Representation-Free Model Predictive Control for Dynamic Motions in Quadrupeds

Yanran Ding[1], *Student Member, IEEE,* Abhishek Pandala[2], *Student Member, IEEE,* Chuanzheng Li[1], *Student Member, IEEE,* Young-Ha Shin[3], *Student Member, IEEE,* and Hae-Won Park[3], *Member, IEEE*

***Abstract*—This paper presents a novel Representation-Free Model Predictive Control (RF-MPC) framework for controlling various dynamic motions of a quadrupedal robot in three dimensional (3D) space. Our formulation directly represents the rotational dynamics using the rotation matrix, which liberates us from the issues associated with the use of Euler angles and quaternion as the orientation representations. With a variation-based linearization scheme and a carefully constructed cost function, the MPC control law is transcribed to the standard Quadratic Program (QP) form. The MPC controller can operate at real-time rates of 250 Hz on a quadruped robot. Experimental results including periodic quadrupedal gaits and a controlled backflip validate that our control strategy could stabilize dynamic motions that involve singularity in 3D maneuvers.**

***Index Terms*—Model Predictive Control, Legged Robots, Dynamics, Under-actuated Robots**

I. Introduction

THE quadrupedal animals possess extraordinary competence of navigating harsh terrains by executing agile yet well-coordinated movements. For example, mountain goats demonstrate their extraordinary mobility on traversing steep cliffs [2]. Domesticated canine animals could be trained to execute a variety of acrobatic Parkour maneuvers [3]. These remarkable abilities of quadrupedal animals motivated the development of many quadrupedal robots. Minitaur [4] realized various dynamic running gaits; ANYmal [5] and HyQ [6] could navigate challenging terrains autonomously; MIT Cheetah robots achieved galloping [7], high speed bounding [8] and dynamic yet robust locomotion [9]. As the capabilities of quadrupedal robot rapidly grow, related researches have geared towards motions beyond locomotion on flat terrains. For example, ANYmal demonstrated the stair climbing capability [10]; MIT Cheetah 2 overcame obstacles by planning jumping trajectories online [11]; MIT Cheetah 3 achieved leaping onto high platforms [12]; MIT Mini Cheetah could execute 360° backflips [13]. In general, the ability of quadrupedal robots is being developed towards applications that involve more dynamic maneuvers in increasingly complex scenarios.

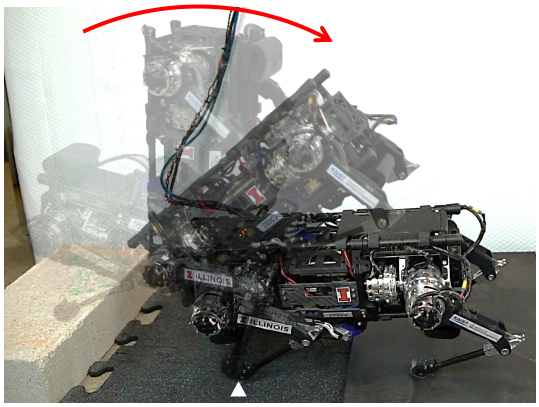

Fig. 1: Quadruped robot Panther performing a controlled backflip that involves passing through the upright pose, which corresponds to the singularity in the Euler angle representation. Throughout the controlled backflip, the feet pair indicated by the white triangle is kept in contact with the ground. The red arrow indicates the direction of the backflip and the shadowed images are snapshots during the backflip.

Controlling a robot to achieve similar mobility that rivals their animal counterparts encompasses many challenges. For instance, the control design should be capable of embracing the inherent dynamics when robots are under-actuated. In addition, the controller must take into consideration the constraints imposed by hardware capacity and the environment. Available solutions for dynamic locomotion include heuristic controller [14], inverse dynamics control [15] and hierarchical operational space control [16]. Simulation result of backflip in Cassie is presented in [17]. Recent hardware results on ANYmal [18] demonstrated the potential of reinforcement

Manuscript received November 11, 2019; re-submitted July 16, 2020; revised November 23, 2020; accepted December 14, 2020. This paper was recommended for publication by Editor Eiichi Yoshida upon evaluation of the reviewers' comments. This work is supported in part by NAVER LABS Corp. under grant 087387, Air Force Office of Scientific Research under grant FA2386-17-1-4665, National Science Foundation under grant 1752262, the Mechanical Engineering Department of Korea Advanced Institute of Science and Technology (KAIST). *(Corresponding author: Hae-Won Park)*, email: haewonpark@kaist.ac.kr

The preliminary version of this paper [1] was presented in ICRA 2019.

Yanran Ding and Chuanzheng Li are with [1]the Department of Mechanical Science and Engineering, University of Illinois at Urbana-Champaign, Urbana,IL, 61820 USA (e-mail: {yding35, cli67}@illinois.edu)

Abhishek Pandala is with [2]the Department of Mechanical Engineering, Virginia Polytechnic Institute and State University, VA, 24061, USA (e-mail: agp19@vt.edu).

Young-Ha Shin and Hae-Won Park are with [3]the Department of Mechanical Engineering, Korea Advanced Institute of Science and Technology, Daejeon-34141, South Korea (e-mail: {shsin000,haewonpark}@kaist.ac.kr)

This paper has supplementary downloadable multimedia material available at http://ieeexplore.ieee.org, provided by the authors. This material includes a video that presents the simulation and experiment results of the proposed MPC controller on a quadruped robot. Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier:

learning in dynamic legged locomotion. Recent times have seen a surge in the use of optimization-based approaches, especially Model Predictive Control (MPC) for legged robots. Successful applications of MPC on humanoid [19] [20] and quadrupeds [21] [22] have shown the efficacy of MPC in planning and controlling a wide variety of dynamic motions.

Most of the MPC control frameworks in the quadrupedal robot community use Euler angles as the orientation representation [23]–[26] since many of locomotion tasks do not involve large deviation from the nominal orientation. However, Euler angles representation has the singularity issue [27] (also known as Gimbal lock), which requires the motions to avoid singular orientations of the Euler angle representation. This disadvantage of Euler angles representation restricts the quadrupedal robot from executing motions such as climbing up near-vertical cliff as the mountain goat, or the acrobatic Parkour as the trained dogs. That is because these motions are highly likely to involve passing through the vicinity of singularity. Quaternion [28] is a singularity-free representation, but as mentioned in [29], quaternions have two local charts that covers the special orthogonal group $SO(3)$ twice. This ambiguity could cause unwinding phenomenon [29], where the body may start arbitrarily close to the desired attitude and yet rotate through large angles before reaching the desired orientation. Widely adopted by the Unmanned Aerial Vehicle (UAV) community, quaternions are often used in reactive controllers which instantaneously respond to the state of the vehicle. Sign function has been used in the reactive controller [30] to eliminate the ambiguity of the quaternion representation. In [31], hybrid dynamic algorithm has been introduced to solve the ambiguity of the quaternion representation. However, for predictive controllers such as MPC, switching of local charts is undesirable. The orientation of a rigid body is originally parameterized by the rotation matrix, which evolves on $SO(3)$ [32]. Although other orientation representations could be re-aligned to avoid their corresponding issues in a specific motion, the rotation matrix possesses advantages as a global parametrization that is compact and singularity-free.

In this manuscript we present the development and experiment results of a representation-free model predictive control (RF-MPC) framework that can be used to stabilize the robot in arbitrary orientation while leveraging the predictive capabilities of MPC.

A. Contribution

In this work, we venture to address the aforementioned problems in the following ways:

1) We introduce a novel MPC formulation for controlling legged robots in dynamic 3D motions using rotation matrices. Specifically, a variation-based linearization technique [33] [34] is used to generate linear dynamics of the single rigid body (SRB) model, which leads to a representation-free MPC (RF-MPC) formulation with consistent performance even when executing motions that involve complex 3D rotations such as acrobatic motions in gymnastics that go through 90° pitch angle.
2) The original orientation error involves the matrix logarithm map, which is a nonlinear function of the optimization variables. We propose an orientation error term in affine form of the state variables as an approximation to the original orientation error. This approximate orientation error term enables the MPC to be transcribed into a Quadratic Program (QP), which in term facilitates real-time MPC control.
3) We implement the proposed RF-MPC on a quadruped robot Panther to realize real-time control of various gaits and a controlled backflip that passes through the singularity.

