IEEE Robotics and Automation Letters (RAL) paper presented at
2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
October 14-18, 2024. Abu Dhabi, UAE

GMPC: Geometric Model Predictive Control for Wheeled Mobile Robot Trajectory Tracking

Jiawei Tang[1], Shuang Wu[2], Bo Lan[1], Yahui Dong[1], Yuqiang Jin[3], Guangjian Tian[2], Wen-An Zhang[3], *Senior Member, IEEE*, and Ling Shi[1], *Fellow, IEEE*

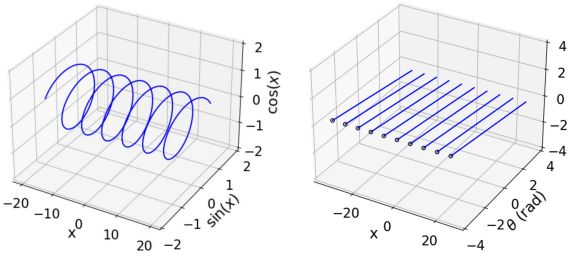
***Abstract*—The configuration of most robotic systems lies in continuous transformation groups. However, in mobile robot trajectory tracking, many recent works still naively utilize optimization methods for elements in vector space without considering the manifold constraint of the robot configuration. In this letter, we propose a geometric model predictive control (MPC) method for wheeled mobile robot trajectory tracking. We first derive the error dynamics of the wheeled mobile robot trajectory tracking by considering its manifold constraint and kinematic constraint simultaneously. After that, we utilize the relationship between the Lie group and Lie algebra to convexify the tracking control problem, which enables us to solve the problem efficiently. Thanks to the Lie group formulation, our method tracks the trajectory more smoothly than existing nonlinear MPC. Simulations and physical experiments verify the effectiveness of our proposed methods. Our pure Python-based simulation platform is publicly available to benefit further research in the community.**

***Index Terms*—motion control, autonomous agents**

I. Introduction

In recent decades, the wheeled mobile robot (WMR) has been widely applied in many fields, such as autonomous vehicles, intelligent warehouses, and smart agriculture. The widespread adoption of WMRs can be attributed to their remarkable flexibility and efficiency advantages. With technological advancements, WMRs can navigate different terrains, automate complicated tasks, and improve overall productivity. Recent research breakthroughs in robotics further contribute to the growing interest in WMRs [1]–[3].

The WMR belongs to the class of nonholonomic systems, characterized by a set of nonintegrable first-order differential constraints. These constraints arise from the assumption that wheeled robots move without slipping. Consequently, the nonholonomic constraint of the WMR can be visualized as a situation where the mobile robot cannot undergo lateral translations. According to Brockett's theorem, nonholonomic systems cannot be stabilized solely through smooth time-invariant feedback control laws [4]. As a result, developing an appropriate controller for achieving trajectory tracking of nonholonomic WMRs is generally a challenging task.



(a) $SO(2)$, Lie group (b) θ, Euler angle

Fig. 1: Difference between Lie group and Euler angle representation: The mapping from variable x to Euler angles exhibits discontinuity, whereas the mapping from x to the special orthogonal group $SO(2)$ (an isomorphism of the complex circle group e^{ix}) remains continuous.

To address the trajectory tracking problem, numerous control schemes have been proposed in the existing literature. Time-varying nonlinear state-feedback controllers have been proposed in [4]–[6], and dynamic linear feedback controllers can be found in [7]–[9]. Recently, studies on WMR trajectory tracking with model predictive control (MPC) have appeared. The predictive nature of MPC allows for real-time adaptation and adjustment, making it particularly suitable for dynamic and uncertain environments. The rapid development of computation power benefits the wide dissemination of MPC-based methods [10]–[14]. While MPC has shown promising results in modern robotics, a fundamental difficulty lies in effectively incorporating the manifold constraint of the robot configuration into the MPC framework. The configuration of WMR naturally lies in the continuous transformation group (Lie group) that does not comply with algebraic operations in a vector space [15]. For example, the superposition principle does not hold in the matrix Lie group. Hence, theoretical results developed for MPC in vector space cannot be directly extended to the one in the Lie group. Moreover, when handling the orientation, the singularity issue of the Euler angle and the double-cover issue of the quaternion introduce additional challenges for robotic control and optimization [16].

The major challenge of Lie group MPC comes from the differential geometry calculus. To deal with the manifold constraints, Jackson et.al. [16] developed a modified Newton method for the quaternion, and Alcan et.al. [17] developed a modified differential dynamic programming for the matrix Lie group. Besides, Lu et.al. [18] developed on-manifold MPC

Manuscript received: November 11, 2023; Revised: February 5, 2024; Accepted: March 7, 2024. This paper was recommended for publication by Editor Ashis Banerjee upon evaluation of the Associate Editor and Reviewers' comments. The work of Yuqiang Jin and Wen-An Zhang was supported by the National Natural Science Foundation of China under Grant No. 62173305.

[1]Jiawei Tang, Bo Lan, Yahui Dong, and Ling Shi are with the Department of Electronic and Computer Engineering, Hong Kong University of Science and Technology, Clear Water Bay, Hong Kong SAR (email: jtangas@connect.ust.hk; blanaa@connect.ust.hk; ydongbb@connect.ust.hk; eesling@ust.hk).

[2]Shuang Wu and Guangjian Tian are with the Noah's Ark Lab, Huawei Hong Kong Research Center, Sha Tin, Hong Kong SAR (email: wushuang.noah@huawei.com; Tian.Guangjian@huawei.com).

[3]Yuqiang Jin and Wen-An Zhang are with the College of Information Engineering, Zhejiang University of Technology, Hangzhou, China (email: jinyqiang98@gmail.com; wazhang@zjut.edu.cn).

Digital Object Identifier (DOI): see top of this page.

for trajectory tracking. These methods rely on concepts from differential geometry to derive complex derivative calculations and some need to be implemented in customized optimization solvers. Beyond that, many recent works on mobile robots [10]–[14] still naively employ optimization methods for elements in vector space without considering the manifold constraints of the robot configuration.

In this letter, we propose a novel geometric MPC (GMPC) method for trajectory tracking of WMRs using the matrix Lie group. The continuous nature of the matrix Lie group allows for the generation of smoother trajectories compared to the Euler angle-based method (as depicted in Fig. 1). As motivated by the recent research breakthrough on legged robot control [19], we can explore the relationship of the Lie group and the corresponding Lie algebra to convert the state space of MPC from the Lie group to the vector space. Besides, under the framework of the Lie theory, the nonholonomic kinematic constraint of WMR can be easily formulated as a linear mapping from the control input to the velocity of WMR. These advantages enable us to avoid complicated calculations of Lie group derivatives for WMR while handling the WMR's manifold constraint and kinematic constraint simultaneously. Our contributions are multi-fold.

