

Manuscript received 23 April 2022; revised 10 August 2022 and 2 September 2022; accepted 25 September 2022. Date of publication 12 October 2022; date of current version 3 April 2023. This project was supported in part by Hong Kong RGC ECS under Grant 27202219 and in part by DJI donation. *(Corresponding author: Guozheng Lu.)*

The authors are with the Department of Mechanical Engineering, The University of Hong Kong, Hong Kong (e-mail: gzlu@hku.hk; xuweii@hku.hk; fuzhang@hku.hk).

This article has supplementary material provided by the authors and color versions of one or more figures available at https://doi.org/10.1109/TIE.2022.3212397.

Digital Object Identifier 10.1109/TIE.2022.3212397

respect to (w.r.t.) the identity. Some existing studies [2], [4], [15], [16], [17], [18] adopted this method to represent $\mathrm{SO}(3)$ in MPC formulation, but would encounter singularities when the second angle reaches $\pm 90^\circ$.

Another possible method for handling manifold constraints in MPC is overparameterization. Although singularities can be avoided, the overparameterization method usually introduces extra constraints or special treatments. For instance, a unit quaternion is often chosen to parameterize $SO(3)$ in MPC [7], [19], but brings an extra constraint of normalization to reinforce the unit length explicitly. Similarly, the variation-based linearization method proposed in [20] presented the manifold dynamics of 2-D 2-sphere $\mathbb{S}^2$ by 3-D vectors, resulting in a constrained linear time-varying system. Kalabić [21] gave up the standard form of MPC in the Euclidean space but presented the attitude state on $SO(3)$ by a rotation matrix and designed a very special cost function by calculating the matrix trace, making the cost weight tuning less intuitive due to the overparameterization in the rotation matrix. The overparameterization also makes it difficult to reweight different components in the state (e.g., roll, pitch, and yaw in a rotation).

In this article, we formally address the design of MPC with explicit consideration of manifold constraints. To the best of our knowledge, there are few studies formally consider the design of MPC on general manifolds.

Our first contribution is a singularity free, on-manifold MPC framework with minimal parameterization for robots trajectory tracking. In our framework, the system state is mapped to its local coordinates around each point of the reference trajectory on the state manifold. Unlike the overparameterization methods [7], [19], [21], our method uses local coordinates to represent the system state, and hence, is minimally parameterized. The resultant system is an ordinary nonlinear system in the Euclidean space, where existing MPC techniques, theories, and tuning experiences can be applied directly without involving any extra constraint. Moreover, the point-wise local parameterization in our method effectively avoids singularities suffered by other minimal parameterization methods (e.g., Euler angles) [2], [4], [13], [14], [15], [16], [17], [18]. This geometric approach is inspired by the error-state extended Kalman filter (ESEKF [22], or named multiplicative extended Kalman filter (EKF) [23]), and is actually not the first time applied to control systems. For instance, Hong et al. [24] verified the nonlinear MPC (NMPC) on $\mathrm{SO}(3)$ for legged locomotion in a simulation, but did not generalize this method to general manifolds. Lewis et al. [25] used a similar linearization technique in classic LQR control. Compared to these works, we specifically focus on the MPC controller and its generalization to general manifolds.

We generalize the proposed MPC framework to an unified framework for general on-manifold robotic systems by formalizing two symbolic operations, referring to our previous work [26]. One is the operator $\boxplus$ (and its inverse $\boxminus$), which describes the map from the system state to its local coordinates, and the other is $\oplus$, which denotes the evolvement of the system state over time while naturally ensuring the manifold constraints. This formalization leads to a unified symbolic framework for treating manifold constraints in an MPC. Following this symbolic framework, one can easily design an on-manifold MPC controller by only describing the system using operator $\oplus$ and its state manifold using $\boxplus$ without dealing with the manifold constraints in each individual system.

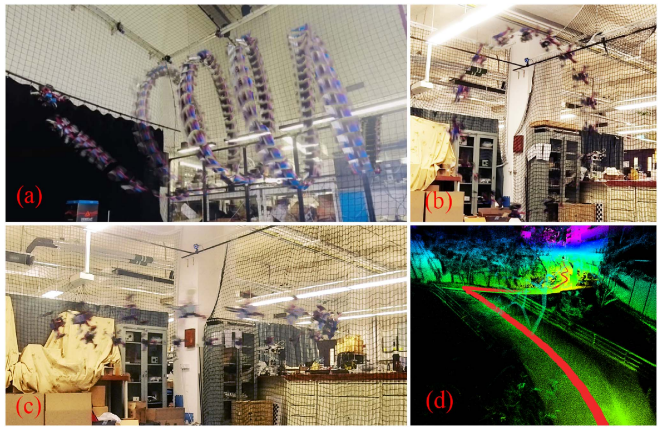


Fig. 1. Example applications of our MPC to robotic systems evolving on different manifolds, including a quadrotor UAV performing aerobatics on $\mathrm{SE}(3)$ (e.g., (a) helix with flips, (b) flipping on vertical circle, (c) loop accelerating), and (d) UGV tracking a trajectory on a curved surface $\mathcal{S}$. A video demonstration is available at: https://youtu.be/3Iacds2trsc.

The proposed framework including the canonical symbolic representation and MPC controller, has been implemented on two robotic platforms evolving on different manifolds, a quadrotor UAV evolving on the Lie group $\mathrm{SE}(3)$ and an unmanned ground vehicle (UGV) following a constant-speed path on a curved surface, which is a non-Lie group manifold (see Fig. 1). Using the same controller without switching, the quadrotor tracks three aggressive trajectories with large actuator efforts (i.e., collective thrust nearly 4 g and angular rate up to 800°/s) and large rotation variation (i.e., rolling and pitching over 360°), demonstrating high tracking performance and singularity-free nature of the proposed framework.

The rest of this article is organized as follows. Section II introduces the definition of operations of $\boxplus/\boxminus$. A canonical, unified and on-manifold MPC formulation for trajectory tracking is proposed in Section III and solved in Section IV. Section V explains the isolation of on-manifold constraints. In Section VI, experimental results on two robotic systems are demonstrated. Finally, Section VII concludes this article.

II. Operations on Differentiable Manifolds

A. Manifolds

A n-manifold is a topological space [27] where the neighborhood at each point is homeomorphic to an open subset of Euclidean space $\mathbb{R}^n$ (the *homeomorphic space*). More specifically, a manifold can be parameterized by a set of local coordinate charts $(U_{\mathbf{x}}, \varphi_{\mathbf{x}})$, where $\varphi_{\mathbf{x}} : \mathcal{M} \mapsto \mathbb{R}^n$ is a map projecting the neighborhood $U_{\mathbf{x}} \subset \mathcal{M}$ around a point $\mathbf{x}$ to the linear homeomorphic space $\mathbb{R}^n$. $\varphi_{\mathbf{x}}$ is called the local coordinate at point $\mathbf{x}$ and inversely, $\varphi_{\mathbf{x}}^{-1}$: $\mathbb{R}^n \mapsto \mathcal{M}$. The region of the neighborhood $U_{\mathbf{x}}$ is related to the choice of $\varphi_{\mathbf{x}}$ and homeomorphic to an n-ball

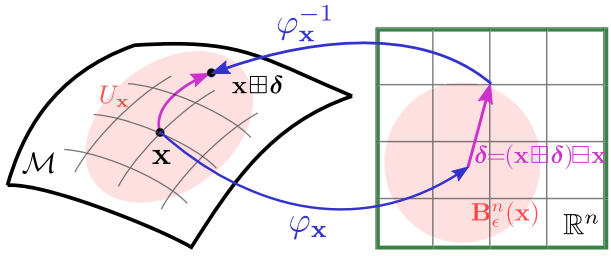


Fig. 2. Illustration of the $\boxplus/\boxminus$ operations on manifold $\mathcal{M}$ with a perturbation in its homeomorphic space $\mathbf{B}_\epsilon^n(\mathbf{x}) \in \mathbb{R}^n$.

TABLE I
OPERATIONS OF DIFFERENT PRIMITIVE MANIFOLDS.
$\mathbf{E}_{12} = [\mathbf{e}_1\ \mathbf{e}_2], \mathbf{e}_1 = [1\ 0\ 0]^T, \mathbf{e}_2 = [0\ 1\ 0]^T$

$\mathcal{M}$	$\mathbf{x}\boxplus\boldsymbol{\delta}$	$\mathbf{y}\boxminus\mathbf{x}$	$\mathbf{x}\oplus\boldsymbol{\delta}^e$
$\mathbb{R}^n$	$\mathbf{x}+\boldsymbol{\delta}$	$\mathbf{y}-\mathbf{x}$	$\mathbf{x}+\boldsymbol{\delta}^e$
$SO(n)$	$\mathbf{x}\cdot\mathrm{Exp}(\boldsymbol{\delta})$	$\mathrm{Log}\left(\mathbf{x}^{-1}\mathbf{y}\right)$	$\mathbf{x}\cdot\mathrm{Exp}(\boldsymbol{\delta}^e)$
$\mathcal{S}$	$\begin{bmatrix}\mathbf{E}_{12}^T\mathbf{x}+\boldsymbol{\delta}\\ F\left(\mathbf{E}_{12}^T\mathbf{x}+\boldsymbol{\delta}\right)\end{bmatrix}$	$\mathbf{E}_{12}^T(\mathbf{y}-\mathbf{x})$	$\begin{bmatrix}\mathbf{E}_{12}^T\mathbf{x}+\boldsymbol{\delta}^e\\ F\left(\mathbf{E}_{12}^T\mathbf{x}+\boldsymbol{\delta}^e\right)\end{bmatrix}$

$\mathbf{B}_\epsilon^n(\mathbf{x}) \in \mathbb{R}^n$ where $\|\mathbf{B}_\epsilon^n(\mathbf{x})\| < \epsilon$ and $\epsilon > 0$. Note that $\varphi_{\mathbf{x}}$, the local coordinate, has the same dimension as the manifold $\mathcal{M}$, and hence, is a minimal parameterization of the neighborhood $U_{\mathbf{x}}$.