We have substantially built on the results of our previous work [1] with the following novel strategies. (1) An improved formulation of the angular dynamics of the rotation matrix, which uses 3-dimensional variation vector instead of the 9-dimensional variation matrix. (2) A better choice of the objective function on the orientation which guarantees positive-definiteness. Previously, we used a special configuration error function [32] to approximate the orientation cost function, which only guarantees positive-definiteness when the predicted rotation matrix is on the $SO(3)$ manifold. (3) A section that provides a simulation case study on the effect of linearization schemes. (4) More experimental results, including trot running and the controlled backflip.

B. Outline

The paper is organized as follows: Section II presents the mathematical derivation of the RF-MPC framework. This section first introduces the single rigid body model, which is used for the derivation of the variation-based linearization of the dynamics. A novel vectorization technique along with a deliberately chosen cost function enables the transcription of RF-MPC into a Quadratic Program (QP). Section III presents the numerical results, which includes various dynamic motions and a simulation case study that justifies the choice of the linearization scheme; Section IV summarizes the implementation details necessary for the application on the robot hardware platform; Section V demonstrates the experimental results, followed by a discussion in Section VI. Section VII provides the concluding remark with an outlook for the future work.

II. Model Predictive Control

Model Predictive Control (MPC), also known as Receding Horizon Control (RHC), considers a model of the system to be controlled and repeatedly solves for the optimal control input subject to the state and control constraints. At each sampling time, a finite horizon optimal control problem is solved and the control signal for the first time-step is applied to the system during the following sampling interval. After that, the same process is repeated with the updated measurements. MPC-based controllers have the capability to incorporate various constraints that are essential to legged locomotion, including unilateral ground reaction force (GRF) and friction cone constraints. Besides, MPC could provide control laws that are discontinuous [35], which could not be easily achieved by conventional control techniques.

The MPC control law could be obtained by solv following constrained optimization problem

$$\begin{aligned} \text{minimize} \quad & \ell_T(\boldsymbol{x}_{t+N|t}) + \sum_{k=0}^{N-1} \ell(\boldsymbol{x}_{t+k|t}, \boldsymbol{u}_{t+k|t}) \\ \text{subject to} \quad & \boldsymbol{x}_{t+k+1|t} = \boldsymbol{f}(\boldsymbol{x}_{t+k|t}) + \boldsymbol{g}(\boldsymbol{x}_{t+k|t})\boldsymbol{u} \\ & k = 0, \cdots, N-1 \\ & \boldsymbol{x}_{t+k|t} \in \mathbb{X}, k = 0, \cdots, N-1 \\ & \boldsymbol{u}_{t+k|t} \in \mathbb{U}, k = 0, \cdots, N-1 \\ & \boldsymbol{x}_{t|t} = \boldsymbol{x}(t) = \boldsymbol{x}_{op} \\ & \boldsymbol{x}_{t+N|t} \in \mathbb{X}_f \end{aligned}$$

where $\boldsymbol{x} \in \mathbb{R}^n, \boldsymbol{u} \in \mathbb{R}^m$ are the state and input respectively; $\ell_T : \mathbb{R}^n \to \mathbb{R}$ is the terminal cost f $\ell : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}$ is the stage cost function; N is the pr horizon; $\boldsymbol{f}(\boldsymbol{x}) + \boldsymbol{g}(\boldsymbol{x})\boldsymbol{u}$ is the control affine dynamic update equation; $\mathbb{X} \subseteq \mathbb{R}^n, \mathbb{U} \subseteq \mathbb{R}^m$ are the feasible polyhedral sets for the state and control; $\mathbb{X}_f$ is the final state set; $\boldsymbol{x}_{t+k|t}$ denotes the state vector at time $t+k$ predicted at time t, using the current state measurement $\boldsymbol{x}_{t|t} = \boldsymbol{x}_{op}$, where the subscript $(\cdot)_{op}$ denotes the variables at the current operating point. The operating point in this manuscript is defined as the current state $\boldsymbol{x}_{op}$ and control $\boldsymbol{u}_{op}$.

In the case that the dynamic update equation $\boldsymbol{f}(\boldsymbol{x}) + \boldsymbol{g}(\boldsymbol{x})\boldsymbol{u}$ is a nonlinear function, a nonlinear MPC (NMPC) could be formulated and solved as a general nonlinear program (NLP) by utilizing trajectory optimization (TO) techniques such as multiple shooting [36] or direct collocation [37].

Our main objective is to formulate a real-time executable MPC scheme for controlling quadruped robotics to perform a variety of dynamic motions. To meet the real-time requirement, the optimization problem posed by the MPC has to be solved robustly at a high rate on the mobile embedded computer, which has limited computational resources. Hence, a simplified model is adopted to reduce the dimensionality of the optimization problem. Since the mass of all legs combined is less than 10% of the total body mass, a single rigid body model serves as a reasonable approximation.

A. 3D Single Rigid Body Model

To mitigate the issue of demanding computational requirement of MPC for high Degrees of Freedom (DoF) system models, simple models or templates [38] that capture the dominant system dynamics are used instead. Templates such as the Linear Inverted Pendulum [39] (LIP) is widely used in humanoid robots [40] [41]. Centroidal dynamics [42] model is used in [43] [44] [45] to capture the major dynamic effect of the complex full-body dynamics model. The quadrupedal robot community has seen an increasing number of work that utilizes the SRB model in three-dimensional (3D) space, which assumes that the entire mass of the robot is lumped into a single rigid body (SRB). The simplicity of the SRB model is enabled by the light leg design, whose inertial effect is negligible compared with the body. Let the state of the single rigid body model be

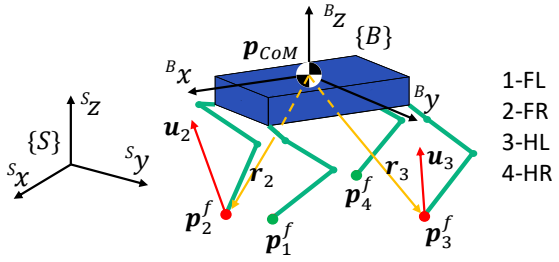

Fig. 2: Illustration of coordinate systems and the 3D single rigid-body model. $\{\boldsymbol{S}\}$ is the inertia frame and $\{\boldsymbol{B}\}$ is the body attached frame. $\boldsymbol{r}_i$ is the position vector from center of mass to each foot in $\{\boldsymbol{S}\}$ and $\boldsymbol{u}_i$ is the GRF of i^{th} contact foot expressed in $\{\boldsymbol{S}\}$. The convention for the numbering of feet is such that FL stands for front-left leg, and HR stands for hind-right leg.

$$\boldsymbol{x} := [\boldsymbol{p} \;\; \dot{\boldsymbol{p}} \;\; \boldsymbol{R} \;\; {}^B\boldsymbol{\omega}] \in \mathbb{R}^{18}, \tag{2}$$

where $\boldsymbol{p} \in \mathbb{R}^3$ is the position of the body Center of Mass (CoM); $\dot{\boldsymbol{p}} \in \mathbb{R}^3$ is the CoM velocity; $\boldsymbol{R} \in SO(3) = \{\boldsymbol{R} \in \mathbb{R}^{3\times3} | \boldsymbol{R}^T\boldsymbol{R} = \mathbb{I}, \det(\boldsymbol{R}) = +1\}$ is the rotation matrix of the body frame $\{\boldsymbol{B}\}$ expressed in the inertial frame $\{\boldsymbol{S}\}$; $\det(\cdot)$ calculates the determinant of a matrix and $\mathbb{I}$ is the 3-by-3 identity matrix. Here, the rotation matrix $\boldsymbol{R}$ is reshaped into vector form. ${}^B\boldsymbol{\omega} \in \mathbb{R}^3$ indicates the angular velocity vector expressed in the body frame $\{\boldsymbol{B}\}$. Variables without superscript on the upper-left corner could be assumed to be expressed in the inertial frame. The illustration of the coordinate system could be found in Fig. 2.