1) We derive the continuous-time error dynamics for the WMR trajectory tracking by considering its manifold constraint and kinematic constraint simultaneously. This derivation enables us to formulate the trajectory tracking as an error-dynamic optimal control problem.

2) We propose different linearization schemes for the error dynamics to convexification. We show the rationale behind why the proposed linearization scheme is suitable for trajectory tracking and how it helps in the design of convex MPC.

3) We conduct various simulations and physical experiments with different WMRs and tracking scenarios, which verified the efficiency of our linearization scheme and control method. Our pure Python-based simulation is open-source to facilitate further research in the community[1].

The remains of this letter are organized as follows. In Section II, some preliminaries on the special Euclidean group and the wheeled mobile robot are presented. In Section III, the main results of the GMPC method are presented. In Section IV, simulations and physical experiments are provided to evaluate the performance of our method. In Section V, the letter is concluded and some future directions are discussed.

Notations: The main notations used in this letter are as follows.

$\mathbb{R}^n$	n-dimensional vector space
SE(2)	special Euclidean group
$\mathfrak{se}(2)$	Lie algebra associated with $SE(2)$
$X \in SE(2)$	state of the rigid body moving in the plane
$X_d \in SE(2)$	desired state of the rigid body moving in the plane
$\zeta \in \mathbb{R}^3$	velocity of the rigid body moving in the plane
$\zeta^\wedge \in \mathfrak{se}(2)$	element in Lie algebra
$u \in \mathbb{R}^2$	control input of the WMR
$\Psi \in SE(2)$	difference between X_d and X
$\psi^\wedge \in \mathfrak{se}(2)$	Lie algebra associated with Ψ
$\dot{X} = \frac{d}{dt}X$	first-order time-derivative of X

[1]https://github.com/Garyandtang/GMPC-Tracking-Control

We use $I_{n\times m}$ and $0_{n\times m}$ to represent the $n \times m$ identity matrix and zero matrix, respectively. For notational clarity, the subscript will be dropped if the matrix dimension is clear.

II. Preliminaries

In this section, some useful mathematical results on the special Euclidean group and the wheeled mobile robot are provided, and the problem to be solved is introduced. More details can be found in [15] for the Lie theory and [7] for the wheeled mobile robot.

A. Special Euclidean Group $SE(2)$

Consider a rigid body in the plane. The position of the rigid body is described by a vector $p = [x, y]^\top \in \mathbb{R}^2$, and the orientation is described by $\theta \in S^1$. The orientation can also be represented by a rotation matrix $R \in SO(2)$, where the special orthogonal group $SO(2)$ is defined as

$$SO(2) = \{R \in \mathbb{R}^{2\times 2} \mid R^\top R = I, \det R = 1\}.$$

Specifically, the rotation matrix $R \in SO(2)$ is represented as

$$R = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}.$$

Hence, the state of the rigid body X can be represented using the homogeneous representation, i.e.,

$$X = \begin{bmatrix} R & p \\ 0_{1\times 2} & 1 \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta & x \\ \sin\theta & \cos\theta & y \\ 0 & 0 & 1 \end{bmatrix} \in \mathbb{R}^{3\times 3}.$$

The combination of the set of all X and the operation of matrix multiplication constitute a Lie group known as special Euclidean group $SE(2)$, i.e.,

$$SE(2) = \left\{ \begin{bmatrix} R & p \\ 0_{1\times 2} & 1 \end{bmatrix} \in \mathbb{R}^{3\times 3} \mid R \in SO(2),\ p \in \mathbb{R}^2 \right\}.$$

The velocity of the rigid body $\zeta = [v, w]^\top \in \mathbb{R}^3$ contains the linear velocity $v = [v_x, v_y]^\top \in \mathbb{R}^2$ and the angular velocity $w \in \mathbb{R}$ in the body frame. Under the framework of Lie theory and geometric control, the velocity lies in the tangent space of the Lie group. The space is also known as the Lie algebra $\mathfrak{se}(2)$ of $SE(2)$. The linear map from the velocity ζ to the element in $\mathfrak{se}(2)$ is denoted as $(\cdot)^\wedge$, i.e.,

$$(\cdot)^\wedge : \mathbb{R}^3 \to \mathfrak{se}(2); \quad \zeta \to \zeta^\wedge = \begin{bmatrix} 0 & -w & v_x \\ w & 0 & v_y \\ 0 & 0 & 0 \end{bmatrix}.$$

The inverse map of $(\cdot)^\wedge$ is denoted as $(\cdot)^\vee$. Given a continuous time-varying velocity $\zeta(t)$, the rigid body motion on a plane is described as follows

$$\dot{X}(t) = X(t)\zeta(t)^\wedge, \tag{1}$$

where $X(t) \in SE(2)$ and $\zeta(t) \in \mathbb{R}^3$. Notation $\dot{X}(t)$ describes the velocity of the rigid body in the global frame at time t.

B. Exponential and Logarithmic Map

The exponential map $\exp(\cdot)$ provides an exact means of mapping elements from the Lie algebra to the corresponding elements in the Lie group. For elements in $SE(2)$, the exponential map $\exp(\cdot)$ and its inverse map, logarithmic map $\log(\cdot)$, can be expressed as follows

$$\exp(\cdot) : \mathfrak{se}(2) \to SE(2),\ \zeta^\wedge \to X = \exp(\zeta^\wedge),$$
$$\log(\cdot) : SE(2) \to \mathfrak{se}(2),\ X \to \zeta^\wedge = \log(X).$$

For convenience, we define the capital exponential map and capital logarithmic map to convert vector elements $\zeta \in \mathbb{R}^3$ directly with elements $X \in SE(2)$ as follows

$$\mathrm{Exp}(\cdot) : \mathbb{R}^3 \to SE(2),\ \zeta \to X = \mathrm{Exp}(\zeta), \tag{2}$$
$$\mathrm{Log}(\cdot) : SE(2) \to \mathbb{R}^3,\ X \to \zeta = \mathrm{Log}(X). \tag{3}$$

C. Wheeled Mobile Robot Kinematics

Consider a nonholonomic wheeled mobile robot that cannot slip in the lateral direction. The corresponding Pfaffian constraint is

$$\dot{x}\sin(\theta) - \dot{y}\cos(\theta) = 0.$$

The kinematic model can be obtained by expressing the entire range of possible velocities, which is shown as follows

$$\begin{bmatrix}\dot{x}\\ \dot{y}\\ \dot{\theta}\end{bmatrix} = \begin{bmatrix}\cos\theta & 0\\ \sin\theta & 0\\ 0 & 1\end{bmatrix}\begin{bmatrix}\mu\\ \omega\end{bmatrix} = C(\theta)u, \tag{4}$$

where $u = [\mu, \omega]^\top \in \mathbb{R}^2$ is the control input, including linear velocity μ and angular velocity ω.