B. Operations of $\boxplus/\boxminus$

Referring to [26], [28], and [29], we utilize a pair of symbolic operations $\boxplus/\boxminus$ as follows to describe the local coordinate $\varphi_{\mathbf{x}}$ around a point $\mathbf{x} \in \mathcal{M}$.

$$\boxplus : \mathcal{M} \times \mathbb{R}^n \mapsto \mathcal{M}, \quad \mathbf{x}\boxplus\boldsymbol{\delta} = \varphi_{\mathbf{x}}^{-1}(\boldsymbol{\delta})\ \forall\boldsymbol{\delta} \in \mathbf{B}_\epsilon^n(\mathbf{x}) \tag{1a}$$

$$\boxminus : \mathcal{M} \times \mathcal{M} \mapsto \mathbb{R}^n, \quad \mathbf{y}\boxminus\mathbf{x} = \varphi_{\mathbf{x}}(\mathbf{y})\ \forall\mathbf{y} \in U_{\mathbf{x}}. \tag{1b}$$

Intuitively, $\boxplus$ applies a small change or perturbation $\boldsymbol{\delta}$ in the homoeomorphic space at a point to yield a new point on the manifold $\mathcal{M}$. Conversely, $\boxminus$ represents the difference between two points on the manifold in the homoeomorphic space. The on-manifold operations are visualized in Fig. 2. For a chosen $\boxplus/\boxminus$, the following axioms are held for every $\mathbf{x} \in \mathcal{M}$ from the definition:

$$\mathbf{x}\boxplus\mathbf{0} = \mathbf{x}, \quad \mathbf{x}\boxplus(\mathbf{y}\boxminus\mathbf{x}) = \mathbf{y}, \quad (\mathbf{x}\boxplus\boldsymbol{\delta})\boxminus\mathbf{x} = \boldsymbol{\delta}. \tag{2}$$

C. Examples of Practically Important Manifolds

In this section, we give examples of $\boxplus/\boxminus$ for some commonly used manifolds. A summary of these examples is shown in Table I.

1) Example 1: Euclidean Space $\mathbb{R}^n$: For $\mathcal{M} = \mathbb{R}^n$, the homoeomorphic space is itself at any point and based on which, the operations $\boxplus/\boxminus$ are standard vector addition and subtraction, and thus, $U_{\mathbf{x}} = \mathbf{B}_\epsilon^n(\mathbf{x}) = \mathbb{R}^n$.

2) Example 2: Special Orthogonal Group $SO(n)$: Special orthogonal group $\mathrm{SO}(n) \triangleq \{\mathbf{R} \in \mathbb{R}^{n\times n} | \mathbf{R}^T\mathbf{R} = \mathbf{I}, \det\mathbf{R} = 1\}$ for $n = 2, 3$ are the most common workspaces for robotic systems involving rotations. Due to its Lie group structure, there exists an exponential map $\mathrm{Exp}(\boldsymbol{\delta}) = \exp_{\mathrm{SO}(n)}(\lfloor\boldsymbol{\delta}\rfloor)$ and its inverse map $\mathrm{Log}(\mathbf{x}) = \log_{\mathrm{SO}(n)}(\mathbf{x})^\vee$ at each point [10], which naturally serves the $\boxplus/\boxminus$ operations. Because the exponential map and its inverse represent the geodesics (i.e., the shortest path) of the rotation, the homeomorphic neighborhood should be confined $\|\mathbf{B}_\epsilon^n(\mathbf{x})\| < \pi$.

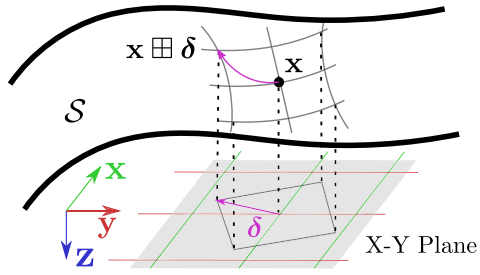


Fig. 3. Illustration of the $\boxplus$ operations on 2-D surface $\mathcal{S}$ with a perturbation in the homoeomorphic space, i.e., the X–Y plane.

3) Example 3: 2-D Surface $\mathcal{S}$: A ground surface can be generally modeled as a smooth surface manifold embedded in $\mathbb{R}^3$: $\mathcal{S} \triangleq \{\mathbf{x} \in \mathbb{R}^3 | z = F(x, y)\}$, where $z = F(x, y)$ is the height parameterized by horizontal coordinates (x, y) [12]. The perturbation $\boldsymbol{\delta} \in \mathbb{R}^2$ is directly represented in the X–Y plane of the space that embeds $\mathcal{S}$ (see Fig. 3). The homeomorphic neighborhood $\mathbf{B}_\epsilon^2(\mathbf{x})$ is the domain of function $F(x, y)$.

4) Example 4: Compound Manifolds: The cartesian product [28] of two (and by induction arbitrary number of) primitive manifolds $\mathcal{M}_1, \mathcal{M}_2$ is considered as a compound manifold $\mathcal{M} = \mathcal{M}_1 \times \mathcal{M}_2$, and satisfies

$$\begin{pmatrix}\mathbf{x}_1\\ \mathbf{x}_2\end{pmatrix} \boxplus \begin{bmatrix}\boldsymbol{\delta}_1\\ \boldsymbol{\delta}_2\end{bmatrix} = \begin{pmatrix}\mathbf{x}_1\boxplus\boldsymbol{\delta}_1\\ \mathbf{x}_2\boxplus\boldsymbol{\delta}_2\end{pmatrix}, \begin{pmatrix}\mathbf{y}_1\\ \mathbf{y}_2\end{pmatrix} \boxminus \begin{pmatrix}\mathbf{x}_1\\ \mathbf{x}_2\end{pmatrix} = \begin{bmatrix}\mathbf{y}_1\boxminus\mathbf{x}_1\\ \mathbf{y}_2\boxminus\mathbf{x}_2\end{bmatrix} \tag{3}$$

for $\mathbf{x}_1 \in \mathcal{M}_1, \mathbf{x}_2 \in \mathcal{M}_2, , \mathbf{y}_1 \in U_{\mathbf{x}_1} \subset \mathcal{M}_1, \mathbf{y}_2 \in U_{\mathbf{x}_2} \subset \mathcal{M}_2$, $\boldsymbol{\delta}_1 \in \mathbf{B}_{\epsilon_1}^{n_1} \subset \mathbb{R}^{n_1}$, and $\boldsymbol{\delta}_2 \in \mathbf{B}_{\epsilon_2}^{n_2} \subset \mathbb{R}^{n_2}$.

III. Symbolic MPC Formulation on Manifolds

In this section, we propose a unified MPC formulation with explicit consideration of manifold constraints. The MPC formulation utilizes the operation $\boxplus/\boxminus$ to represent the state deviation from a given trajectory. Moreover, we introduce a notation $\oplus$ to describe the system state equation in a unified way we term as the canonical representation.

A. Canonical Representation of On-Manifold Systems

Considering a robotic system lying on a n-manifold $\mathcal{M}$, its discretization with sampling period Δt can be written into a compact form, termed as the canonical form, as follows:

$$\mathbf{x}_{k+1} = \mathbf{x}_k \oplus (\Delta t\mathbf{f}(\mathbf{x}_k, \mathbf{u}_k)) \tag{4}$$

where we used the Euler method in the discretization assuming that the state flows along a constant direction during one sampling period. $\mathbf{x} \in \mathcal{M}$ is the system state, $\mathbf{u} \in \mathbb{R}^m$ is the control input in the actuator space, vector $\mathbf{f}(\mathbf{x}_k, \mathbf{u}_k) \in \mathbb{R}^l$ represents the state perturbation caused by the control input, and the

operation $\oplus$ in (4) denotes compactly the "addition" of a state at its present time and the input-caused perturbation vector, to yield the next state that naturally remains on the manifold $\mathcal{M}$. Mathematically, $\oplus$ can be thought as a map of the following form:

$$\oplus : \mathcal{M} \times \mathbb{R}^l \mapsto \mathcal{M}. \tag{5}$$

In particular, if the system operates in $\mathbb{R}^n$, $\oplus$ reduces to the usual vector addition, leading to the usual discrete model $\mathbf{x}_{k+1} = \mathbf{x}_k + \Delta t \mathbf{f}(\mathbf{x}_k, \mathbf{u}_k)$. More generally, $\oplus$ depends on the specific system and its state manifold in consideration. For example, a system of 3-D pendulum with position $\mathbf{q} \in \mathbb{S}^2$ and angular velocity $\boldsymbol{\omega} \in \mathbb{R}^3$ satisfies $\dot{\mathbf{q}} = \boldsymbol{\omega} \times \mathbf{q}$ [20], so one discretization method that naturally encodes the manifold constraints on $\mathbb{S}^2$ is $\mathbf{q}_{k+1} = \mathrm{Exp}(\boldsymbol{\omega}_k \Delta t)\mathbf{q}_k$. Casting it into the canonical form (4) leads to $\mathbf{f}(\mathbf{x}_k, \mathbf{u}_k) = \boldsymbol{\omega}_k \in \mathbb{R}^3$ and $\mathbf{x}_k \oplus (\Delta t \mathbf{f}(\mathbf{x}_k, \mathbf{u}_k)) = \mathrm{Exp}(\Delta t \mathbf{f}(\mathbf{x}_k, \mathbf{u}_k))\mathbf{x}_k$. For the attitude kinematics $\dot{\mathbf{R}} = \mathbf{R}\lfloor\boldsymbol{\omega}\rfloor$, one discretization method that naturally encodes the manifold constraints is $\mathbf{R}_{k+1} = \mathbf{R}_k \mathrm{Exp}(\boldsymbol{\omega}_k \Delta t)$. Casting it into the canonical form (4) leads to $\mathbf{f}(\mathbf{x}_k, \mathbf{u}_k) = \boldsymbol{\omega}_k \in \mathbb{R}^3$ and $\mathbf{x}_k \oplus (\Delta t \mathbf{f}(\mathbf{x}_k, \mathbf{u}_k)) = \mathbf{x}_k \mathrm{Exp}(\Delta t \mathbf{f}(\mathbf{x}_k, \mathbf{u}_k))$.

In principle, the operation $\oplus$ depends on both the system dynamics and its state manifold. In practice, systems operating on the same manifold usually can share the same definition of $\oplus$. As a consequence, we could assign the operation $\oplus$ to each manifold and tweak the state equation of each individual system based on the already defined $\oplus$. Table I summarizes the $\oplus$ operations for some commonly used manifolds.