The input to the dynamical system is the GRF $\boldsymbol{u}_i \in \mathbb{R}^3$ at contact foot locations $\boldsymbol{p}_i^f \in \mathbb{R}^3$. The GRFs create the external wrench to the rigid body, where $i \in \{1, 2, 3, 4\}$ is the index for the front left (FL), front right (FR), hind left (HL) and hind right (HR), respectively, as shown in Fig. 2. The foot positions $\boldsymbol{p}_i^f$ relative to CoM are denoted as $\boldsymbol{r}_i = \boldsymbol{p}_i^f - \boldsymbol{p}$. Therefore, the net external wrench $\boldsymbol{\mathcal{F}} \in \mathbb{R}^6$ exerted on the body is:

$$\boldsymbol{\mathcal{F}} = \begin{bmatrix} \boldsymbol{F} \\ \boldsymbol{\tau} \end{bmatrix} = \sum_{i=1}^{4} \begin{bmatrix} \mathbb{I} \\ \hat{\boldsymbol{r}}_i \end{bmatrix} \boldsymbol{u}_i, \tag{3}$$

where $\boldsymbol{F}$ and $\boldsymbol{\tau}$ are the total force and torque applied at the CoM; the hat map $\hat{(\cdot)} : \mathbb{R}^3 \to \mathfrak{so}(3)$ maps an element from $\mathbb{R}^3$ to the space of skew-symmetric matrices $\mathfrak{so}(3)$, which represents the cross-product under multiplication as $\hat{\boldsymbol{\alpha}}\boldsymbol{\beta} = \boldsymbol{\alpha} \times \boldsymbol{\beta}$, for all $\boldsymbol{\alpha}, \boldsymbol{\beta} \in \mathbb{R}^3$. The inverse of the hat map is the vee map $(\cdot)^\vee : \mathfrak{so}(3) \to \mathbb{R}^3$. The full dynamics of the rigid body can be written as

$$\dot{\boldsymbol{x}} = \begin{bmatrix} \dot{\boldsymbol{p}} \\ \ddot{\boldsymbol{p}} \\ \dot{\boldsymbol{R}} \\ {}^B\dot{\boldsymbol{\omega}} \end{bmatrix} = \begin{bmatrix} \dot{\boldsymbol{p}} \\ \frac{1}{M}\boldsymbol{F} + \boldsymbol{a}_g \\ \boldsymbol{R} \cdot {}^B\hat{\boldsymbol{\omega}} \\ {}^B\boldsymbol{I}^{-1}(\boldsymbol{R}^T\boldsymbol{\tau} - {}^B\hat{\boldsymbol{\omega}}{}^B\boldsymbol{I}{}^B\boldsymbol{\omega}) \end{bmatrix}, \tag{4}$$

where $\boldsymbol{u} = [\boldsymbol{u}_1^T, \boldsymbol{u}_2^T, \boldsymbol{u}_3^T, \boldsymbol{u}_4^T]^T \in \mathbb{R}^{12}$ is the control vector; M is the mass of the rigid body; $\boldsymbol{a}_g = [0, 0, -g]^T$ is the gravitational acceleration vector; ${}^B\boldsymbol{I} \in \mathbb{R}^{3\times3}$ is the fixed moment of inertia tensor in the body frame $\{\boldsymbol{B}\}$. The inertia properties of the robot could be found in Table I.

TABLE I: System Parameters of the Robot

Parameter	Value	Unit
M	5.5	kg
I_{xx}	0.026	kg·m^2
I_{yy}	0.112	kg·m^2
I_{zz}	0.075	kg·m^2
Body length	0.3	m
Body width	0.2	m
Link length	0.14	m

To develop a representation-free control approach, we decided to directly parameterize orientation using the rotation matrix. This completely avoids the singularities and complexities when using local coordinates such as Euler angles. It also avoids the ambiguities when using quaternions to represent attitude dynamics. As the quaternion double covers the special orthogonal group $SO(3)$, the control design needs to switch between the local charts.

The rotational dynamics of (4) is nonlinear for it involves the rotation matrix $\boldsymbol{R}$, which evolves on the $SO(3)$ manifold. In Section II-B we present a variation-based linearization scheme for linearizing the rotational dynamics.

B. Variation-based Linearization

Although the nonlinear MPC (1) could be solved to obtain the control input, the presence of local optimum resulting from the nonlinear dynamics complicates the solution process. Furthermore, convoluted nonlinear optimization does not lend itself well to embedded implementations. To meet the real-time constraint, we strive to formulate the MPC as a Quadratic Program (QP) that can be efficiently solved on embedded systems. A variation-based linearization scheme for the rotation matrix is proposed to linearize the nonlinear rotational dynamics. The error on the non-Euclidean $SO(3)$ manifold is approximated by the corresponding variation [46] with respect to the operating point. Then the variation dynamics is derived based on the system model (4) in the manner of [33]. Recent work [47] achieved underactuated two-leg balancing on MIT Mini Cheetah using variational-based linearization on the $SO(3)$ manifold.

Assuming that the predicted variables are close to the operating point, the variation of the rotation matrix on $\mathfrak{so}(3)$ could be approximated by $\delta\boldsymbol{R}$ using the derivative of the error function on $SO(3)$ as in [48]. The variation $\delta\boldsymbol{R} \in \mathfrak{so}(3)$ is a local approximation of the displacement between two points on the $SO(3)$ manifold. The rotation matrix at the k^{th} prediction step is approximated using the first-order Taylor expansion of matrix exponential map,

$$\boldsymbol{R}_k \approx \boldsymbol{R}_{op}\exp(\delta\boldsymbol{R}_k) \approx \boldsymbol{R}_{op}(\mathbb{I} + \delta\boldsymbol{R}_k), \tag{5}$$

where we use the commutativity of small rotations based on the assumption of $\delta\boldsymbol{R}$ being small.

The nonlinear dynamics of the rotation matrix is given as

$$\dot{\boldsymbol{R}} = \boldsymbol{R}^B\hat{\boldsymbol{\omega}}, \tag{6}$$

where the first-order approximation of rotation matrix $\boldsymbol{R}_k$ is presented in (5). To get a linear approximation for $\dot{\boldsymbol{R}}$, we define the variation of angular velocity $\delta\boldsymbol{\omega}_k$ as

$$\delta\boldsymbol{\omega}_k = \boldsymbol{\omega}_k - \boldsymbol{R}_k^T\boldsymbol{R}_{op}\boldsymbol{\omega}_{op}, \tag{7}$$

where the transport map $\boldsymbol{\omega}_{op} \to \boldsymbol{R}_k^T\boldsymbol{R}_{op}\boldsymbol{\omega}_{op}$ enables comparison between tangent vectors at different points. This procedure is required because the tangent vectors $\dot{\boldsymbol{R}}_k \in T_{R_k}SO(3)$ and $\dot{\boldsymbol{R}}_{op} \in T_{R_{op}}SO(3)$ lie in different tangent spaces and cannot be compared directly, where $T_{R_{op}}SO(3)$ refers to the tangent space of $SO(3)$ at $\boldsymbol{R}_{op}$. Hence, the angular velocity $\boldsymbol{\omega}$ could be deducted from (7) as

$$\begin{aligned}\boldsymbol{\omega}_k =& \boldsymbol{R}_k^T\boldsymbol{R}_{op}\boldsymbol{\omega}_{op} + \delta\boldsymbol{\omega}_k\\ =&(\mathbb{I} + \delta\boldsymbol{R}_k)^T\boldsymbol{\omega}_{op} + \delta\boldsymbol{\omega}_k\\ =&\boldsymbol{\omega}_{op} + \delta\boldsymbol{\omega}_k - \delta\boldsymbol{R}_k\boldsymbol{\omega}_{op},\end{aligned} \tag{8}$$

where $\boldsymbol{R}_k$ is replaced by the expression in (5). The last step in (8) is due to the fact that $\delta\boldsymbol{R}_k$ is a skew-symmetric matrix by construction. Applying the hat map $\hat{(\cdot)}$ to $\boldsymbol{\omega}_k$ and substituting (8) into (6) yields