D. Trajectory Tracking for the WMR

Consider the trajectory on special Euclidean group $SE(2)$, we define the motion of the actual state $X(t) \in SE(2)$ and the desired state $X_d(t) \in SE(2)$ both as function of time t, i.e.,

$$\dot{X}(t) = X(t)\zeta(t)^\wedge, \quad \dot{X}_d(t) = X_d(t)\zeta_d(t)^\wedge. \tag{5}$$

Following the group operation to calculate the relative pose from the body frame to reference frame [20]. The error between $X(t)$ and $X_d(t)$ is defined as

$$\Psi(t) = X_d(t)^{-1}X(t) \in SE(2). \tag{6}$$

For the wheeled mobile robot tracking control, we are interested in the design of control input u such that the error (6) can be driven to $I \in SE(2)$ while subject to mobile robot kinematic constraint (4) and control limit constraint. In the following section, we will detail our main ideas for solving the trajectory tracking problem with geometric model predictive control.

III. Main Results

In this section, we introduce our novel control method to solve the trajectory tracking control problem. We first derive the error dynamics of a rigid body subject to the WMR kinematic constraint and formulate the tracking control problem as a continuous-time optimal control. After that, a convex MPC algorithm is developed based on the Lie theory mechanism to realize real-time performance.

A. Error-dynamic Optimal Control

Consider the error between the actual trajectory $X(t)$ and the desired trajectory $X_d(t)$. Taking time-derivative on both sides of (6), we have

$$\begin{aligned}\frac{d}{dt}\Psi(t) = \dot{\Psi}(t) &= \frac{d}{dt}\left(X_d(t)^{-1}\right)X(t) + X_d(t)^{-1}\frac{d}{dt}X(t)\\ &= X_d(t)^{-1}\frac{d}{dt}X(t) - X_d(t)^{-1}\frac{d}{dt}\left(X_d(t)\right)X_d(t)^{-1}X(t)\\ &= X_d(t)^{-1}X(t)\zeta(t)^\wedge - X_d(t)^{-1}X_d(t)\zeta_d(t)^\wedge X_d(t)^{-1}X(t)\\ &= \Psi(t)\zeta(t)^\wedge - \zeta_d(t)^\wedge\Psi(t).\end{aligned}$$

Hence, we obtain the following error dynamics for a rigid body tracking a reference trajectory in the plane.

$$\dot{\Psi}(t) = \Psi(t)\zeta(t)^\wedge - \zeta_d(t)^\wedge\Psi(t). \tag{7}$$

Recall that the WMR follows the kinematics model described by (4). The mapping from the control input $u(t)$ to the local velocity $\zeta(t)$ can be obtained with $C(0)$, i.e.,

$$\zeta(t) = C(0)u(t), \tag{8}$$

where $C(0)$ is a constant matrix. Combining (7) with (8), the overall error dynamics of a WMR in $SE(2)$ representation is shown as follows

$$\dot{\Psi}(t) = \Psi(t)\zeta(t)^\wedge - \zeta_d(t)^\wedge\Psi(t), \tag{9a}$$
$$\zeta(t) = C(0)u(t). \tag{9b}$$

(9) implies that given a reference velocity ζ_d, the map from error state Ψ and control u to the generalized velocity $\dot{\Psi}$ is an injective function, and it considers both the rigid body's manifold constraint and the WMR's kinematic constraint simultaneously.

Based on the above result on error dynamics, we can formulate the trajectory tracking for the WMR as a continuous-time optimal control.

Problem 1: (Error-dynamic Optimal Control)

$$\min_{u(t)}\ J = \phi(\Psi(t_f)) + \int_0^{t_f} l(\Psi(\tau), u(\tau))d\tau \tag{10a}$$
$$\text{s.t.}\quad \dot{\Psi}(t) = \Psi(t)\zeta(t)^\wedge - \zeta_d(t)^\wedge\Psi(t), \tag{10b}$$
$$\zeta(t) = C(0)u(t), \tag{10c}$$
$$\underline{u} \le u(t) \le \overline{u}, \tag{10d}$$
$$\Psi(0) = \Psi_{\text{init}}, \tag{10e}$$
$$0 \le t \le t_f, \tag{10f}$$

where $\phi(\cdot)$ and $l(\cdot,\cdot)$ are the terminal cost and the running time cost, respectively, t_f is trajectory duration, $\underline{u}$ and $\overline{u}$ are the lower bound and the upper bound of the control input, respectively. Ψ_{init} is the initial tracking error.

Note that *Problem 1* is a nonconvex optimization problem since the dynamics constraint (10b) evolves the Lie group constraint from $SE(2)$ and the cost function (10a) design should respect the group structure. In what follows, we detail our proposed method on the system dynamics linearization and the cost function design, which benefits the convex MPC formulation.

B. Convex MPC Formulation

Since the exponential map allows us to map the element in Lie algebra to the Lie group. Define $\psi(t)^\wedge$ as an element of the Lie algebra $\mathfrak{se}(2)$ corresponding to $\Psi(t) \in SE(2)$. By capital exponential map (2), we have

$$\Psi(t) = \mathrm{Exp}(\psi(t)). \tag{11}$$

Taking Taylor expression on (11) at the identity, we have

$$\Psi(t) = \mathrm{Exp}(\psi(t)) = I + \sum_{i=1}^{\infty} \frac{1}{i!}(\psi(t)^\wedge)^i. \tag{12}$$

The first-order approximation of (11) is as follows

$$\Psi(t) = \mathrm{Exp}(\psi(t)) \approx I + \psi(t)^\wedge. \tag{13}$$

Taking time-derivative on both sides of (13), we have

$$\dot{\Psi}(t) \approx \frac{d}{dt}(I + \psi(t)^\wedge) = \dot{\psi}(t)^\wedge. \tag{14}$$

Substitute (13) and (14) into (9a), we have

$$\begin{aligned} \dot{\psi}(t)^\wedge &\approx \Psi(t)\zeta(t)^\wedge - \zeta_d(t)^\wedge \Psi(t) \\ &\approx (I + \psi(t)^\wedge)\zeta(t)^\wedge - \zeta_d(t)^\wedge(I + \psi(t)^\wedge) \\ &= \zeta(t)^\wedge - \zeta_d(t)^\wedge + \psi(t)^\wedge\zeta(t)^\wedge - \zeta_d(t)^\wedge\psi(t)^\wedge. \end{aligned} \tag{15}$$