B. On-Manifold MPC Formulation for Trajectory Tracking

Based on the canonical system representation in (4) and further making use of $\boxminus$ to minimally represent the deviation between the actual state and trajectory, we can formally formulate the MPC problem for systems evolving on manifold

$$\begin{aligned} \min_{\mathbf{u}_k} \ & \sum_{k=0}^{N-1} \left(\|\mathbf{x}_k \boxminus \mathbf{x}_k^d\|_{\mathbf{Q}_k}^2 + \|\mathbf{u}_k - \mathbf{u}_k^d\|_{\mathbf{R}_k}^2 \right) + \|\mathbf{x}_N \boxminus \mathbf{x}_N^d\|_{\mathbf{P}_N}^2 \\ \text{s.t.} \ & \mathbf{x}_{k+1} = \mathbf{x}_k \oplus \Delta t \mathbf{f}(\mathbf{x}_k, \mathbf{u}_k), \quad \mathbf{x}_0 = \mathbf{x}_{\text{init}} \\ & \mathbf{u}_k \in \mathbb{U}, \quad k = 0, \ldots, N-1 \end{aligned} \tag{6}$$

where the superscript $(\cdot)^d$ denotes the reference trajectory satisfying $\mathbf{x}_{k+1}^d = \mathbf{x}_k^d \oplus \Delta t \mathbf{f}(\mathbf{x}_k^d, \mathbf{u}_k^d)$ and $\mathbf{u}_k^d \in \mathbb{U}$. $\mathbf{Q}_k \succ 0$, $\mathbf{P}_N \succ 0$, and $\mathbf{R}_k \succ 0$ are penalty matrices of the stage state, terminal state and input, respectively, and $\oplus$ are defined on the manifold $\mathcal{M}$ where the system state lies on. $\mathbb{U} = \{\mathbf{u} \in \mathbb{R}^m | \mathbf{u}_{\min} \leq \mathbf{u} \leq \mathbf{u}_{\max}\}$ is the boundary of inputs.

IV. On-Manifold MPC Solution Based on the Error System

When solving the on-manifold MPC problem in (6), the manifold constraints that $\mathbf{x}_k \in \mathcal{M}$ must be always satisfied during the optimization. To address this issue, in this section, we transform the original system (4) with the manifold constraints into a minimally parameterized error system without such constraints.

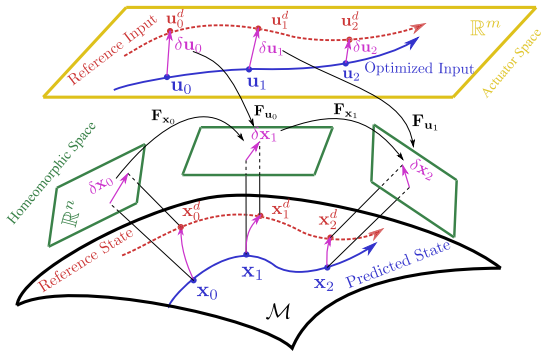


Fig. 4. Interpretation of the error system and its linearization.

The resultant error system is an ordinary nonlinear system in the normal Euclidean space, transforming the on-manifold MPC formulation (6) into a standard MPC that can be solved efficiently. An illustration of the transformation between the original system state and the error system is shown in Fig. 4.

A. Linearized Error System

To avoid the manifold constraints encoded in the $\oplus$ operation in (4), we parameterize the system state $\mathbf{x}_k \in \mathcal{M}$ by its error-state $\delta\mathbf{x}_k \in \mathbb{R}^n$ defined by the $\boxplus/\boxminus$ in Section II as follows:

$$\delta\mathbf{x}_k = \mathbf{x}_k \boxminus \mathbf{x}_k^d \in \mathbb{R}^n, \quad \delta\mathbf{u}_k = \mathbf{u}_k \boxminus \mathbf{u}_k^d \in \mathbb{R}^m. \tag{7}$$

Substituting (7) into (4) leads to

$$\begin{aligned} \delta\mathbf{x}_{k+1} &= \left(\mathbf{x}_k \oplus \Delta t \mathbf{f}(\mathbf{x}_k, \mathbf{u}_k)\right) \boxminus \left(\mathbf{x}_k^d \oplus \Delta t \mathbf{f}\left(\mathbf{x}_k^d, \mathbf{u}_k^d\right)\right) \\ &= \left(\left(\mathbf{x}_k^d \boxplus \delta\mathbf{x}_k\right) \oplus \Delta t \mathbf{f}\left(\mathbf{x}_k^d \boxplus \delta\mathbf{x}_k, \mathbf{u}_k^d \boxplus \delta\mathbf{u}_k\right)\right) \\ &\quad \boxminus \left(\mathbf{x}_k^d \oplus \Delta t \mathbf{f}\left(\mathbf{x}_k^d, \mathbf{u}_k^d\right)\right) \end{aligned} \tag{8}$$

which is an equivalent representation of the original system (4). To linearize the error system (8), performing the Taylor series expansion and keeping terms up to the first order leads to the linearization as follows:

$$\delta\mathbf{x}_{k+1} \approx \mathbf{F}_{\mathbf{x}_k} \delta\mathbf{x}_k + \mathbf{F}_{\mathbf{u}_k} \delta\mathbf{u}_k \tag{9}$$

where $\mathbf{F}_{\mathbf{x}_k}$ is the Jacobian w.r.t. $\delta\mathbf{x}_k$ (evaluated at zero) and can be computed by the chain rule as follows:

$$\begin{aligned} \mathbf{F}_{\mathbf{x}_k} = & \frac{\partial \left(\left((\mathbf{x}_k^d \boxplus \delta\mathbf{x}_k) \oplus \Delta t \mathbf{f}(\mathbf{x}_k^d, \mathbf{u}_k^d) \right) \boxminus \left(\mathbf{x}_k^d \oplus \Delta t \mathbf{f}(\mathbf{x}_k^d, \mathbf{u}_k^d) \right) \right)}{\partial \delta\mathbf{x}_k} \\ & + \frac{\partial \left(\left(\mathbf{x}_k^d \oplus \Delta t \mathbf{f}(\mathbf{x}_k^d \boxplus \delta\mathbf{x}_k, \mathbf{u}_k^d) \right) \boxminus \left(\mathbf{x}_k^d \oplus \Delta t \mathbf{f}(\mathbf{x}_k^d, \mathbf{u}_k^d) \right) \right)}{\partial \Delta t \mathbf{f}(\mathbf{x}_k^d \boxplus \delta\mathbf{x}_k, \mathbf{u}_k^d)} \\ & \cdot \Delta t \frac{\partial \mathbf{f}(\mathbf{x}_k^d \boxplus \delta\mathbf{x}_k, \mathbf{u}_k^d)}{\partial \delta\mathbf{x}_k} \bigg|_{\delta\mathbf{x}_k = 0} \\ = & \mathbf{G}_{\mathbf{x}_k} + \Delta t \mathbf{G}_{\mathbf{f}_k} \cdot \frac{\partial \mathbf{f}(\mathbf{x}_k^d \boxplus \delta\mathbf{x}_k, \mathbf{u}_k^d)}{\partial \delta\mathbf{x}_k} \bigg|_{\delta\mathbf{x}_k = 0} \end{aligned} \tag{10}$$

and $\mathbf{F}_{\mathbf{u}_k}$ is the Jacobian w.r.t. $\delta\mathbf{u}_k$ (evaluated at zero)

$$\begin{aligned}\mathbf{F}_{\mathbf{u}_k} &= \frac{\partial\left(\left(\mathbf{x}_k^d \oplus \Delta t\mathbf{f}(\mathbf{x}_k^d, \mathbf{u}_k^d \boxplus \delta\mathbf{u}_k)\right) \boxminus \left(\mathbf{x}_k^d \oplus \Delta t\mathbf{f}(\mathbf{x}_k^d, \mathbf{u}_k^d)\right)\right)}{\partial \Delta t\mathbf{f}(\mathbf{x}_k^d, \mathbf{u}_k^d \boxplus \delta\mathbf{u}_k)} \\ &\quad \cdot \Delta t \frac{\partial \mathbf{f}(\mathbf{x}_k^d, \mathbf{u}_k^d \boxplus \delta\mathbf{u}_k)}{\partial \delta\mathbf{u}_k}\bigg|_{\delta\mathbf{u}_k=0} \\ &= \Delta t \mathbf{G}_{\mathbf{f}_k} \frac{\partial \mathbf{f}(\mathbf{x}_k^d, \mathbf{u}_k^d \boxplus \delta\mathbf{u}_k)}{\partial \delta\mathbf{u}_k}\bigg|_{\delta\mathbf{u}_k=0}\end{aligned} \tag{11}$$

where

$$\mathbf{G}_{\mathbf{x}_k} = \frac{\partial\left(\left(\left(\mathbf{x}_k^d \boxplus \boldsymbol{\delta}\right) \oplus \Delta t\mathbf{f}(\mathbf{x}_k^d, \mathbf{u}_k^d)\right) \boxminus \left(\mathbf{x}_k^d \oplus \Delta t\mathbf{f}(\mathbf{x}_k^d, \mathbf{u}_k^d)\right)\right)}{\partial \boldsymbol{\delta}}\bigg|_{\boldsymbol{\delta}=\mathbf{0}}$$

$$\mathbf{G}_{\mathbf{f}_k} = \frac{\partial\left(\left(\mathbf{x}_k^d \oplus \left(\Delta t\mathbf{f}(\mathbf{x}_k^d, \mathbf{u}_k^d) + \boldsymbol{\delta}\right)\right) \boxminus \left(\mathbf{x}_k^d \oplus \Delta t\mathbf{f}(\mathbf{x}_k^d, \mathbf{u}_k^d)\right)\right)}{\partial \boldsymbol{\delta}}\bigg|_{\boldsymbol{\delta}=\mathbf{0}}. \tag{12}$$

B. Error-State MPC Formulation

Based on the linearized system (9), the original on-manifold MPC problem in (6) can be transformed into a standard MPC problem in the Euclidean space as follows:

$$\begin{aligned}&\min_{\delta\mathbf{u}_k} \sum_{k=0}^{N-1} \left(\|\delta\mathbf{x}_k\|^2_{\mathbf{Q}_k} + \|\delta\mathbf{u}_k\|^2_{\mathbf{R}_k}\right) + \|\delta\mathbf{x}_N\|^2_{\mathbf{P}_N} \\ &\text{s.t.} \quad \delta\mathbf{x}_{k+1} = \mathbf{F}_{\mathbf{x}}\delta\mathbf{x}_k + \mathbf{F}_{\mathbf{u}}\delta\mathbf{u}_k, \quad \delta\mathbf{x}_0 = \delta\mathbf{x}_{\text{init}} \\ &\qquad \delta\mathbf{u}_k \in \delta\mathbb{U}_k, \quad k = 0, \ldots, N-1\end{aligned} \tag{13}$$

where $\delta\mathbf{x}_{\text{init}} = \mathbf{x}_{\text{init}} \boxminus \mathbf{x}_0^d$ and $\delta\mathbb{U}_k = \{\delta\mathbf{u} \in \mathbb{R}^m | \mathbf{u}_{\min} - \mathbf{u}_k^d \leq \delta\mathbf{u} \leq \mathbf{u}_{\max} - \mathbf{u}_k^d\}$ is the equivalent initial state and input constraints. The optimization problem in (13) is a standard quadratic programming (QP) problem in terms of $\delta\mathbf{U} = (\delta\mathbf{u}_0, \ldots, \delta\mathbf{u}_{N-1})$, which can be solved efficiently by existing QP solvers.