$$\begin{aligned}\dot{\boldsymbol{R}}_k &= \boldsymbol{R}_k\hat{\boldsymbol{\omega}}_k = \boldsymbol{R}_{op}(\mathbb{I} + \delta\boldsymbol{R}_k)(\hat{\boldsymbol{\omega}}_{op} + \widehat{\delta\boldsymbol{\omega}_k} - \widehat{\delta\boldsymbol{R}_k\boldsymbol{\omega}_{op}})\\ &= \boldsymbol{R}_{op}\hat{\boldsymbol{\omega}}_{op} + \boldsymbol{R}_{op}\widehat{\delta\boldsymbol{\omega}_k} - \boldsymbol{R}_{op}\widehat{\delta\boldsymbol{R}_k\boldsymbol{\omega}_{op}} + \boldsymbol{R}_{op}\delta\boldsymbol{R}_k\hat{\boldsymbol{\omega}}_{op},\end{aligned} \tag{9}$$

where the higher order variation terms are neglected. Using the properties of cross product in [49] Table 2.1, the following equality

$$\widehat{\delta\boldsymbol{R}_k\boldsymbol{\omega}_{op}} = \delta\boldsymbol{R}_k\hat{\boldsymbol{\omega}}_{op} - \hat{\boldsymbol{\omega}}_{op}\delta\boldsymbol{R}_k, \tag{10}$$

is used to derive the expression of $\dot{\boldsymbol{R}}_k$

$$\dot{\boldsymbol{R}}_k = \boldsymbol{R}_{op}\hat{\boldsymbol{\omega}}_{op} + \boldsymbol{R}_{op}\hat{\boldsymbol{\omega}}_{op}\delta\boldsymbol{R}_k + \boldsymbol{R}_{op}\widehat{\delta\boldsymbol{\omega}_k}. \tag{11}$$

The dynamics of angular velocity $\dot{\boldsymbol{\omega}}_k$ is linearized as

$$\begin{aligned}{}^B\boldsymbol{I}\dot{\boldsymbol{\omega}}_k =&\boldsymbol{R}_{op}^T\boldsymbol{\tau}_{op} + \delta\boldsymbol{R}_k^T\boldsymbol{\tau}_{op} + \boldsymbol{R}_{op}^T\delta\boldsymbol{\tau}_k +\\ &- \hat{\boldsymbol{\omega}}_{op}{}^B\boldsymbol{I}\boldsymbol{\omega}_{op} - \delta\hat{\boldsymbol{\omega}}_k{}^B\boldsymbol{I}\boldsymbol{\omega}_{op} - \hat{\boldsymbol{\omega}}_{op}{}^B\boldsymbol{I}\delta\boldsymbol{\omega}_k,\end{aligned} \tag{12}$$

in which $\delta\boldsymbol{\tau}_k$ is the variation of the net torque,

$$\delta\boldsymbol{\tau}_k = (\sum_{i=1}^{4}\hat{\boldsymbol{u}}_{i,op})\delta\boldsymbol{p}_k + (\sum_{i=1}^{4}\hat{\boldsymbol{r}}_{i,op}\cdot\delta\boldsymbol{u}_{i,k}), \tag{13}$$

where $\boldsymbol{u}_{op}$ is GRF applied at the current step, $\delta\boldsymbol{u}$ is the variation of GRF from $\boldsymbol{u}_{op}$. Note that the GRF applied at the next time step is $\boldsymbol{u}_{op} + \delta\boldsymbol{u}_1$.

The linearized dynamics will be used as affine dynamics as well as in the construction of objective function in the MPC formulation.

C. Vectorization

After the dynamics of rotation matrix $\boldsymbol{R}$ and angular velocity $\boldsymbol{\omega}$ are linearized, the matrix variables in (11) and (12) are still difficult to be formulated into the standard QP form. This section proposes a vectorization technique which uses Kronecker product [50] to transform matrix-matrix products into matrix-vector products.

Let us define a vector $\boldsymbol{\xi} \in \mathbb{R}^3$ be such that the skew-symmetric matrix $\hat{\boldsymbol{\xi}} = \delta\boldsymbol{R} \in \mathfrak{so}(3)$ is an element in the tangent

space at the operating point. Let $\boldsymbol{N} \in \mathbb{R}^{9\times 3}$ be a constant matrix such that $vec(\hat{v}) = \boldsymbol{N}v, \forall v \in \mathbb{R}^3$, where $vec(\cdot)$ is the vectorization function. Then $vec(\delta \boldsymbol{R}) = \boldsymbol{N} \cdot \boldsymbol{\xi}$. The second and third terms of (11) could be vectorized as:

$$\begin{aligned} vec(\boldsymbol{R}_{op}\hat{\boldsymbol{\omega}}_{op}\delta\boldsymbol{R}_k) &= (\mathbb{I}\otimes \boldsymbol{R}_{op}\hat{\boldsymbol{\omega}}_{op})\boldsymbol{N}\boldsymbol{\xi}_k \\ vec(\boldsymbol{R}_{op}\widehat{\delta\boldsymbol{\omega}_k}) &= (\mathbb{I}\otimes \boldsymbol{R}_{op})vec(\widehat{\delta\boldsymbol{\omega}_k}), \end{aligned} \tag{14}$$

where $\otimes$ is the Kronecker tensor operator. To derive the expression for $vec(\widehat{\delta\boldsymbol{\omega}_k})$, one plugs (5) into (7)

$$vec(\widehat{\delta\boldsymbol{\omega}_k}) = \boldsymbol{N}(\boldsymbol{\omega}_k - \boldsymbol{\omega}_{op} + \hat{\boldsymbol{\omega}}_{op}\boldsymbol{\xi}_k). \tag{15}$$

The vectorized version of (11) is

$$vec(\dot{\boldsymbol{R}}_k) = \boldsymbol{C}^c_\xi + \boldsymbol{C}^\xi_\xi\boldsymbol{\xi}_k + \boldsymbol{C}^\omega_\xi\boldsymbol{\omega}_k, \tag{16}$$

where the constants are defined as

$$\begin{aligned} \boldsymbol{C}^c_\xi &= vec(\boldsymbol{R}_{op}\hat{\boldsymbol{\omega}}_{op}) - (\mathbb{I}\otimes \boldsymbol{R}_{op})\boldsymbol{N}\boldsymbol{\omega}_{op} \\ \boldsymbol{C}^\xi_\xi &= (\mathbb{I}\otimes \boldsymbol{R}_{op}\hat{\boldsymbol{\omega}}_{op})\boldsymbol{N} - (\mathbb{I}\otimes \boldsymbol{R}_{op})\boldsymbol{N}\hat{\boldsymbol{\omega}}_{op} \\ \boldsymbol{C}^\omega_\xi &= (\mathbb{I}\otimes \boldsymbol{R}_{op})\boldsymbol{N}. \end{aligned} \tag{17}$$

The discrete orientation dynamics in terms of $\boldsymbol{\xi}$ is derived by propagating the rotation matrix using the forward Euler integration scheme,

$$\boldsymbol{R}_{k+1} = \boldsymbol{R}_k + \dot{\boldsymbol{R}}_k dt, \tag{18}$$

where dt is the MPC sampling time. When the rotation matrix is approximated by the first-order expansion in (5), the above expression could be simplified to the following form,

$$\delta\boldsymbol{R}_{k+1} = \delta\boldsymbol{R}_k + \boldsymbol{R}^T_{op}\dot{\boldsymbol{R}}_k dt. \tag{19}$$

Vectorizing (19) gives

$$\boldsymbol{N}\boldsymbol{\xi}_{k+1} = \boldsymbol{N}\boldsymbol{\xi}_k + dt(\mathbb{I}\otimes \boldsymbol{R}^T_{op})vec(\dot{\boldsymbol{R}}_k). \tag{20}$$

The discrete dynamics in $\boldsymbol{\xi}$ is obtained by putting in (16) and pre-multiply with $\boldsymbol{N}^*$, the left pseudo-inverse of $\boldsymbol{N}$

$$\boldsymbol{\xi}_{k+1} = \boldsymbol{\xi}_k + dt\boldsymbol{N}^*(\mathbb{I}\otimes \boldsymbol{R}^T_{op})(\boldsymbol{C}^c_\xi + \boldsymbol{C}^\xi_\xi\boldsymbol{\xi}_k + \boldsymbol{C}^\omega_\xi\boldsymbol{\omega}_k). \tag{21}$$