The nonlinearity in (15) still persists due to the presence of the high-order term $\psi(t)^\wedge\zeta(t)^\wedge$. A naive solution to address this issue is to drop this term and approximate (15) as follows

$$\dot{\psi}(t)^\wedge \approx \zeta(t)^\wedge - \zeta_d(t)^\wedge - \zeta_d(t)^\wedge\psi(t)^\wedge. \tag{16}$$

However, (16) is equivalent to (15) when $\psi(t) = 0$. Nevertheless, $\psi(t) = 0$ is attainable only when there is no tracking error. To provide a better approximation, we approximate (15) as follows

$$\begin{aligned} &\dot{\psi}(t)^\wedge = \zeta(t)^\wedge - \zeta_d(t)^\wedge + \psi(t)^\wedge\zeta(t)^\wedge - \zeta_d(t)^\wedge\psi(t)^\wedge \\ &= \zeta(t)^\wedge - \zeta_d(t)^\wedge + \psi(t)^\wedge\zeta_d(t)^\wedge - \zeta_d(t)^\wedge\psi(t)^\wedge \\ &+ \psi(t)^\wedge(\zeta(t) - \zeta_d(t))^\wedge \\ &\approx \zeta(t)^\wedge - \zeta_d(t)^\wedge + \psi(t)^\wedge\zeta_d(t)^\wedge - \zeta_d(t)^\wedge\psi(t)^\wedge. \end{aligned} \tag{17}$$

Note that (17) is equivalent to (15) when $\zeta(t) = \zeta_d(t)$. Since we aim to minimize the different between $\zeta(t)$ and $\zeta_d(t)$, the the high-order term $\psi(t)^\wedge(\zeta(t) - \zeta_d(t))^\wedge$ we drop in the last approximation of (17) will tend to be small. This indicates that (17) is a better approximation to (15) than (16). We will conduct a numerical comparison of these two linearization schemes in Section IV.

Note that the operation $(\cdot)^\wedge$ is linear. Therefore, (17) is a linearized error dynamics for the WMR. Take $(\cdot)^\vee$ operation on both sides on (17), we have

$$\dot{\psi}(t) = \zeta(t) - \zeta_d(t) + \mathrm{adm}(\zeta_d(t))\psi(t), \tag{18}$$

where $\mathrm{adm}(\zeta_d(t))$ is expressed as follows

$$\mathrm{adm}(\zeta_d(t)) = \begin{bmatrix} 0 & w_d(t) & -v_{y,d}(t) \\ -w_d(t) & 0 & v_{x,d}(t) \\ 0 & 0 & 0 \end{bmatrix}.$$

Combining (18) with (9b), the overall linearized error dynamics of a wheeled mobile robot is

$$\dot{\psi}(t) = A(t)\psi(t) + B(t)u(t) + c(t), \tag{19}$$

where

$$A(t) := \mathrm{adm}(\zeta_d(t)), \quad B(t) := C(0), \quad c(t) := -\zeta_d(t).$$

Different from (9), the state and control input in (19) are in vector space. The above linearization process allows us to utilize the algebraic operations of vector space and further enables us to use existing off-the-shelf solvers to solve optimal control problems.

As we convert the problem to vector space, we can directly use the weighted Euclidean norm to penalize the state and control input. Thus, we define the cost function for tracking control as follows.

$$J = \psi(t_f)^\top Q_f \psi(t_f) + \int_0^{t_f} \psi(\tau)^\top Q \psi(\tau) + \hat{u}(\tau)^\top H \hat{u}(\tau) d\tau,$$

where $Q_f \in \mathbb{R}^{3\times3}$, $Q \in \mathbb{R}^{3\times3}$ and $H \in \mathbb{R}^{2\times2}$ are the final state penalty weight matrix, intermediate state penalty weight matrix, and the control penalty weight matrix, respectively. $\hat{u}(t) = u(t) - u_d(t)$, and $u_d(t)$ is the control input of the reference trajectory satisfying $\zeta_d(t) = C(0)u_d(t)$.

Based on the above results on error dynamics linearization and objective function design, we have the following linear quadratic optimal control problem for trajectory tracking.

Problem 2: (Linear Quadratic Optimal Control)

$$\begin{aligned} \min_{u(t)} \ & \psi(t_f)^\top Q_f \psi(t_f) + \int_0^{t_f} \psi(\tau)^\top Q \psi(\tau) + \hat{u}(\tau)^\top H \hat{u}(\tau) d\tau \\ \text{s.t. } & \dot{\psi}(t) = A(t)\psi(t) + B(t)u(t) + c(t), \\ & \underline{u} \le u(t) \le \overline{u}, \\ & \psi(0) = \psi_{\text{init}}, \\ & 0 \le t \le t_f, \end{aligned}$$

where $\psi_{\text{init}} = \mathrm{Log}(\Psi_{\text{init}})$.

Note that *Problem 2* is a continuous-time optimal control problem. To make it solvable by off-the-shelf optimization solvers, we utilize the direct transcription method [21] to discrete the continuous-time functions to some sequences of $N+1$ real numbers, i.e., for $0 \le t \le t_f$,

$$\begin{aligned} t &\to t_0, \ldots, t_k, \ldots, t_N, \\ \psi(t) &\to \psi_0, \ldots, \psi_k, \ldots, \psi_N, \\ u(t) &\to u_0, \ldots, u_k, \ldots, u_N, \end{aligned}$$

where ψ_k and u_k are the approximations to $\psi(t_k)$, $u(t_k)$ at $t = t_k$, respectively. Therefore, the finite-dimensional convex MPC tracking control is formulated as follows

Problem 3: (Convex MPC Tracking Control)

$$\min_{\{u_k\}_{k=0}^{T-1}} \ \psi_T^\top Q_f \psi_T + \sum_{k=0}^{T-1} \psi_k^\top Q_k \psi_k + \hat{u}_k^\top H_k \hat{u}_k \tag{20a}$$

$$\text{s.t. } \psi_{k+1} = A_k\psi_k + B_k u_k + c_k, \tag{20b}$$

$$\underline{u} \le u_k \le \overline{u}, \tag{20c}$$

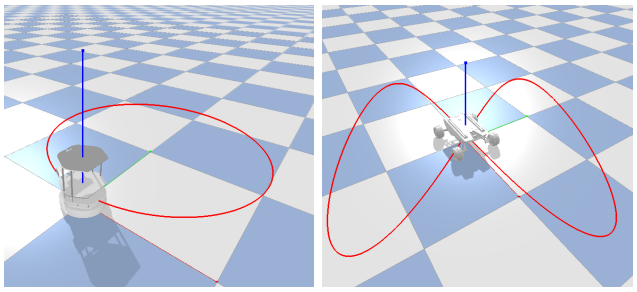
$$\psi_0 = \psi_{\text{init}}, \tag{20d}$$

$$k = 0, 1, \ldots, T-1, \tag{20e}$$

where Q_k, H_k, A_k, B_k, c_k can be obtained through numerical integration and $T \le N$ is the prediction horizon. Since all constraints are linear and the objective is quadratic, *Problem*

(a) Turtlebot 3 (b) Scout mini

Fig. 2: Wheeled mobile robot platforms.