C. Minimum Parameterization and Singularity

Since the error-state system (8) is minimally parameterized, the resultant MPC formulation (13) does not have any redundant parameters when compared with the quaternion-based [19] or even rotation matrix-based approaches [21]. In fact, the error-state system (8) is an ordinary nonlinear system in the Euclidean space. Hence, the resultant MPC (13) can directly adopt design and tuning experiences from classical MPC approaches. For example, by adjusting the corresponding element in the penalty matrices $\mathbf{Q}_k$ and $\mathbf{R}_k$, we can easily reweight different state components (e.g., position, attitude, etc.) or different element in each component (e.g., roll, pitch, and yaw).

Albeit the minimal parameterization, the on-manifold MPC is also practically singularity free in the entire workspace, which is stated in Theorem 1 as follows.

Theorem 1: For any reference trajectory $\mathbf{x}_k^d \in \mathcal{M}$, the on-manifold MPC in (13) is singularity free if it bounds the state error within the neighbor of $\mathbf{x}_k^d$, i.e., $\delta\mathbf{x}_k \in \mathbf{B}_\epsilon^n(\mathbf{x}_k^d)$, at all time steps on the trajectory.

Proof: Because the state error are always within the neighborhood where the local coordinate charts (i.e., the operations $\boxplus/\boxminus$ equivalently) exist by the definition in (1), the error system and the resultant on-manifold MPC is singularity free at all time steps on the trajectory. Moreover, since local coordinate charts exist at any points on the manifold by the manifold definition, the on-manifold MPC holds the singularity-free property for any trajectories in the workspace.

Remarks: As previously mentioned in Section II-C, the neighborhood $\mathbf{B}_\epsilon^n(\mathbf{x})$ is often very large (e.g., for $\mathbb{R}^n$, $\mathbf{B}_\epsilon^n(\mathbf{x})$ is the entire space of $\mathbb{R}^n$; for $SO(3)$, $\mathbf{B}_\epsilon^n(\mathbf{x})$ is the entire ball with radius π excluding the sphere, and for $\mathcal{S}$, $\mathbf{B}_\epsilon^n(\mathbf{x})$ is the entire domain of the surface), the condition that the state error is bounded within this neighborhood can be easily satisfied given a stable controller (which it should always satisfy). Therefore, the on-manifold MPC controller is singularity free in practice. Since this holds for all reference trajectory, the on-manifold MPC controller is singularity free in the entire workspace. In contrast, existing minimally parameterized MPC methods (e.g., [2], [4], [16], and [17]) use Euler angles to parameterize the state, which is well-known singular at certain points in the workspace.

D. Generality and Stability

From the aforementioned process, by formalizing the two operations $\boxplus$ and $\oplus$, we have obtained a unified way to treat the manifold constraints in the system modeling (4), MPC formulation (6), linearization (10) and (11), and solving (13). Since local coordinate charts (hence, $\boxplus$) exist for all manifolds by the manifold definition, and that the operation $\oplus$ is a compact notation of the existing system state equation, the proposed framework is applicable to a wide class of manifolds. Nevertheless, our method requires the existence of the Jacobian matrices $\mathbf{G}_{\mathbf{x}}$ and $\mathbf{G}_{\mathbf{f}}$ as in (12), which usually requires differentiable manifolds.

The derived framework is compatible with different choices of the local coordinates (affecting $\boxplus$) and discretization method (affecting $\oplus$). Different $\boxplus$ and $\oplus$ would lead to different linearized system, and hence, affect the stability of the designed MPC. However, regardless of the choice of $\boxplus$ and $\oplus$, they all come to a minimally parameterized error system in the form of (8), which can be treated as a usual nonlinear system, enabling the borrow of mature NMPC design and analysis techniques in the existing literature [30], [31]. The results (e.g., stability analysis) are usually dependent on the particular system in consideration and the choices of $\boxplus$ and $\oplus$, and hence beyond the scope of this paper.

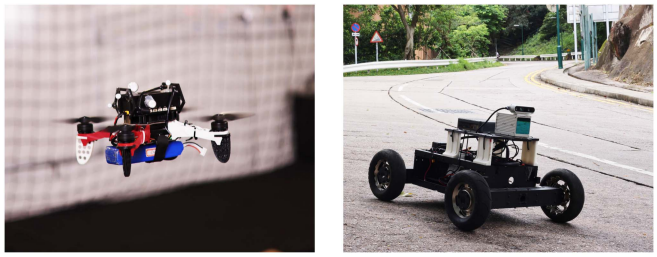
V. Isolation of Manifold Constraints

Besides of the generality to a wide class of differentiable manifolds, our proposed framework also allows the manifold constraints to be decoupled from the specific system. That is, the operations $\boxplus$ and $\oplus$ and the derivatives $\mathbf{G}_{\mathbf{x}}, \mathbf{G}_{\mathbf{f}}$ in (12) are only specific to the type of manifolds while being independent from system dynamics. Hence, they can be computed and saved beforehand. More specifically, the state equation in (6) breaks into the manifold-specific operation $\oplus$ and system-specific part $\mathbf{f}(\mathbf{x}, \mathbf{u})$, and the two matrices $\mathbf{F}_{\mathbf{x}}$ and $\mathbf{F}_{\mathbf{u}}$ in the linearized

TABLE II
MANIFOLD-SPECIFIC PARTS OF DIFFERENT PRIMITIVE MANIFOLDS

$\mathcal{M}$	$\mathbf{G}_{\mathbf{x}_k}$	$\mathbf{G}_{\mathbf{f}_k}$
$\mathbb{R}^n$	$\mathbf{I}_n$	$\mathbf{I}_n$
$\mathrm{SO}(2)$	1	1
$\mathrm{SO}(3)$	$\mathrm{Exp}\left(-\Delta t \mathbf{f}_k\right)$	$\mathbf{A}\left(\Delta t \mathbf{f}_k\right)^T$
$\mathcal{S}$	$\mathbf{I}_2$	$\mathbf{I}_2$

f_k denotes $\mathrm{f}(\mathrm{x}_k^{\mathrm{d}}, \mathrm{u}_k^{\mathrm{d}})$ for simplicity. Details of the calculation and Jacobian matrix $\mathrm{A}(\cdot)$ are given in Appendix.



(a) (b)

Fig. 5. Robotic platforms for experimental verification. (a) Quadrotor UAV with onboard computer, weight 1.1 kg. (b) DJI RoboMaster UGV with Livox Avia LiDAR.

error-state system (13) and resultant MPC solution (13) break into manifold-specific parts $\mathbf{G}_{\mathbf{x}}$ and $\mathbf{G}_{\mathbf{f}}$ and system-specific parts $\frac{\partial \mathbf{f}(\mathbf{x} \boxplus \delta\mathbf{x}, \mathbf{u})}{\partial \delta \mathbf{x}}|_{\delta\mathbf{x}=\mathbf{0}}$ and $\frac{\partial \mathbf{f}(\mathbf{x}, \mathbf{u} \boxplus \delta\mathbf{u})}{\partial \delta \mathbf{u}}|_{\delta\mathbf{u}=\mathbf{0}}$ as shown in (10) and (11).

When deploying the proposed MPC, the nice isolation property between the manifold constraints and system dynamics allows to only define the system-specific parts without repeating calculation of manifold constraints if the manifold is already defined, and automatically construct a large *compound manifolds* by defining and concatenating its simple *primitive manifolds*. The $\boxplus/\boxminus$ and $\oplus$ operations for commonly used simple manifolds and their manifold-specific parts are, respectively, summarized in Tables I and II. If a new primitive manifold is encountered, one could follow the procedures in Section II to define its $\boxplus\backslash\boxminus$ and $\oplus$ operations and the $\mathbf{G}_{\mathbf{x}}$ and $\mathbf{G}_{\mathbf{f}}$ defined in (12).

VI. APPLICATIONS AND EXPERIMENTAL RESULTS

To demonstrate the proposed MPC framework, we apply the design process and implementation to two real-time robotic systems (see Fig. 5): a quadrotor UAV on $\mathrm{SE}(3)$ and an UGV moving on a curved surface (i.e., $\mathcal{S} \times SO(2)$).

A. Application to Quadrotor UAVs

1) Experiment Setup: The algorithm have been implemented and evaluated on a quadrotor [see Fig. 5(a)] equipped with autopilot Pixhawk 4 (PX4) Mini and DJI Manifold-2 C on-board computer (quad-core Intel i7). The vehicle state is obtained by an EKF running on the autopilot with an external motion capture system (VICON). Algorithms and communications are implemented in the robot operating system. The OOQP solver [32] is deployed to solve the MPC problem in (13).

2) System Modeling: The UAV body frame is defined as forward-right-down. The system state consists of the UAV position in the inertial frame $\mathbf{p}$, the UAV velocity in the inertial frame $\mathbf{v}$, the orientation $\mathbf{R} \in SO(3)$ from the inertial frame to the body frame and the body angular rate $\boldsymbol{\omega}$. The resultant system state equation is

$$\dot{\mathbf{p}} = \mathbf{v}, \quad \mathbf{v} = \mathbf{g} - a_T \mathbf{R}\mathbf{e}_3, \quad \dot{\mathbf{R}} = \mathbf{R}\lfloor\boldsymbol{\omega}\rfloor \tag{14}$$

where a_T denotes the scalar thrust acceleration and $\mathbf{g}$ is the gravity vector with fixed length of $9.81\ \mathrm{m/s}^2$.