The angular velocity dynamics in (12) is also vectorized. The second term in (12) is $\delta\boldsymbol{R}^T_k\boldsymbol{\tau}_{op} = (\mathbb{I}\otimes\boldsymbol{\tau}^T_{op})\boldsymbol{N}\boldsymbol{\xi}_k$ and $\delta\boldsymbol{\tau}_k$ is defined in (13). The last two terms could be further derived,

$$\begin{aligned} &-\delta\hat{\boldsymbol{\omega}}_k{}^B\boldsymbol{I}\boldsymbol{\omega}_{op} - \hat{\boldsymbol{\omega}}_{op}{}^B\boldsymbol{I}\delta\boldsymbol{\omega}_k \\ &= (\widehat{{}^B\boldsymbol{I}\boldsymbol{\omega}_{op}} - \hat{\boldsymbol{\omega}}_{op}{}^B\boldsymbol{I})\delta\boldsymbol{\omega} \\ &= (\widehat{{}^B\boldsymbol{I}\boldsymbol{\omega}_{op}} - \hat{\boldsymbol{\omega}}_{op}{}^B\boldsymbol{I})(\boldsymbol{\omega}_k - \boldsymbol{\omega}_{op} - \hat{\boldsymbol{\omega}}_{op}\boldsymbol{\xi}_k). \end{aligned} \tag{22}$$

Assembling these expressions into (12) and rearranging the corresponding terms gives the vectorization of ${}^B\boldsymbol{I}\dot{\boldsymbol{\omega}}_k$

$${}^B\boldsymbol{I}\dot{\boldsymbol{\omega}}_k = \boldsymbol{C}_{\dot{\omega}} + \boldsymbol{C}^{\delta p}_{\dot{\omega}}\boldsymbol{p}_k + \boldsymbol{C}^\xi_{\dot{\omega}}\boldsymbol{\xi}_k + \boldsymbol{C}^\omega_{\dot{\omega}}\boldsymbol{\omega}_k + \boldsymbol{C}^{\delta u}_{\dot{\omega}}\delta\boldsymbol{u}_k, \tag{23}$$

where

$$\begin{aligned} \boldsymbol{C}^c_{\dot{\omega}} &= -\hat{\boldsymbol{\omega}}_{op}{}^B\boldsymbol{I}\boldsymbol{\omega}_{op} + \boldsymbol{R}^T_{op}\boldsymbol{\tau}_{op} - (\widehat{{}^B\boldsymbol{I}\boldsymbol{\omega}_{op}} - \hat{\boldsymbol{\omega}}_{op}{}^B\boldsymbol{I})\boldsymbol{\omega}_{op} \\ &\quad - \boldsymbol{R}^T_{op}(\sum\hat{\boldsymbol{u}}_{op})\boldsymbol{p}_{op} \\ \boldsymbol{C}^{\delta p}_{\dot{\omega}} &= \boldsymbol{R}^T_{op}(\sum_i\hat{\boldsymbol{u}}^i_{op}) \\ \boldsymbol{C}^\xi_{\dot{\omega}} &= (\mathbb{I}\otimes\boldsymbol{\tau}^T_{op})\boldsymbol{N} - (\widehat{{}^B\boldsymbol{I}\boldsymbol{\omega}_{op}} - \hat{\boldsymbol{\omega}}_{op}{}^B\boldsymbol{I})\hat{\boldsymbol{\omega}}_{op} \\ \boldsymbol{C}^\omega_{\dot{\omega}} &= \widehat{{}^B\boldsymbol{I}\boldsymbol{\omega}_{op}} - \hat{\boldsymbol{\omega}}_{op}{}^B\boldsymbol{I} \\ \boldsymbol{C}^{\delta u}_{\dot{\omega}} &= \boldsymbol{R}^T_{op}[\hat{\boldsymbol{r}}^1_{op}, \hat{\boldsymbol{r}}^2_{op}, \hat{\boldsymbol{r}}^3_{op}, \hat{\boldsymbol{r}}^4_{op}]. \end{aligned} \tag{24}$$

The discrete dynamics of $\boldsymbol{\omega}$ is propagated using forward Euler scheme $\boldsymbol{\omega}_{k+1} = \boldsymbol{\omega}_k + dt\dot{\boldsymbol{\omega}}_k$.

D. Discrete-time Affine Dynamics

The single rigid body model introduced in Section II-A has nonlinear dynamics in $\boldsymbol{R}$ and $\boldsymbol{\omega}$. Hence, a variation-based linearization scheme is proposed in Section II-B to linearize the nonlinear dynamics. Section II-C presents a vectorization method to reformulate the matrix variables into the vector form. Based on the aforementioned procedures, the new set of state and control vectors are defined as,

$$\begin{aligned} \boldsymbol{x}_k &:= [\boldsymbol{p}^T_k \;\; \dot{\boldsymbol{p}}^T_k \;\; \boldsymbol{\xi}^T_k \;\; {}^B\boldsymbol{\omega}^T_k]^T \in \mathbb{R}^{12} \\ \delta\boldsymbol{u}_k &:= [\delta\boldsymbol{u}^T_{1,k} \;\; \delta\boldsymbol{u}^T_{2,k} \;\; \delta\boldsymbol{u}^T_{3,k} \;\; \delta\boldsymbol{u}^T_{4,k}]^T \in \mathbb{R}^{12}, \end{aligned} \tag{25}$$

where the new control input $\delta\boldsymbol{u}_i \in \mathbb{R}^3$ is the variation of GRF from the operating point $\boldsymbol{u}_{i,op}$ for the i^{th} leg.

By assembling the corresponding terms from (21), (23) into matrix form, the discrete-time affine dynamics could be expressed in the state-space form:

$$\boldsymbol{x}_{t+k+1|t} = \boldsymbol{A}|_{op}\cdot\boldsymbol{x}_{t+k|t} + \boldsymbol{B}|_{op}\cdot\delta\boldsymbol{u}_{t+k|t} + \boldsymbol{d}|_{op}, \tag{26}$$

where $\boldsymbol{A}|_{op} \in \mathbb{R}^{n\times n}, \boldsymbol{B}|_{op} \in \mathbb{R}^{n\times m}$, and $\boldsymbol{d}|_{op} \in \mathbb{R}^n$ are matrices constructed by the measurements at the operating point. Therefore, nonlinear dynamics have been linearized about the operating point to result in a locally-valid linear time-varying (LTV) system. This system can be stabilized to track reference trajectories.

The discrete-time affine dynamics are imposed as equality constraint as in (1b).

E. Cost Function

The cost function in the nonlinear MPC formulation (1) includes both terminal and stage costs. In this work, the cost is set as a quadratic function that penalizes deviation from the reference trajectories. The stage cost is

$$\ell(\boldsymbol{x}_k, \boldsymbol{u}_k) = ||\boldsymbol{x}_k - \boldsymbol{x}_{d,k}||^2_{\boldsymbol{Q}_x} + ||\boldsymbol{u}_k - \boldsymbol{u}_{d,k}||^2_{\boldsymbol{R}_u}, \tag{27}$$

where $||\boldsymbol{x}||^2_{\boldsymbol{Q}}$ is a shorthand notation of the matrix norm $\boldsymbol{x}^T\boldsymbol{Q}\boldsymbol{x}$ where $\boldsymbol{Q}$ is a positive definite matrix; $\boldsymbol{x}_{d,k}$ and $\boldsymbol{u}_{d,k}$ are the desired state and control at the k^{th} prediction step; $\boldsymbol{Q}_x$ and $\boldsymbol{R}_u$ are the block diagonal positive definite gain matrices for state and control, respectively. The first term in (27) consisting of the error functions of the state vector could be decomposed as:

$$||\boldsymbol{x}_k - \boldsymbol{x}_{k,d}||^2_{\boldsymbol{Q}} = ||\boldsymbol{e}_{p_k}||^2_{\boldsymbol{Q}_p} + ||\boldsymbol{e}_{\dot{p}_k}||^2_{\boldsymbol{Q}_{\dot{p}}} + ||\boldsymbol{e}_{R_k}||^2_{\boldsymbol{Q}_R} + ||\boldsymbol{e}_{\omega_k}||^2_{\boldsymbol{Q}_\omega}, \tag{28}$$