(a) Circular trajectory (b) Butterfly-shaped trajectory

Fig. 3: Trajectory tracking scenarios.

TABLE I: System parameters of two WMR platforms

Platform	Min μ	Max μ	Min ω	Max ω	Ctrl. Freq.
Turtlebot 3	-0.22 m/s	0.22 m/s	-2.84 rad/s	2.84 rad/s	50 Hz
Scout mini	-3 m/s	3 m/s	-2.523 rad/s	2.523 rad/s	50 Hz

3 can be easily solved by off-the-shelf quadratic programming (QP) solvers, such as OSQP [22] and qpOASES [23].

Remark 1: In our implementation, we use the Euler method for numerical integration, i.e.,

$$A_k = I + A(t_k)\Delta t,\ B_k = B(t_k)\Delta t,\ c_k = c(t_k)\Delta t,$$
$$Q_k = Q\Delta t,\ H_k = H\Delta t,\ \Delta t = t_k - t_{k-1}.$$

Different numerical integration methods, such as the Runge-Kutta and Trapezoidal methods [21], are also applicable in the MPC formulation.

IV. Experiments

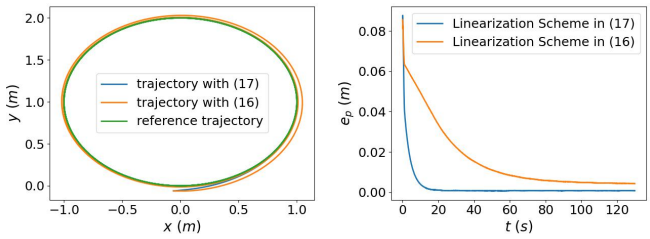
In this section, we conduct simulations and physical experiments to evaluate the performance of the proposed control method on two types of wheeled mobile robots.

A. Simulation Experiments

Our simulation platform is created in PyBullet [24] with the symbolic framework CasADi [25]. The Lie-group-related calculation is implemented based on Manif [26]. We consider two types of wheeled mobile robots, i.e., the two-wheel-drive Turtlebot 3[2] (Fig. 2a) and the four-wheel-drive Scout mini[3] (Fig. 2b). The system parameters of these two WMRs set in the simulation are summarized in TABLE I.

We consider two trajectory tracking scenarios as shown in Fig. 3: (a) a circular trajectory with constant control input u_d, and (b) a butterfly-shaped trajectory with time-varying control input. The trajectories are integrated by u_d from the origin through the forward Euler method. To evaluate the robustness performance, we randomly initialize the start pose of WMRs around the origin.

We compare our method (*GMPC*) with two baseline control methods: (a) a nonlinear model predictive control (*NMPC*) method with orientation represented by Euler angle proposed in [11][4] and (b) a feedback linearization tracking controller (*FBC*) proposed in [7]. We also compare different system dynamics linearization schemes in our *GMPC*, which are described in (16) and (17). Since the performance of tracking controllers is highly dependent on the parameter tuning, we carefully turn the penalty matrices Q, Q_f, and H of *GMPC* and *NMPC* and the feedback gain of *FBC*. In addition, We set the prediction horizon $T = 10$ for the MPC-based controllers. The GMPC is implemented with qpOASES [23], and the NMPC is implemented with IPOPT [27].



(a) Trajectories (b) Position errors

Fig. 4: Circular trajectory tracking with Turtlebot 3 using different linearization schemes. The initial position of the robot is $[-0.06, -0.06]^\top$, and the initial orientation is 0.

We follow [28] to evaluate the tracking error between the actual trajectory and the reference trajectory. Specially, the position error $e_p(t)$ is obtained by Euclidean norm, i.e.,

$$e_p(t) = \|p(t) - p_d(t)\|_2,$$

where $p(t)$ and $p_d(t)$ are the actual and the reference position, respectively. The orientation error is obtained by Riemannian metric, i.e.,

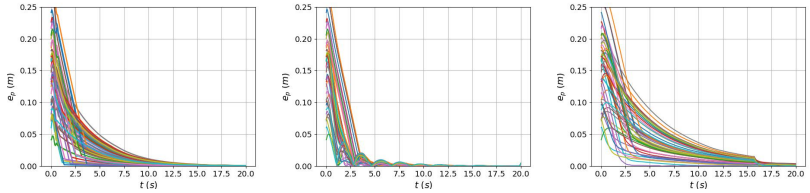
$$e_R(t) = \|\operatorname{Log}(R_d(t)^{-1}R(t))\|_2,$$

where $R(t)$ and $R_d(t)$ are the actual and the reference rotation matrix, respectively.

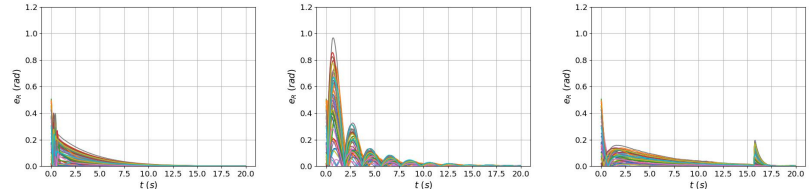
1) Linearization Schemes Comparison: In our GMPC implementation, we conduct a comparison of the linearization schemes discussed in Section III. The results of this comparison are shown in Fig. 4. It is evident from the figure that when employing the linearization scheme outlined in (16), the convergence of position error is relatively slow. Furthermore, this particular linearization scheme leads to non-negligible

[2]https://emanual.robotis.com/docs/en/platform/turtlebot3/overview/
[3]https://global.agilex.ai/more/download/11

[4]As stated in the introduction, many recent works also rely on a similar Euler-based formulation in nonlinear MPC implementation [10]–[14].