3) Canonical Representation: As explained in Section III, the quadrotor model in (14) can be discretized and cast into the canonical form (4), where

$$\mathcal{M} = \mathbb{R}^3 \times \mathbb{R}^3 \times \mathrm{SO}(3), \quad \dim(\mathcal{M}) = 9 \tag{15a}$$

$$\mathbf{x} = (\mathbf{p}\ \mathbf{v}\ \mathbf{R}) \in \mathcal{M}, \quad \mathbf{u} = \begin{bmatrix} a_T & \boldsymbol{\omega}^T \end{bmatrix}^T \in \mathbb{R}^4 \tag{15b}$$

$$\mathbf{f}(\mathbf{x}, \mathbf{u}) = \begin{bmatrix} \mathbf{v}^T & (\mathbf{g} - a_T \mathbf{R}\mathbf{e}_3)^T & \boldsymbol{\omega}^T \end{bmatrix}^T \in \mathbb{R}^9 \tag{15c}$$

the system-specific parts are

$$\left.\frac{\partial \mathbf{f}(\mathbf{x} \boxplus \delta\mathbf{x}, \mathbf{u})}{\partial \delta \mathbf{x}}\right|_{\delta\mathbf{x}=\mathbf{0}} = \begin{bmatrix} \mathbf{0} & \mathbf{I}_3 & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & a_T \mathbf{R}\lfloor \mathbf{e}_3 \rfloor \\ \mathbf{0} & \mathbf{0} & \mathbf{0} \end{bmatrix} \tag{16a}$$

$$\left.\frac{\partial \mathbf{f}(\mathbf{x}, \mathbf{u} \boxplus \delta\mathbf{u})}{\partial \delta \mathbf{u}}\right|_{\delta\mathbf{u}=\mathbf{0}} = \begin{bmatrix} \mathbf{0} & \mathbf{0} \\ -\mathbf{R}\mathbf{e}_3 & \mathbf{0} \\ \mathbf{0} & \mathbf{I}_3 \end{bmatrix}. \tag{16b}$$

As can be seen, we viewed the state manifold as a compound manifold (i.e., $\mathbb{R}^3 \times \mathbb{R}^3 \times SO(3)$) in order to leverage the already defined $\boxplus$ and $\oplus$ in Table I and derivatives $\mathbf{G}_{\mathbf{x}}$ and $\mathbf{G}_{\mathbf{f}}$ in Table II. Other choices, such as treating the manifold as an integrated one (i.e., $SE(3)$), can also fit into our proposed framework but its discussion is omitted due to the space limit.

4) Experimental Results: The implemented MPC has a predictive horizon $N = 8$ and runs on the onboard computer to calculate the command of thrust acceleration a_T and body angular rate $\boldsymbol{\omega}$ at 100 Hz. The thrust acceleration command is then mapped to the throttle command sent to the PX4 by $\mathrm{thr} = k_T a_T$, where the coefficient k_T is computed as $\mathrm{thr_h}/9.81$ with $\mathrm{thr_h}$ being the throttle at hovering. The body angular rate command computed by the MPC is sent to the angular rate controller, which runs on the PX4 at 250 Hz.

To demonstrate the tracking performance, a comparison study with an NMPC in [2] is conducted on its recommended simulator RotorS [33] with the model parameters of AscTec Firefly. The control action of the NMPC are thrust acceleration, roll angle, pitch angles, and yaw rate, which are then tracked by an underlying attitude controller. In order to make a fair comparison, the two controllers share the same angular rate controller, and all other parameters of each method are tuned to the best extent. The results are shown in Fig. 6. As can be seen, our proposed controller significantly outperforms NMPC in tracking accuracy for tracking a polynomial trajectory. This improvement is not

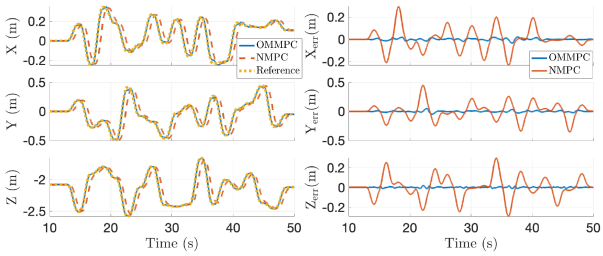


Fig. 6. Simulation comparison of our on-manifold MPC (OMMPC) and the NMPC proposed in [2] on an open-source simulator RotorsS [33]. (Left) Reference and actual trajectory. (Right) Trajectory tracking error.

surprising as our method utilizes the angular rate as the direct control, which avoids the response delay in the attitude loop in NMPC. Besides the tracking accuracy, attitude constraint that the roll or pitch angles must not exceed $\pm 45^\circ$ is set in NMPC to avoid singularities causing control failures, while there is no such constraint in our method as it is singularity free.

Furthermore, we test the proposed MPC scheme on real quadrotors on three more aggressive aerobatic trajectories as shown in Fig. 7: (a) a 3-D helix trajectory with flips, (b) a vertical circle with flips, and (c) a horizontal circle with speed linearly accelerating up to 5 m/s and maximum acceleration up to 2g. These trajectories are designed offline and the reference state and input are generated according to the quadtotor differential flatness [34]. For trajectories (a) and (b), the MPC parameters are set as $\mathbf{Q}_k = \text{diag}([15\,000, 15\,000, 15\,000, 40, 40, 40, 80, 80, 80])$, $\mathbf{R}_k = \text{diag}([0.5, 0.6\ 0.6, 0.6])$, and $\mathbf{P}_N = \mathbf{Q}_k$. For trajectory (c), to ensure a higher position tracking accuracy, we increase the weight on position errors by setting $\mathbf{Q}_k = \text{diag}([100\,000, 100\,000, 25\,000, 80, 80, 80, 200, 200, 100])$, while the rest parameters are kept the same.

The trajectory tracking results are detailed in Fig. 7. As shown in the last column, these trajectories require very large actuator efforts (i.e., collective thrust nearly 4 g and angular rate up to 800°/s) of fast changing, which are usually more challenging than those demonstrated in [2], [35], and [36] due to the motor saturation and delay. To track these trajectories, the UAV attitude has to change over 360° in either roll and pitch directions (see the fourth column). All these make the trajectories rather difficult to track. The tracking performance are shown in the second and third column for position and the fourth column for the attitude. As can be seen, the proposed MPC controller is able to track all these trajectories with small errors (0.08 m for position and 8° for attitude in average) despite the high aggressiveness. More importantly, our controller is singularity free in the whole workspace, allowing the same controller to be used in all these three tasks with over 360° attitude changes in both pitching and rolling directions, while the conventional MPC controllers like [2] based on Euler angles has to restrict the operation range of roll or pitch angle within $\pm 90^\circ$ to avoid singularities. For the horizontal circle trajectory, in Fig. 7(c), we additionally show its position error in relation to the speed in Fig. 8(a). As can be seen, the position error generally increases with the speed due to the increased rotor drag that is not modeled nor compensated in our modeling (14) (notice that our framework can also consider this rotor drag), but still, the overall error remains less than 0.1 m. In comparison, the PID controllers used in [37], although considered the rotor drag (and identified the related coefficients with quite some efforts), has a tracking error at 5 m/s more than 0.1 m even with a larger radius (1.8 m versus 1.3 m in ours) that requires less actuator efforts. Finally, Fig. 8(b) shows that the average computation of our MPC controller per control cycle. As can be seen, the computation time is merely 1.2 ms in average and 2 ms in the worst case, which are far less than the control period (i.e., 10 ms for the 100-Hz MPC) and can be well implemented for real-time robotic applications.

Remarks: The aforementioned quadrotor UAV experiment gives an example applying the proposed on-manifold MPC framework to a quadrotor system by simply describing the system state (15), deriving the system-specific derivatives (16) and leveraging the manifold-specific parts in Table II. As mentioned in Section IV, our method based on the error system maps the system state to the local coordinates at each point along the reference trajectory, leading to a minimally parameterized, singularity-free MPC controller able to track trajectories in the whole workspace. In contrast, other minimally parameterized quadrotor MPC methods (e.g., Euler angles in [2]) has to restrict the attitude range in order to avoid singularities. Moreover, the high computational efficiency of our method allows the MPC to calculate low-level control command (i.e., thrust acceleration and angular rates) at a high frequency (i.e., 100 Hz), contributing to fast control response and high tracking accuracy comparable to methods even with sophisticated aerodynamic modeling and compensation (e.g., [37]).

B. Application to UGVs Moving on Curved Surfaces

1) Experimental Setup: We implement the proposed MPC framework on the chassis of a DJI RoboMaster ground vehicle [see Fig. 5(b)], equipped with a Livox Avia LiDAR and a NUC (Intel i7) onboard computer. The Mecanum wheels are equipped with rubber tires such that the vehicle has no lateral motion. The robot provides interfaces for longitudinal speed and heading (i.e., yaw), which are the control input computed by the MPC controller. The vehicle's state is estimated by a real-time LiDAR-inertial Odometry FAST-LIO [38] running onboard. Both the FAST-LIO and MPC run on the same onboard computer at 50 Hz. The MPC horizon is set to 45. The penalty matrices are $\mathbf{Q}_k = \text{diag}([10, 10, 10])$, $\mathbf{R}_k = \text{diag}([0.5, 0.5])$ and $\mathbf{P}_N = \mathbf{Q}_k$. We conduct the experiment on a road with significant height variation (see Figs. 1(d) and 10).