where $\boldsymbol{Q}_p, \boldsymbol{Q}_{\dot{p}}, \boldsymbol{Q}_R, \boldsymbol{Q}_\omega$ are diagonal positive definite weighting matrices; $\boldsymbol{e}_{p_k}, \boldsymbol{e}_{\dot{p}_k}$ are the error terms for deviations from the corresponding reference trajectories $\boldsymbol{p}^d_k, \dot{\boldsymbol{p}}^d_k$ constructed from the user input. The error function for the rotation matrix and angular velocity are given by [32] as

$$\boldsymbol{e}_{R_k} = \log(\boldsymbol{R}^T_{d,k}\cdot\boldsymbol{R}_k)^\vee \tag{29}$$

$$\boldsymbol{e}_{\omega_k} = \boldsymbol{\omega}_k - \boldsymbol{R}^T_k\boldsymbol{R}_{d,k}\boldsymbol{\omega}_{d,k}, \tag{30}$$

where $\boldsymbol{R}_{d,k}$ and $\boldsymbol{\omega}_{d,k}$ are the desired rotation matrix and angular velocity trajectories. The terminal cost function is similarly defined.

In the stage cost expression (28), all the error terms are in linear form of the state and control vectors except the error of orientation $\boldsymbol{e}_{R_k}$, which involves matrix logarithm map as shown in (29). A linear approximation of the nonlinear error term on the rotation matrix is used. Taking the hat map on (29) and applying the matrix exponential map give

$$\exp(\hat{e}_{R_k}) = \boldsymbol{R}_{d,k}^T \boldsymbol{R}_k \approx \boldsymbol{R}_{d,k}^T \boldsymbol{R}_{op} \exp(\hat{\boldsymbol{\xi}}_k), \tag{31}$$

where the same approximation is made here as in (5). Taking the matrix logarithm of (31) and then applying the vee map give the approximate error term on $\boldsymbol{R}_k$,

$$\boldsymbol{e}_{R_k} = \log(\boldsymbol{R}_{d,k}^T \cdot \boldsymbol{R}_{op})^\vee + \boldsymbol{\xi}_k. \tag{32}$$

The error function defined in (32) is in linear form of the state variable $\boldsymbol{\xi}_k$ since both $\boldsymbol{R}_{d,k}^T$ and $\boldsymbol{R}_{op}$ are known matrices. The error function could be interpreted as the sum of (a) the geodesic between $\boldsymbol{R}_{op}$ and $\boldsymbol{R}_{d,k}$ and (b) the vector $\boldsymbol{\xi}_k$ which lies in the tangent space at $\boldsymbol{R}_{op}$. The orientation error function Ψ on $\boldsymbol{R}$ could therefore be defined as,

$$\Psi(\boldsymbol{\xi}_k) = \boldsymbol{e}_{R_k}^T \boldsymbol{Q}_R \boldsymbol{e}_{R_k} = ||\boldsymbol{e}_{R_k}||^2_{\boldsymbol{Q}_R}, \tag{33}$$

which is in quadratic form of $\boldsymbol{\xi}_k$. Given that the weighting matrix $\boldsymbol{Q}_R$ is positive definite, the orientation error function Ψ is positive definite.

The terminal cost ℓ_T is similarly defined as (28) with terminal gains. The cost of control is constructed as

$$||\boldsymbol{u}_k - \boldsymbol{u}_{d,k}||^2_{\boldsymbol{R}_u} = ||\boldsymbol{u}_{op} + \delta\boldsymbol{u}_k - \boldsymbol{u}_{d,k}||^2_{\boldsymbol{R}_u}, \tag{34}$$

where $\delta\boldsymbol{u}_k$ is the k^{th} predicted variation from the operating point $\boldsymbol{u}_{op}$.

F. Force Constraints

The force constraints are imposed to ensure that the solved GRF are physically feasible. When the foot is in contact with the ground, the normal force should be non-negative and the tangent forces should lie within the friction cone, which is prescribed as

$$\{\boldsymbol{u}_i | \boldsymbol{u}_i^n \geq 0, ||\boldsymbol{u}_i^t||_2 \leq \mu |u_i^n|\}, \tag{35}$$

where μ is the coefficient of friction; superscript $(\cdot)^t$ and $(\cdot)^n$ indicate tangential and normal force components, respectively. $||\cdot||_2$ is the 2-norm and $|\cdot|$ takes the absolute value of a scalar.

Since the friction cone constraint is a second-order cone constraint, it is not admissible to the QP formulation with linear constraints. Instead, the conservative friction pyramid [51] is used as an approximation of the friction cone. In addition, the normal force is bounded to ensure that the commanded torque does not exceed the actuator limits. The feasible control set $\mathbb{U}$ in (1) is defined as:

$$\begin{aligned} \mathbb{U}_i := \{\delta\boldsymbol{u}_i \mid & |u_{i,op}^{x/y} + \delta u_{i,k}^{x/y}| \leq \mu |u_{i,op}^z + \delta u_{i,k}^z|, \\ & u_{i,k}^{z,min} \leq u_{i,op}^z + \delta u_{i,k}^z \leq u_{i,k}^{z,max}, \\ & u_{i,k}^{z,min} \geq 0\}, \end{aligned} \tag{36}$$

where the z axis is aligned with the normal vector of the ground and x, y are two axes orthogonal to each other that lie in the tangent plane at the contact point; $u_{i,k}^{z,min}$ and $u_{i,k}^{z,max}$ are the minimum and maximum normal forces at the k^{th} predicted step for leg i. If leg i was in swing phase, then the value for both lower and upper bounds are set to zero so that the swing leg controller takes over and guides the foot towards the next foothold position. It could be observed that (36) denotes an intersection of a finite set of closed halfspaces in $\mathbb{R}^3$. Hence, $\mathbb{U}_i$ is a polyhedron. Similarly, the feasible force set $\mathbb{U}$ is also polyhedral.

G. Quadratic Program Formulation

Given the convex quadratic cost function from Section II-E, the affine dyanmics from Section II-D and linear force constraints from Section II-F, the general nonlinear MPC problem (1) could be be reformulated as a Quadratic Program (QP),

$$\text{min.} \quad \gamma^N \ell_T(\boldsymbol{x}_{t+N|t}) + \sum_{k=0}^{N-1} \gamma^k \ell(\boldsymbol{x}_{t+k|t}, \delta\boldsymbol{u}_{t+k|t}) \tag{37a}$$

$$\text{s.t.} \quad \boldsymbol{x}_{t+k+1|t} = \boldsymbol{A}\boldsymbol{x}_{t+k|t} + \boldsymbol{B}\delta\boldsymbol{u}_{t+k|t} + \boldsymbol{d} \tag{37b}$$

$$\delta\boldsymbol{u}_{t+k|t} \in \mathbb{U}_k, k = 0, \cdots, N-1 \tag{37c}$$

$$\boldsymbol{x}_{t|t} = \boldsymbol{x}(t) = \boldsymbol{x}_{op}, \tag{37d}$$

where the cost function is defined in (27); the decay rate factor $\gamma \in (0, 1]$ discounts cost further from the current moment. The definition of the affine dynamics could be found in (26) and the force constraint is expressed in (36). Note that we lifted the explicit constraints on the state vectors but instead relied on the cost function for the state regulation. The QP in (37) could be rewritten in a more compact form. Following the formulation in [52], the new optimization variable $\boldsymbol{z}$ is constructed as

$$\boldsymbol{z} = [\delta\boldsymbol{u}_0^T, \boldsymbol{x}_1^T, \cdots, \delta\boldsymbol{u}_{N-1}^T, \boldsymbol{x}_N^T]^T \in \mathbb{R}^{24N}, \tag{38}$$

such that (37) could be transcribed into the standard QP form [53],

$$\begin{aligned} \text{minimize} \quad & \frac{1}{2}\boldsymbol{z}^T \boldsymbol{P}\boldsymbol{z} + \boldsymbol{c}^T\boldsymbol{z} \\ \text{subject to} \quad & \boldsymbol{A}_{ineq} \cdot \boldsymbol{z} \leq \boldsymbol{b}_{ineq} \\ & \boldsymbol{A}_{eq} \cdot \boldsymbol{z} = \boldsymbol{b}_{eq}, \end{aligned} \tag{39}$$

where $\boldsymbol{P} \in \mathbb{R}^{N(n+m)}$ is a symmetric positive definite matrix assembled from the gain matrices $\boldsymbol{Q}_x, \boldsymbol{R}_u$; the inequality constraint $\boldsymbol{A}_{ineq} \cdot \boldsymbol{z} \leq \boldsymbol{b}_{ineq}$ imposes the force constraints; the equality constraint $\boldsymbol{A}_{eq} \cdot \boldsymbol{z} = \boldsymbol{b}_{eq}$ respects the linear dynamics.