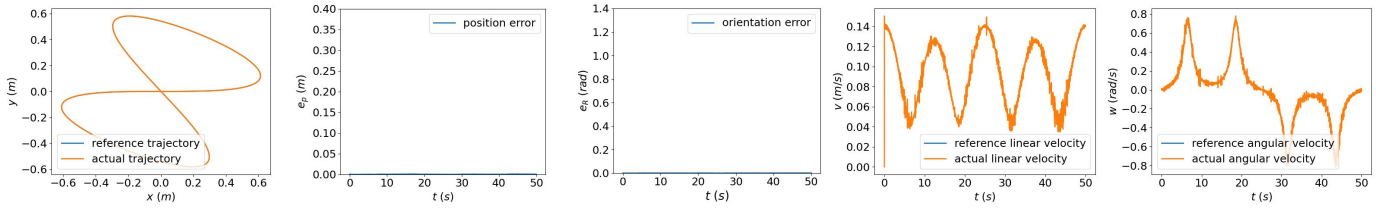


(a) GMPC position error (b) NMPC position error (c) FBC position error

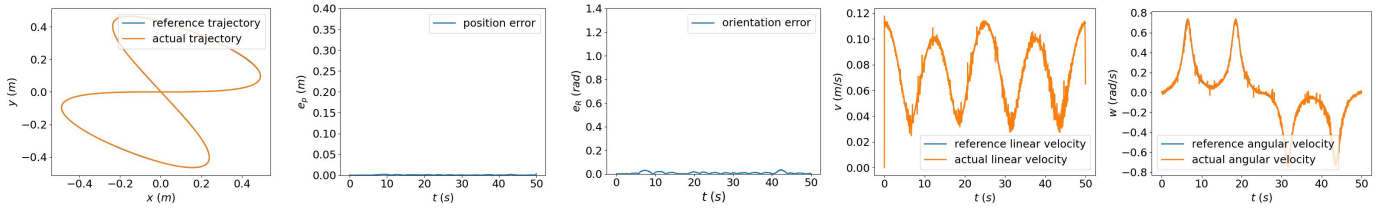


(d) GMPC orientation error (e) NMPC orientation error (f) FBC orientation error

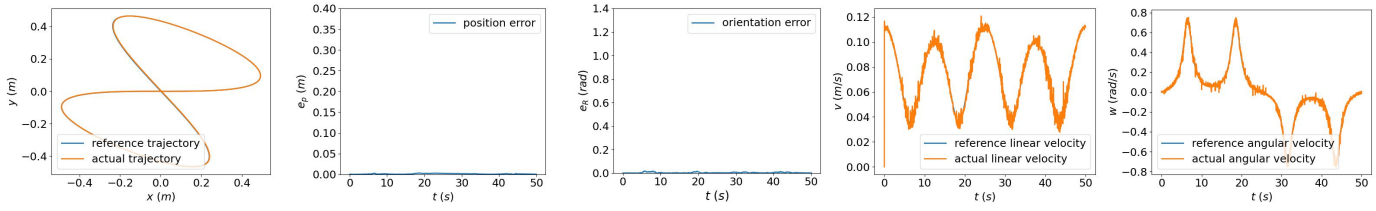
Fig. 5: Monte Carlo tests of tracking a circular trajectory with Turtlebot 3. The initial position of the robot is randomly selected between $[-0.2, 0]^\top$ and $[-0.2, 0]^\top$, and the initial orientation is randomly selected between $[-\frac{\pi}{6}, 0]$.



(a) Trajectory (GMPC) (b) Position error (c) Orientation error (d) Linear velocity (e) Angular velocity



(f) Trajectory (NMPC) (g) Position error (h) Orientation error (i) Linear velocity (j) Angular velocity

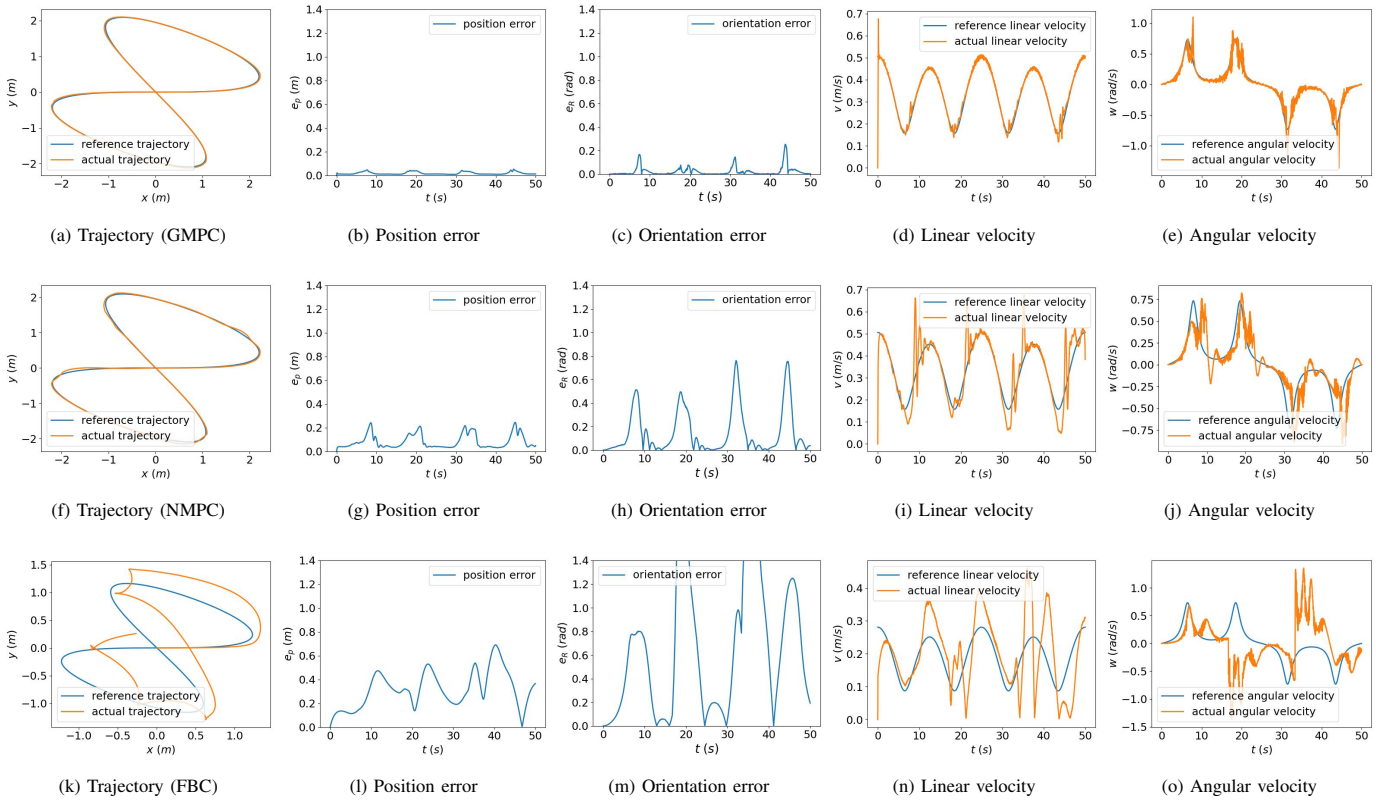


(k) Trajectory (FBC) (l) Position error (m) Orientation error (n) Linear velocity (o) Angular velocity