2) System Modeling: Shown in Fig. 9, the UGV state consists of the vehicle position $\mathbf{p} \in \mathbb{R}^3$ in the inertial frame $(\mathbf{x}, \mathbf{y}, \mathbf{z})$ and its attitude represented by the body frame $(\mathbf{x}_b, \mathbf{y}_b, \mathbf{z}_b)$. Note that since the UGV is restricted on the surface with the $\mathbf{z}_b$ always pointing to the surface normal, there is a one to one map between the body frame and the frame $(\tilde{\mathbf{x}}, \tilde{\mathbf{y}}, \mathbf{z})$, which undergoes only a rotation around $\mathbf{z}$ from the inertial frame and can, hence, be represented by a rotation $\mathbf{R} \in SO(2)$. As a consequence, the

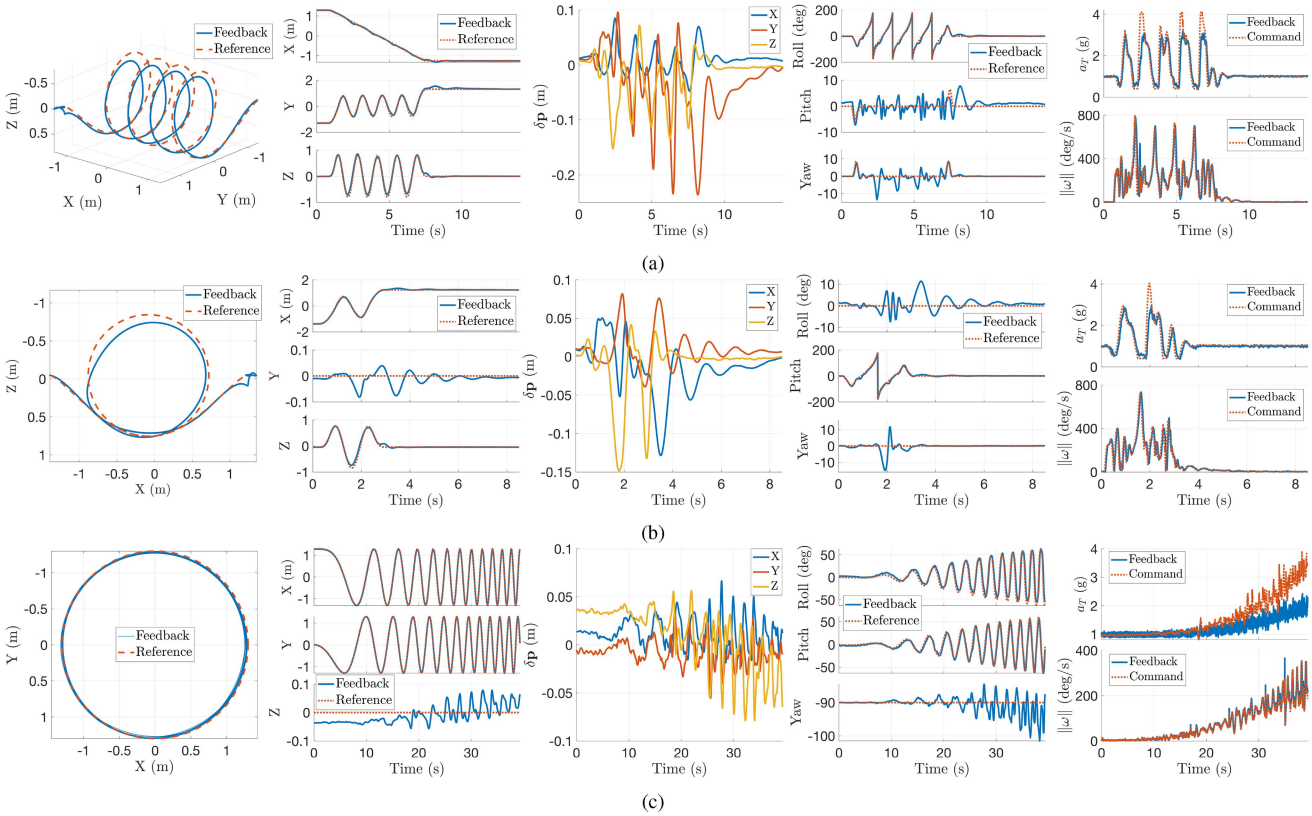


Fig. 7. Quadrotor aerobatics. (a) 3-D helix trajectory with flips. (b) Vertical circle with flips. (c) Horizontal circle with speed linearly increasing from zero to 5 m/s. From columns left to right: 3-D trajectory, position response, position tracking error, attitude response in Euler angles, and control efforts (consisting of the thrust acceleration normalized by the gravitational acceleration, and angular rate displayed by norm with unit degree per second). For all subplots except the third column, the solid blue line denotes the feedback state (or measured control action) and the dashed red line denotes the reference state (or control command). For the control action, the feedback thrust acceleration and angular rate are measured by the accelerometer and gyroscope of the onboard inertial measurement unit (IMU).

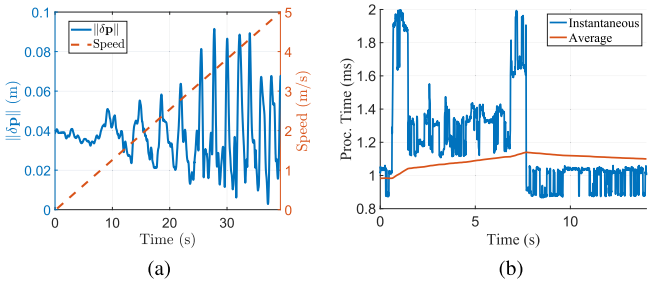


Fig. 8. Tracking accuracy and computational efficiency. (a) Position tracking error and flight speed in the horizontal circling flight. (b) Computation time per control cycle in the 3-D helix tracking flight.

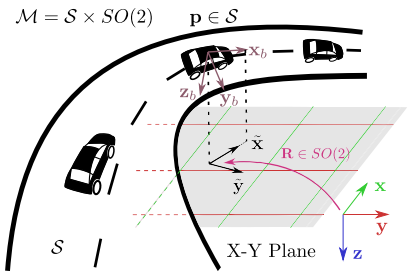


Fig. 9. Illustration of compound manifold $\mathcal{S} \times SO(2)$.

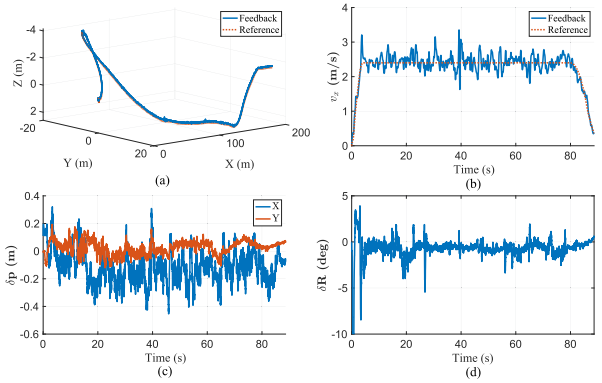


Fig. 10. Control performance of the UGV moving on a curved surface with a constant speed of 2.4 m/s. (a) 3-D trajectory. (b) Longitudinal speed v_x. (c) Position tracking error. (d) Orientation tracking error.

robot state is $\mathbf{x} = (\mathbf{p}, \mathbf{R}) \in \mathcal{S} \times SO(2)$. The control action of the robot is the longitudinal velocity v_x and the yaw rate ω_z.

Denote $\mathbf{x}_b(\mathbf{p}, \mathbf{R})$ the map from the frame $(\tilde{\mathbf{x}}, \tilde{\mathbf{y}}, \tilde{\mathbf{z}})$ represented by $\mathbf{R} \in SO(2)$ to the body axis $\mathbf{x}_b$ and denote $\mathbf{z}_b(\mathbf{p})$ the body axis $\mathbf{z}_b$, which always points to the surface normal, and hence, depends on the robot's location on the surface. Then, the robot's

model in continuous time is

$$\dot{\mathbf{p}} = \mathbf{x}_b(\mathbf{p},\mathbf{R})v_x, \quad \dot{\mathbf{R}} = \mathbf{R}\lfloor \mathbf{z}^T\mathbf{z}_b(\mathbf{p})\omega_z \rfloor \tag{17}$$

where the angular rate $\omega_z\mathbf{z}_b(\mathbf{p})$ is projected to $\mathbf{z}$ to drive $\mathbf{R}$.

3) Canonical Representation: We discretize the system (17) into the canonical form (4), where manifolds $\mathcal{S}$ and $SO(2)$ are defined, respectively, in Section II as follows:

$$\mathcal{M} = \mathcal{S}\times SO(2), \quad \dim(\mathcal{M}) = 3 \tag{18a}$$

$$\mathbf{x} = (\mathbf{p}\ \mathbf{R}) \in \mathcal{M}, \quad \mathbf{u} = \begin{bmatrix} v_x & \omega_z \end{bmatrix} \in \mathbb{R}^2 \tag{18b}$$

$$\mathbf{f}(\mathbf{x},\mathbf{u}) = \begin{bmatrix} \mathbf{E}_{12}^T\mathbf{x}_b(\mathbf{p},\mathbf{R})v_x \\ \mathbf{z}^T\mathbf{z}_b(\mathbf{p})\omega_z \end{bmatrix} = \begin{bmatrix} \alpha\mathbf{R}\mathbb{e}_1 v_x \\ \beta\omega_z \end{bmatrix} \in \mathbb{R}^3 \tag{18c}$$

where

$$\alpha(\mathbf{p},\mathbf{R}) = \frac{1}{\sqrt{1+(\mathbf{h}^T\mathbf{R}\mathbb{e}_1)^2}}, \ \beta(\mathbf{p}) = \frac{1}{\sqrt{1+\mathbf{h}^T\mathbf{h}}} \tag{19a}$$

$$\mathbf{h} = \begin{bmatrix} F_x' & F_y' \end{bmatrix}^T \in \mathbb{R}^2, \ \mathbb{e}_1 = \begin{bmatrix} 1 & 0 \end{bmatrix}^T, \ \mathbb{e}_2 = \begin{bmatrix} 0 & 1 \end{bmatrix}^T. \tag{19b}$$