III. Numerical Results

This section first defines a function that quantifies the closeness to singularity for Euler angles in the numerical sense. Then we present the simulation results of RF-MPC stabilizing various periodic gaits and an aperiodic 3D acrobatic maneuver. Furthermore, RF-MPC is compared with the MPC that uses Euler angles as orientation representation (EA-MPC) in the acrobatic maneuver. The resulting QPs from the MPC formulation in all of the simulations are solved using MATLAB *quadprog*. Gain values and gait pattern parameters for the following simulations could be found in Table II.

A. Singularity in Euler Angles

Orientation representations include Axis-a
and quaternion [54]. In this work, we comp
RF-MPC with EA-MPC, which is based on
[55] used on MIT Mini Cheetah, with paran

Here, we define a function to quantify th
singularity of Euler angles. Such function he
singularity and its neighborhood. Assuming
the Z-Y-X sequence in body frame $\{\boldsymbol{B}\}$, wh
to the X-Y-Z sequence in stationary inertial
Euler angles $\boldsymbol{\Theta} = [\phi\ \theta\ \psi]^T$, where ϕ is the r
and ψ is the yaw. The attitude of frame $\{\boldsymbol{B}$
a sequence of rotations in frame $\{\boldsymbol{S}\}$ as

$$\boldsymbol{R} = \boldsymbol{R}_z(\psi)\boldsymbol{R}_y(\theta)\boldsymbol{R}_x(\phi), \tag{40}$$

where $\boldsymbol{R}_x(\phi)$ means a positive rotation of angle ϕ around the x-axis of frame $\{\boldsymbol{S}\}$.

We define $\mathcal{T}_{\boldsymbol{\Theta}} : \mathbb{R}^3 \to \mathbb{R}^{3\times3}$ to be the matrix that converts $\dot{\boldsymbol{\Theta}}$ to the angular velocity expressed in $\{\boldsymbol{S}\}$ as

$$\boldsymbol{\omega} = \mathcal{T}_{\boldsymbol{\Theta}} \cdot \dot{\boldsymbol{\Theta}} = \begin{bmatrix} \cos(\theta)\cos(\psi) & -\sin(\psi) & 0 \\ \cos(\theta)\sin(\psi) & \cos(\psi) & 0 \\ -\sin(\theta) & 0 & 1 \end{bmatrix} \dot{\boldsymbol{\Theta}}. \tag{41}$$

The matrix $\mathcal{T}_{\boldsymbol{\Theta}}$ in equation (41) is invertible when $\theta \neq \pm\frac{\pi}{2}$, and $\dot{\boldsymbol{\Theta}}$ could be calculated using the following equation

$$\dot{\boldsymbol{\Theta}} = \begin{bmatrix} \cos(\psi)/\cos(\theta) & \sin(\psi)/\cos(\theta) & 0 \\ -\sin(\psi) & \cos(\psi) & 0 \\ \cos(\psi)\tan(\theta) & \sin(\psi)\tan(\theta) & 1 \end{bmatrix} \boldsymbol{\omega}. \tag{42}$$

In this work, we use the function

$$\kappa^{-1}(\mathcal{T}_{\boldsymbol{\Theta}}) \in (0, 1] \tag{43}$$

to quantify singularity in Euler angles, where $\kappa(\cdot)$ calculates the condition number of a matrix. When the robot approaches singular poses, the condition number $\kappa(\mathcal{T}_{\boldsymbol{\Theta}})$ increases rapidly, and its inverse $\kappa^{-1}(\mathcal{T}_{\boldsymbol{\Theta}})$ tends to 0.

Here, a pose control simulation is conducted to investigate the singularity of Euler angles. As shown in Fig. 3(a), the singular pose $\boldsymbol{R}_s$ is shown as the shadowed box; the desired pose $\boldsymbol{R}_d$ is shown as the solid box. All feet of the robot are assumed to be fix in this simulation so that the force constraints (36) could be lifted to focus on the effect of singularity. The desired poses are varied from the singular pose to the pose rotated 1 rad around the +y axis. A 0.5 s simulation is conducted in each desired pose and the CoM deviation at the end of the simulation is plotted for both RF-MPC and EA-MPC. As could be observed in Fig. 3(b), while RF-MPC remains stable, EA-MPC is significantly affected by singularity once $|log(\boldsymbol{R}_s^T\boldsymbol{R}_d)^\vee| < 0.3$ rad, which corresponds to $\kappa^{-1}(\mathcal{T}_{\boldsymbol{\Theta}}) < 0.15$.

B. Walking Trot

The data of a walking trot simulation is shown in Fig. 4. The robot starts from a stationary pose and accelerates in the x-direction until it reaches the final velocity of 0.5 m/s. As could be seen in Fig. 4 (a), the velocity in the x-direction reaches 0.5 m/s and the velocity deviation for all directions is within

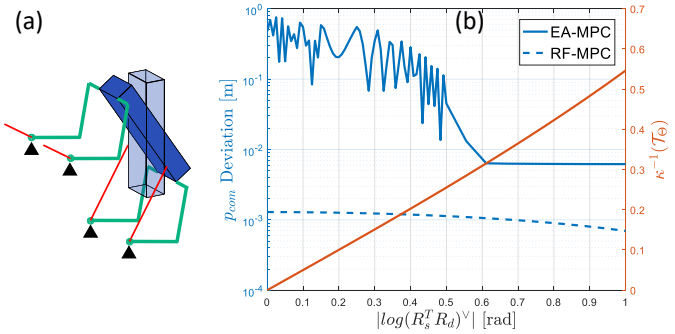

Fig. 3: The pose control simulation result of a investigation on the singularity of Euler angles. (a) The schematics of the pose control, where the shaded box represents the singular pose; the solid box represents the commanded pose; the red lines represent the GRF, with the force constraints (36) lifted. (b) The CoM position deviations (log scale) after a 0.5 s simulation of both RF-MPC and EA-MPC are plotted against $|log(\boldsymbol{R}_s^T\boldsymbol{R}_d)^\vee|$.

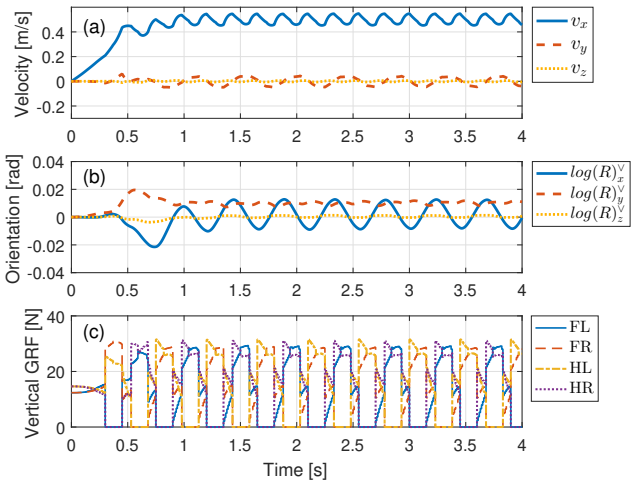

Fig. 4: Simulation data of walking trot where the robot starts from static pose and accelerates in the x-direction (a) CoM velocity; the robot accelerates at 0.5 m/s^2 and reaches the desired velocity of 0.5 m/s in the x-direction. (b) Orientation in terms of the log map of the rotation matrix (c) Vertical ground reaction forces of four legs.