Fig. 6: Butterfly-shaped trajectory tracking with Turtlebot 3. The initial position and initial orientation are aligned with the start one of the reference trajectory.

steady-state errors when tracking trajectories. In contrast, the linearization scheme described in (17) exhibits notable superior performance. The position error convergence is significantly faster compared to the previous one. Furthermore, (17) effectively mitigates the steady-state error issue that plagued the previous scheme, resulting in a much closer alignment between the actual and desired trajectory.

2) Circular Trajectory Tracking Comparison: Fig. 5 shows the Monte Carlo test of tracking a circular trajectory with Turtlebot 3. In this scenario, three controllers demonstrate convergence in both position and orientation errors. Our controller outperforms other methods in terms of orientation convergence speed. It can be easily verified that GMPC exhibits smooth convergence in terms of position error and orientation error by virtue of the continuity and smoothness properties of the matrix Lie group. Although NMPC can also successfully track circular trajectories, it encounters the singularity problem from Euler-based formulation and numerical instability from nonlinear equality constraints, resulting in overshoot and oscillation. The FBC controller avoids overshooting but suffers from unstable feedback control due to the singularity issue of the Euler angle. Consequently, both $e_p(t)$ and $e_R(t)$ increase



(a) Trajectory (GMPC) (b) Position error (c) Orientation error (d) Linear velocity (e) Angular velocity

(f) Trajectory (NMPC) (g) Position error (h) Orientation error (i) Linear velocity (j) Angular velocity

(k) Trajectory (FBC) (l) Position error (m) Orientation error (n) Linear velocity (o) Angular velocity

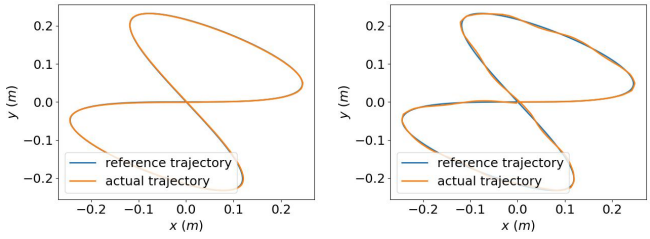
Fig. 7: Butterfly-shaped trajectory tracking with Scout mini. The initial position and initial orientation are aligned with the start one of the reference trajectory.

when the value of θ is around π ($t = 16s$).

3) Butterfly-shaped Trajectory Tracking Comparison: Fig. 6 and Fig. 7 display the results of tracking the butterfly-shaped trajectory using Turtlebot 3 and Scout mini, respectively. When employed on Turtlebot 3, all three controllers can track the reference trajectory. However, when applied to Scout mini, it becomes evident that the FBC fails to track the trajectory successfully. The major reason is that using the same feedback gain, the FBC cannot adapt to the model residuals of the four-wheel-drive model. GMPC can track the reference velocities closely and generate a smooth actual trajectory. However, NMPC struggles to track the reference velocities closely (as shown by the actual linear and angular velocity shown in Fig. 7i and Fig. 7j), which results in large position error and orientation error and unsmooth actual trajectory. (as shown in Fig. 7g, Fig. 7h and Fig. 7a). The improvement of our GMPC showcased the advantage of the matrix Lie group formulation.

B. Physical Experiments

1) Butterfly-shaped Trajectory Test: In this scenario, we demonstrate the physical butterfly-shaped trajectories generated by GMPC and NMPC. We use a Turtlebot3 connected with Ubuntu computer equipped with an AMD Ryzen 9 3900x 12-core CPU as our test platform. The robot's pose is estimated using onboard sensors and transmitted to the computer via Wi-Fi. Fig. 8 illustrates the trajectories from GMPC and NMPC. It is important to note that GMPC exhibits precise tracking of the reference trajectory, whereas NMPC generates a less smooth trajectory. When comparing these physical trajectories

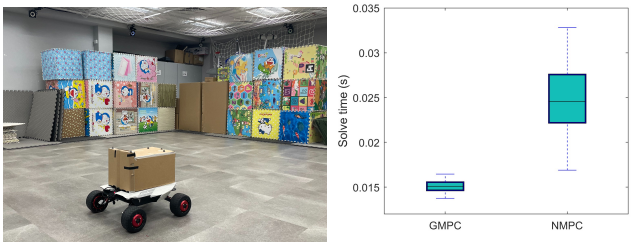


(a) GMPC results (b) NMPC results

Fig. 8: Physical butterfly-shaped trajectories comparison.

to the ones generated in simulations, NMPC proves to be more susceptible to environmental uncertainties, resulting in less smooth paths. However, GMPC maintains a close and smooth tracking of the reference trajectory.

2) Onboard Computer Test: In this scenario, we deploy GMPC and NMPC to a Scout mini equipped with an Intel NUC onboard computer (Intel i7-1165G7 CPU) to test the runtime performance. The experiment takes place indoors, and the OptiTrack Motion Capture system, consisting of 14 cameras, estimates the pose of the WMR. Fig. 9a shows our test arena. The task for the WMR is to track the circular trajectory. In this case, both GMPC and NMPC can track the reference trajectory with a 1cm position error. However, notable differences were observed in their runtime performance. The quarterback chart comparison in Fig. 9b demonstrates that our GMPC outperforms NMPC in terms of solving time, being approximately 1.5 times faster. Furthermore, the GMPC solving time variance is significantly lower than NMPC. The rationale behind the



(a) Test arena (b) Runtime performance

Fig. 9: Onboard computer test.

improved runtime performance of our GMPC controller is its convex formulation for trajectory tracking. As we convert the problem to the vector space and formulate the convex optimization problem, we can adopt QP solvers to solve the problem. This offers computational advantages in efficiency and stability over solving nonlinear programming problems with off-the-shelf NLP solvers.