The manifold-specific parts of the system matrices are referred to Table II, and the system-specific parts are given as follows:

$$\frac{\partial \mathbf{f}(\mathbf{x}\boxplus\delta\mathbf{x},\mathbf{u})}{\partial\delta\mathbf{x}} = \begin{bmatrix} \mathbf{R}\mathbb{e}_1 v_x \frac{\partial\alpha}{\partial\delta\mathbf{p}} & \mathbf{R}\left(\frac{\partial\alpha}{\partial\delta\mathbf{R}}\mathbb{e}_1 + \alpha\mathbb{e}_2\right)v_x \\ \frac{\partial\beta}{\partial\delta\mathbf{p}}\omega_z & 0 \end{bmatrix} \in \mathbb{R}^{3\times 3} \tag{20a}$$

$$\frac{\partial \mathbf{f}(\mathbf{x},\mathbf{u}\boxplus\delta\mathbf{u})}{\partial\delta\mathbf{u}} = \begin{bmatrix} \alpha\mathbf{R}\mathbb{e}_1 & \mathbf{0}_{2\times 1} \\ 0 & \beta \end{bmatrix} \in \mathbb{R}^{3\times 2} \tag{20b}$$

where

$$\frac{\partial\alpha}{\partial\delta\mathbf{p}} = -\frac{\mathbf{h}^T\mathbf{R}\mathbb{e}_1\mathbb{e}_1^T\mathbf{R}^T}{\sqrt{(1+(\mathbf{h}^T\mathbf{R}\mathbb{e}_1)^2)^3}}\frac{\partial\mathbf{h}(\mathbf{p}\boxplus\delta\mathbf{p})}{\partial\delta\mathbf{p}} \in \mathbb{R}^{1\times 2} \tag{21a}$$

$$\frac{\partial\alpha}{\partial\delta\mathbf{R}} = -\frac{\mathbb{e}_1^T\mathbf{R}^T\mathbf{h}\mathbf{h}^T}{\sqrt{(1+(\mathbf{h}^T\mathbf{R}\mathbb{e}_1)^2)^3}}\frac{\partial((\mathbf{R}\boxplus\delta\mathbf{R})\mathbb{e}_1)}{\partial\delta\mathbf{R}} \in \mathbb{R} \tag{21b}$$

$$\frac{\partial\beta}{\partial\delta\mathbf{p}} = -\frac{\mathbf{h}^T}{\sqrt{(1+\mathbf{h}^T\mathbf{h})^3}}\frac{\partial\mathbf{h}(\mathbf{p}\boxplus\delta\mathbf{p})}{\partial\delta\mathbf{p}} \in \mathbb{R}^{1\times 2} \tag{21c}$$

$$\frac{\partial\mathbf{h}(\mathbf{p}\boxplus\delta\mathbf{p})}{\partial\delta\mathbf{p}} = \begin{bmatrix} F_{xx}'' & F_{xy}'' \\ F_{yx}'' & F_{yy}'' \end{bmatrix}, \frac{\partial((\mathbf{R}\boxplus\delta\mathbf{R})\mathbb{e}_1)}{\partial\delta\mathbf{R}} = \mathbf{R}\mathbb{e}_2. \tag{21d}$$

4) Trajectory Generation and Surface Approximation: To generate a reference trajectory on the ground surface, we first manually control the ground vehicle to travel along the road and run FAST-LIO to build a 3-D point cloud map of the traveled environment [see Fig. 1(d)]. Then, based on the traveled trajectory, a 2-D smooth path on the X–Y plane is generated. This 2-D path is then mapped to the ground surface to generate the desired trajectory $\mathbf{p}^d \in \mathcal{S}$ with a constant longitudinal speed v_x^d. With the desired trajectory $\mathbf{p}^d$, we solve for the body axis $\mathbf{x}_b$ from the first equation of (17), the body axis $\mathbf{y}_b$ by fitting a ground surface at the current point of $\mathbf{p}^d$ from the point cloud map, and the body axis $\mathbf{z}_b$ by $\mathbf{x}_b\times\mathbf{y}_b$. This leads to the desired attitude $\mathbf{R}^d$, from which we further determine the desired yaw rate ω_z^d from the second equation of (17). Besides the yaw rate, the other control input is the longitudinal speed, whose desired value v_x^d is specified already.

5) Experimental Results: As shown in Fig. 1(d), Fig. 10(a), and the supplemental video, with our MPC controller, the UGV is able to move on the curved ground surface smoothly and track the desired trajectory on the surface with a relatively constant longitudinal speed as expected. The average tracking error in the position, orientation, and longitudinal speed are 0.2 m, 0.7°, and 0.03 m/s, respectively, despite the significant height variation, sharp turns, curved ground, and uneven road conditions with various disturbances.

Remarks: This experiment verifies that the proposed framework can be applied to systems evolving on non-Lie group manifold by following the same procedure, i.e., defining the system manifold $\mathcal{S}\times SO(2)$ by $\boxplus$, $\oplus$, and providing system-specific parts (20).

VII. Conclusion

This article proposed a unified MPC framework for trajectory tracking on manifolds for robotic systems. A canonical, symbolic, minimal-parameterization, singularity-free framework with modeling, error-system derivation, linearization, and control was established. The proposed algorithm was implemented on quadrotor UAV with aggressive maneuvers and UGV moving on curved surfaces, showing a level of generality and high tracking accuracy.

Appendix

We derive the manifold-specific parts $\mathbf{G}_\mathbf{x}$ and $\mathbf{G}_\mathbf{f}$ defined in (12) of the primitive manifolds shown in Table II. Denote $\mathbf{v} = \Delta t\mathbf{f}(\mathbf{x}_k^d,\mathbf{u}_k^d)$ for simplicity.

A. Euclidean Space $\mathbb{R}^n$

$$\mathbf{G}_{\mathbf{x}_k} = \frac{\partial\left(\mathbf{x}_k^d+\boldsymbol{\delta}+\mathbf{v}-\left(\mathbf{x}_k^d+\mathbf{v}\right)\right)}{\partial\boldsymbol{\delta}} = \mathbf{I}_n \tag{22a}$$

$$\mathbf{G}_{\mathbf{f}_k} = \frac{\partial\left(\mathbf{x}_k^d+\mathbf{v}+\boldsymbol{\delta}-\left(\mathbf{x}_k^d+\mathbf{v}\right)\right)}{\partial\boldsymbol{\delta}} = \mathbf{I}_n. \tag{22b}$$

B. Special Orthogonal Group $SO(3)$

$$\begin{aligned}\mathbf{G}_{\mathbf{x}_k} &= \frac{\partial}{\partial\boldsymbol{\delta}}\mathrm{Log}\left(\left(\mathbf{x}_k^d\cdot\mathrm{Exp}(\mathbf{v})\right)^{-1}\mathbf{x}_k^d\cdot\mathrm{Exp}(\boldsymbol{\delta})\cdot\mathrm{Exp}(\mathbf{v})\right) \\ &\approx \frac{\partial}{\partial\boldsymbol{\delta}}\mathrm{Log}\left(\mathrm{Exp}\left(\mathrm{Exp}(\mathbf{v})^{-1}\boldsymbol{\delta}\right)\right) = \frac{\partial}{\partial\boldsymbol{\delta}}\left(\mathrm{Exp}(\mathbf{v})^{-1}\boldsymbol{\delta}\right) \\ &= \mathrm{Exp}(-\mathbf{v})\end{aligned} \tag{23a}$$

$$\begin{aligned}\mathbf{G}_{\mathbf{f}_k} &= \frac{\partial}{\partial \boldsymbol{\delta}} \mathrm{Log}\left(\left(\mathbf{x}_k^d \cdot \mathrm{Exp}(\mathbf{v})\right)^{-1} \mathbf{x}_k^d \cdot \mathrm{Exp}(\mathbf{v}+\boldsymbol{\delta})\right) \\ &= \frac{\partial}{\partial \boldsymbol{\delta}} \mathrm{Log}\left(\mathrm{Exp}(-\mathbf{v}) \cdot \mathrm{Exp}(\mathbf{v}+\boldsymbol{\delta})\right) = \frac{\partial}{\partial \boldsymbol{\delta}} \left(\mathbf{A}(\mathbf{v})^T \boldsymbol{\delta}\right) \\ &= \mathbf{A}(\mathbf{v})^T \end{aligned} \tag{23b}$$

where

$$\mathbf{A}(\boldsymbol{\theta}) = \mathbf{I}_3 + \left(\frac{1-\cos\|\boldsymbol{\theta}\|}{\|\boldsymbol{\theta}\|}\right)\frac{\lfloor\boldsymbol{\theta}\rfloor}{\|\boldsymbol{\theta}\|} + \left(1-\frac{\sin\|\boldsymbol{\theta}\|}{\|\boldsymbol{\theta}\|}\right)\frac{\lfloor\boldsymbol{\theta}\rfloor^2}{\|\boldsymbol{\theta}\|^2}. \tag{24}$$

C. Special Orthogonal Group $SO(2)$

$$\begin{aligned}\mathbf{G}_{\mathbf{x}_k} &= \frac{\partial \mathrm{Log}\left(\left(\mathbf{x}_k^d \cdot \mathrm{Exp}(v)\right)^{-1} \mathbf{x}_k^d \cdot \mathrm{Exp}(\delta) \cdot \mathrm{Exp}(v)\right)}{\partial \delta} \\ &\approx \frac{\partial \mathrm{Log}\left(\mathrm{Exp}(\delta)\right)}{\partial \delta} = 1 \end{aligned} \tag{25a}$$

$$\begin{aligned}\mathbf{G}_{\mathbf{f}_k} &\approx \frac{\partial \mathrm{Log}\left(\left(\mathbf{x}_k^d \mathrm{Exp}(v)\right)^{-1} \mathbf{x}_k^d \mathrm{Exp}(v+\delta)\right)}{\partial \delta} \\ &\approx \frac{\partial \mathrm{Log}\left(\mathrm{Exp}(\delta)\right)}{\partial \delta} = 1. \end{aligned} \tag{25b}$$

D. Example 3: 2-D Surface $\mathcal{S}$

$$\begin{aligned}\mathbf{G}_{\mathbf{x}_k} &= \frac{\partial}{\partial \boldsymbol{\delta}} \mathbf{E}_{12}^T \left(\left(\mathbf{x}_k^d \boxplus \boldsymbol{\delta}\right) \oplus \mathbf{v} - \left(\mathbf{x}_k^d \oplus \mathbf{v}\right)\right) \\ &= \frac{\partial}{\partial \boldsymbol{\delta}} \left(\mathbf{E}_{12}^T \begin{bmatrix} \mathbf{E}_{12}^T \mathbf{x}_k^d + \boldsymbol{\delta} \\ F\left(\mathbf{E}_{12}^T \mathbf{x}_k^d + \boldsymbol{\delta}\right) \end{bmatrix} - \mathbf{E}_{12}^T \mathbf{x}_k^d \right) \\ &= \frac{\partial}{\partial \boldsymbol{\delta}} \left(\mathbf{E}_{12}^T \mathbf{x}_k^d + \boldsymbol{\delta} - \mathbf{E}_{12}^T \mathbf{x}_k^d\right) = \mathbf{I}_2 \\ \mathbf{G}_{\mathbf{f}_k} &= \frac{\partial}{\partial \boldsymbol{\delta}} \mathbf{E}_{12}^T \left(\mathbf{x}_k^d \oplus (\mathbf{v}+\boldsymbol{\delta}) - \left(\mathbf{x}_k^d \oplus \mathbf{v}\right)\right) \\ &= \frac{\partial}{\partial \boldsymbol{\delta}} \mathbf{E}_{12}^T \left(\begin{bmatrix} \mathbf{E}_{12}^T \mathbf{x}_k^d + \mathbf{v} + \boldsymbol{\delta} \\ F\left(\mathbf{E}_{12}^T \mathbf{x}_k^d + \mathbf{v} + \boldsymbol{\delta}\right) \end{bmatrix} - \begin{bmatrix} \mathbf{E}_{12}^T \mathbf{x}_k^d + \mathbf{v} \\ F\left(\mathbf{E}_{12}^T \mathbf{x}_k^d + \mathbf{v}\right) \end{bmatrix} \right) \\ &= \frac{\partial}{\partial \boldsymbol{\delta}} \left(\mathbf{E}_{12}^T \mathbf{x}_k^d + \mathbf{v} + \boldsymbol{\delta} - \mathbf{E}_{12}^T \mathbf{x}_k^d - \mathbf{v}\right) = \mathbf{I}_2. \end{aligned}$$

REFERENCES

[1] G. Williams, P. Drews, B. Goldfain, J. M. Rehg, and E. A. Theodorou, "Information-theoretic model predictive control: Theory and applications to autonomous driving," *IEEE Trans. Robot.*, vol. 34, no. 6, pp. 1603–1622, Dec. 2018.