V. Conclusion

In this letter, we presented a novel geometric model predictive control method for wheeled mobile robot trajectory tracking. We explored the relationship between the Lie group and the corresponding Lie algebra of the WMR to derive its error dynamics for trajectory tracking. Through choosing a suitable linearization scheme, we converted the problem to vector space and formulated the convex optimal control problem, which can be solved efficiently with QP solvers. The simulations and physical experiments demonstrated the superior performance of the proposed method. Future research directions include developing a high-quality trajectory generation algorithm and building a unified planning and control pipeline for WMRs. We hope our geometric control method and open-source code can benefit the development of advanced control in the robotic community.

References

[1] Z. Lin, J. Ma, J. Duan, S. E. Li, H. Ma, B. Cheng, and T. H. Lee, "Policy iteration based approximate dynamic programming toward autonomous driving in constrained dynamic environment," *IEEE Transactions on Intelligent Transportation Systems*, vol. 24, no. 5, pp. 5003–5013, 2023.

[2] Z. He, X. Zhang, S. Jones, S. Hauert, D. Zhang, and N. F. Lepora, "TacMMs: Tactile mobile manipulators for warehouse automation," *IEEE Robotics and Automation Letters*, vol. 8, no. 8, pp. 4729–4736, 2023.

[3] C. Cao, H. Zhu, Z. Ren, H. Choset, and J. Zhang, "Representation granularity enables time-efficient autonomous exploration in large, complex worlds," *Science Robotics*, vol. 8, no. 80, 2023.

[4] C. Canudas de Wit and O. Sordalen, "Exponential stabilization of mobile robots with nonholonomic constraints," in *the 30th IEEE Conference on Decision and Control*, 1991, pp. 692–697 vol.1.

[5] A. Astolfi, "Discontinuous control of nonholonomic systems," *Systems & Control Letters*, vol. 27, no. 1, pp. 37–45, 1996.

[6] D. Kostić, S. Adinandra, J. Caarls, N. van de Wouw, and H. Nijmeijer, "Collision-free tracking control of unicycle mobile robots," in *Proceedings of the 48h IEEE Conference on Decision and Control (CDC) held jointly with 28th Chinese Control Conference*, 2009, pp. 5667–5672.

[7] G. Oriolo, A. De Luca, and M. Vendittelli, "WMR control via dynamic feedback linearization: design, implementation, and experimental validation," *IEEE Transactions on Control Systems Technology*, vol. 10, no. 6, pp. 835–852, 2002.

[8] P. Morin and C. Samson, "Control of nonholonomic mobile robots based on the transverse function approach," *IEEE Transactions on Robotics*, vol. 25, no. 5, pp. 1058–1073, 2009.

[9] J. Pliego-Jiménez, R. Martínez-Clark, C. Cruz-Hernández, and A. Arellano-Delgado, "Trajectory tracking of wheeled mobile robots using only cartesian position measurements," *Automatica*, vol. 133, p. 109756, 2021.

[10] Z. Jian, Z. Lu, X. Zhou, B. Lan, A. Xiao, X. Wang, and B. Liang, "PUTN: A plane-fitting based uneven terrain navigation framework," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2022, pp. 7160–7166.

[11] Z. Jian, Z. Yan, X. Lei, Z. Lu, B. Lan, X. Wang, and B. Liang, "Dynamic control barrier function-based model predictive control to safety-critical obstacle-avoidance of mobile robot," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2023, pp. 3679–3685.

[12] J. Song, G. Tao, Z. Zang, H. Dong, B. Wang, and J. Gong, "Isolating trajectory tracking from motion control: A model predictive control and robust control framework for unmanned ground vehicles," *IEEE Robotics and Automation Letters*, vol. 8, no. 3, pp. 1699–1706, 2023.

[13] S. Khan, J. Guivant, and X. Li, "Design and experimental validation of a robust model predictive control for the optimal trajectory tracking of a small-scale autonomous bulldozer," *Robotics and Autonomous Systems*, vol. 147, p. 103903, 2022.

[14] T. A. Howell, B. E. Jackson, and Z. Manchester, "ALTRO: A fast solver for constrained trajectory optimization," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2019, pp. 7674–7679.

[15] J. Solà, J. Deray, and D. Atchuthan, "A micro lie theory for state estimation in robotics," *ArXiv*, vol. abs/1812.01537, 2018.

[16] B. E. Jackson, K. Tracy, and Z. Manchester, "Planning with attitude," *IEEE Robotics and Automation Letters*, vol. 6, no. 3, pp. 5658–5664, 2021.

[17] G. Alcan, F. J. Abu-Dakka, and V. Kyrki, "Trajectory optimization on matrix lie groups with differential dynamic programming and nonlinear constraints," *ArXiv*, vol. abs/2301.02018, 2023.

[18] G. Lu, W. Xu, and F. Zhang, "On-manifold model predictive control for trajectory tracking on robotic systems," *IEEE Transactions on Industrial Electronics*, vol. 70, no. 9, pp. 9192–9202, 2023.

[19] S. Teng, D. Chen, W. Clark, and M. Ghaffari, "An error-state model predictive control on connected matrix lie groups for legged robot control," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2022, pp. 8850–8857.

[20] A. Barrau and S. Bonnabel, "The invariant extended kalman filter as a stable observer," *IEEE Transactions on Automatic Control*, vol. 62, no. 4, pp. 1797–1812, 2017.

[21] J. T. Betts, *Practical Methods for Optimal Control and Estimation Using Nonlinear Programming*. SIAM, 2010.

[22] B. Stellato, G. Banjac, P. Goulart, A. Bemporad, and S. Boyd, "OSQP: an operator splitting solver for quadratic programs," *Mathematical Programming Computation*, vol. 12, no. 4, pp. 637–672, 2020.

[23] H. Ferreau, C. Kirches, A. Potschka, H. Bock, and M. Diehl, "qpOASES: A parametric active-set algorithm for quadratic programming," *Mathematical Programming Computation*, vol. 6, no. 4, pp. 327–363, 2014.

[24] E. Coumans and Y. Bai, "PyBullet, a Python module for physics simulation for games, robotics and machine learning," http://pybullet.org, 2016–2021.

[25] J. A. Andersson, J. Gillis, G. Horn, J. B. Rawlings, and M. Diehl, "CasADi: a software framework for nonlinear optimization and optimal control," *Mathematical Programming Computation*, vol. 11, pp. 1–36, 2019.

[26] J. Deray and J. Solà, "Manif: A micro Lie theory library for state estimation in robotics applications," *Journal of Open Source Software*, vol. 5, no. 46, p. 1371, 2020.

[27] A. Wächter and L. T. Biegler, "On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming," *Mathematical Programming*, vol. 106, pp. 25–57, 2006.

[28] J. M. Lee, *Introduction to Riemannian Manifolds*. Springer, 2019.