[2] M. Kamel, M. Burri, and R. Siegwart, "Linear vs nonlinear MPC for trajectory tracking applied to rotary wing micro aerial vehicles," *IFAC-PapersOnLine*, vol. 50, no. 1, pp. 3463–3469, 2017.

[3] M. V. Minniti, F. Farshidian, R. Grandia, and M. Hutter, "Whole-body MPC for a dynamically stable mobile manipulator," *IEEE Robot. Automat. Lett.*, vol. 4, no. 4, pp. 3687–3694, Oct. 2019.

[4] M. Neunert et al., "Whole-body nonlinear model predictive control through contacts for quadrupeds," *IEEE Robot. Automat. Lett.*, vol. 3, no. 3, pp. 1458–1465, Jul. 2018.

[5] S. Kuindersma et al., "Optimization-based locomotion planning, estimation, and control design for the atlas humanoid robot," *Auton. Robots*, vol. 40, no. 3, pp. 429–455, 2016.

[6] B. Lindqvist, S. S. Mansouri, A.-a. Agha-mohammadi, and G. Nikolakopoulos, "Nonlinear MPC for collision avoidance and control of UAVs with dynamic obstacles," *IEEE Robot. Automat. Lett.*, vol. 5, no. 4, pp. 6001–6008, Oct. 2020.

[7] D. Falanga, P. Foehn, P. Lu, and D. Scaramuzza, "Pampc: Perception-aware model predictive control for quadrotors," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2018, pp. 1–8.

[8] C. E. Luis, M. Vukosavljev, and A. P. Schoellig, "Online trajectory generation with distributed model predictive control for multi-robot motion planning," *IEEE Robot. Automat. Lett.*, vol. 5, no. 2, pp. 604–611, Apr. 2020.

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

[10] R. M. Murray, Z. Li, and S. S. Sastry, *A Mathematical Introduction to Robotic Manipulation*. Boca Raton, FL, USA: CRC Press, 2017.

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

[12] M. Zhang, X. Zuo, Y. Chen, Y. Liu, and M. Li, "Pose estimation for ground robots: On manifold representation, integration, reparameterization, and optimization," *IEEE Trans. Robot.*, vol. 37, no. 4, pp. 1081–1099, Aug. 2021.

[13] A. Altan, "Performance of metaheuristic optimization algorithms based on swarm intelligence in attitude and altitude control of unmanned aerial vehicle for path following," in *Proc. 4th Int. Symp. Multidisciplinary Stud. Innov. Technol.*, 2020, pp. 1–6.

[14] E. Belge, A. Altan, and R. Hacıoğlu, "Metaheuristic optimization-based path planning and tracking of quadcopter for payload hold-release mission," *Electronics*, vol. 11, no. 8, 2022, Art. no. 1208.

[15] P. Bouffard, A. Aswani, and C. Tomlin, "Learning-based model predictive control on a quadrotor: Onboard implementation and experimental results," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2012, pp. 279–284.

[16] H. Nguyen, M. Kamel, K. Alexis, and R. Siegwart, "Model predictive control for micro aerial vehicles: A survey," in *Proc. Eur. Control Conf.*, 2021, pp. 1556–1563.

[17] J.-P. Sleiman, F. Farshidian, M. V. Minniti, and M. Hutter, "A unified MPC framework for whole-body dynamic locomotion and manipulation," *IEEE Robot. Automat. Lett.*, vol. 6, no. 3, pp. 4688–4695, Jul. 2021.

[18] A. Altan and R. Hacıoğlu, "Model predictive control of three-axis gimbal system mounted on UAV for real-time target tracking under external disturbances," *Mech. Syst. Signal Process.*, vol. 138, 2020, Art. no. 106548.

[19] S. Sun, A. Romero, P. Foehn, E. Kaufmann, and D. Scaramuzza, "A comparative study of nonlinear MPC and differential-flatness-based control for quadrotor agile flight," *IEEE Trans. Robot.*, pp. 1–17, 2022, doi: 10.1109/TRO.2022.3177279.

[20] 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.

[21] U. V. Kalabić, R. Gupta, S. Di Cairano, A. M. Bloch, and I. V. Kolmanovsky, "MPC on manifolds with an application to the control of spacecraft attitude on SO (3)," *Automatica*, vol. 76, pp. 293–300, 2017.

[22] G. Lu and F. Zhang, "IMU-based attitude estimation in the presence of narrow-band noise," *IEEE/ASME Trans. Mechatronics*, vol. 24, no. 2, pp. 841–852, Apr. 2019.

[23] F. L. Markley, "Attitude error representations for Kalman filtering," *J. Guid., Control, Dyn.*, vol. 26, no. 2, pp. 311–317, 2003.

[24] S. Hong, J.-H. Kim, and H.-W. Park, "Real-time constrained nonlinear model predictive control on SO (3) for dynamic legged locomotion," in *Proc. Int. Conf. Intell. Robots Syst.*, 2020, pp. 972–979.

[25] A. D. Lewis and D. R. Tyner, "Geometric Jacobian linearization and LQR theory," *J. Geometric Mechanics*, vol. 2, no. 4, pp. 397–440, 2010.

[26] D. He, W. Xu, and F. Zhang, "Kalman filters on differentiable manifolds," 2021, *arXiv:2102.03804*.

[27] W. M. Boothby, *An Introduction to Differentiable Manifolds and Riemannian Geometry*. Amsterdam, The Netherlands: Academic, 1986.

[28] C. Hertzberg, R. Wagner, U. Frese, and L. Schröder, "Integrating generic sensor fusion algorithms with sound state representations through encapsulation of manifolds," *Inf. Fusion*, vol. 14, no. 1, pp. 57–77, 2013.

[29] M. Bloesch, M. Burri, S. Omari, M. Hutter, and R. Siegwart, "Iterated extended Kalman filter based visual-inertial odometry using direct photometric feedback," *Int. J. Robot. Res.*, vol. 36, no. 10, pp. 1053–1072, 2017.

[30] F. Borrelli, A. Bemporad, and M. Morari, *Predictive Control for Linear and Hybrid Systems*. New York, NY, USA: Cambridge Univ. Press, 2017.
[31] D. Q. Mayne, J. B. Rawlings, C. V. Rao, and P. O. Scokaert, "Constrained model predictive control: Stability and optimality," *Automatica*, vol. 36, no. 6, pp. 789–814, 2000.
[32] E. M. Gertz and S. J. Wright, "Object-oriented software for quadratic programming," *ACM Trans. Math. Softw.*, vol. 29, no. 1, pp. 58–81, 2003.
[33] F. Furrer, M. Burri, M. Achtelik, and R. Siegwart, "RotorS–A modular gazebo MAV simulator framework," in *Robot Operating System (ROS): The Complete Reference*, vol. 1, Cham, Switzerland: Springer, 2016, pp. 595–625.
[34] D. Mellinger and V. Kumar, "Minimum snap trajectory generation and control for quadrotors," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2011, pp. 2520–2525.
[35] M. Bangura and R. Mahony, "Real-time model predictive control for quadrotors," *IFAC Proc. Vols.*, vol. 47, no. 3, pp. 11773–11780, 2014.
[36] P. Foehn and D. Scaramuzza, "Onboard state dependent LQR for agile quadrotors," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2018, pp. 6566–6572.
[37] M. Faessler, A. Franchi, and D. Scaramuzza, "Differential flatness of quadrotor dynamics subject to rotor drag for accurate tracking of high-speed trajectories," *IEEE Robot. Automat. Lett.*, vol. 3, no. 2, pp. 620–626, Apr. 2018.
[38] W. Xu and F. Zhang, "FAST-LIO: A fast, robust LiDAR-inertial odometry package by tightly-coupled iterated Kalman filter," *IEEE Robot. Automat. Lett.*, vol. 6, no. 2, pp. 3317–3324, Apr. 2021.

Guozheng Lu received the B.Eng. degree in automation from the Harbin Institute of Technology, Harbin, China, in 2016. He is currently working toward the Ph.D. degree with the Department of Mechanical Engineering, University of Hong Kong, Hong Kong.

From 2016 to 2019, he was an Engineer with Dajiang Innovations. His research interests include robotic control and trajectory generation.

Wei Xu received the B.S. and M.S. degrees in aerial vehicle design from Beihang University, Beijing, China, in 2015 and 2018, respectively. He is currently working toward the Ph.D. degree with the Department of Mechanical Engineering, University of Hong Kong, Hong Kong.

His research interests include aerial robots control, sensor fusion, and LiDAR SLAM.

Fu Zhang (Member, IEEE) received the B.E. degree in automation from the University of Science and Technology of China, Hefei, China, in 2011, and the Ph.D. degree in mechanical engineering, from the University of California, Berkeley, CA, USA, in 2015.

He joined the University of Hong Kong, Hong Kong, in 2018, where he is currently an Assistant Professor in mechanical engineering. His research interests include mechatronics and robotics, with focus on unmanned aerial vehicle design, control, and LiDAR-based navigation.