±0.1 m/s. Fig. 4 (b) shows that the orientation deviation in all directions is bounded within ±0.02 rad. The vertical GRFs of all four legs are shown in Fig. 4 (c). Further details about the generation of the reference trajectory for trotting could be found in Section III-F.

The simulation is setup such that at each sampling time, the control input is applied to the original nonlinear model (4) simulated using MATLAB $ode45$ to integrate the dynamics. The gait pattern in the walking trot simulation is executed using a time-based schedule.

C. Bounding

To demonstrate the capability of RF-MPC to stabilize dynamic motions with large body attitude oscillation, the bounding simulation is presented. The bounding motion involves an aerial phase when all four feet lose contact with the ground. To stabilize the bounding motion, the reference trajectory is

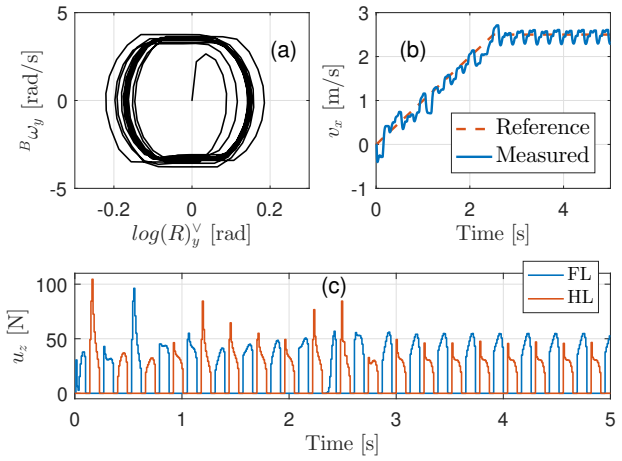

Fig. 5: Simulation data of bounding. (a) Phase portrait of body angle and angular velocity in the y-axis. (b) Velocity tracking performance in the x-direction. (c) Vertical GRFs of FL and HL legs.

designed based on the impulse-scaling principle [8] to preserve the periodicity of the gait. Details about the generation of reference trajectory could be found in Section III-F3. Here, we kindly note that an elaborate reference trajectory is optional for RF-MPC to stabilize bounding. While a trivial reference such as that used for trotting also works, the reference trajectory presented in Section Section III-F3 enables bounding motions with longer aerial phase. Similar to the walking trot, the robot is commanded to start from the static pose and accelerates at 1.0 m/s^2 to reach the final velocity of 2.5 m/s.

Fig. 5 (a) presents the phase portrait of angle and angular velocity along the y-axis. It shows that the motion converges to a periodic orbit. The velocity tracking performance shown in Fig. 5 (b) demonstrates that the MPC controller is capable of stabilizing bounding velocity up to 2.5 m/s. The vertical GRF profiles of legs FL and HL are shown in Fig. 5 (c).

D. Aperiodic Complex Dynamic Maneuver

A complex acrobatic dynamic maneuver is presented in this section to demonstrate that RF-MPC is capable of controlling aperiodic dynamic motions that involves orientations that correspond to singularities in Euler angle representation. RF-MPC is benchmarked with an MPC controller with Euler angles for its orientation representation. In addition, initial condition is perturbed to investigate the robustness of RF-MPC.

Fig. 6 (a) shows the reference trajectory of the acrobatic motion, which is a backflip with a twist. The reference CoM trajectory is colored black and the reference poses are shown in blue. The robot initially stands on a slope with the slope angle of $45°$, with the front of the robot facing upwards and body parallel to the slope. The stance phase of this acrobatic jump consists of 0.1 s of all four feet in contact, followed by 0.1 s of only the hind feet pair in contact with the slope. After the stance phase, all four feet are airborne and the robot enters the aerial phase for 0.3 s before landing. The landing direction of the robot is facing away from the slope. The feed-forward GRF and the dynamically-feasible reference trajectory are generated by solving an off-line TO problem, where the slope is set to be $45°$. Further details about the generation of the reference trajectory is provided in Section III-F.

As could be observed from Fig. 6 (a), when the robot approaches the singularity, EA-MPC becomes unstable and exerted large vertical force that pushes the robot away from the reference trajectory. As shown in Fig. 6 (c) (d), the body orientation and CoM position eventually diverges from the reference trajectory after the robot encounters singularity, which is visualized in Fig. 6 (e). In comparison, Fig. 6 (b) shows that RF-MPC could successfully stabilize the backflip motion that involves singularity. Here, we would like to point out in 6 (e) that the robot actually passes through singularity when the hind legs are in contact as indicated by the duration when $\kappa^{-1}(\mathcal{T}_{\boldsymbol{\Theta}}) < 10^{-1}$, in the light shaded area. Fig. 6 (c) (d) display that the orientation and CoM position deviation are kept small during the motion. The CoM position and orientation deviations are defined as

$$\begin{aligned} |\boldsymbol{e}_p| &= |\boldsymbol{p}(t) - \boldsymbol{p}_d(t)| \\ |\boldsymbol{e}_R| &= |log(\boldsymbol{R}_d^T(t)\boldsymbol{R}(t))^\vee|, \end{aligned} \tag{44}$$

where $\boldsymbol{p}_d$ and $\boldsymbol{R}_d$ are the reference CoM position and body orientation, respectively.

To demonstrate the robustness of RF-MPC, the slope angle for the simulated cases is changed from $45°$ to $53.6°$. Since the body of the robot is parallel to the slope at the beginning of the jump, the initial orientation of the robot is also perturbed. As could be observed from Fig. 6 (c), the backflip could be executed and stabilized by RF-MPC. In contrast, an open-loop simulation shows that in the absence of feedback control, the orientation of the robot quickly deviates from the reference trajectory due to the initial condition perturbation.

E. Comparison of Linearization Schemes

One of the crucial decisions we made in the proposed RF-MPC is to linearize the dynamics about the operating point. The choice is made because RF-MPC represents orientation using the rotation matrix, which presumes $SO(3)$ structure. Such a structure loses its validity when the predicted states are far away from the point where the linearization is performed upon. Nevertheless, a reasonable alternative is linearizing around the reference trajectory, which is a widely used technique in robotics. To investigate which linearization scheme provides more robust behavior, this section presents a simulation case study that compares MPC linearized around the reference trajectory (scheme 1) with MPC linearized around the operating point (scheme 2).

Scheme 1 linearizes dynamics around the reference trajectory, which includes the desired state $\{\boldsymbol{x}^d_{t+k|t}\}$ and control $\{\boldsymbol{u}^d_{t+k|t}\}$ within the prediction horizon, where $k = 0, \cdots, N-1$ and N is the prediction horizon. Hence, $\boldsymbol{A}_k$ and $\boldsymbol{B}_k$ are matrices for a Linear Time-Varying (LTV) system, parametrized by the reference trajectory within the prediction horizon. Scheme 2 linearizes dynamics around the operating point, which involves current state $\boldsymbol{x}_{op}$ and control $\boldsymbol{u}_{op}$, as

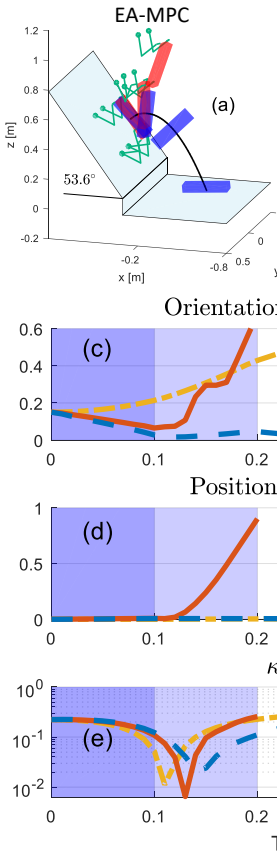

Fig. 6: Simulation results of a cc
the robot performs a twist jump
poses are shown in blue and the
in red; the reference CoM traje
shaded area is when four legs are in contact with the surface; the light-shaded area is when only hind legs are in contact and the non-shaded area is when the robot is in aerial phase. (a) The EA-MPC becomes unstable when the robot is close to the singular pose. (b) RF-MPC could track the reference motion that involves singular poses. (c) The orientation deviation $|\boldsymbol{e}_R|$ of open-loop control, RF-MPC, and the EA-MPC. (d) The CoM position deviation. (e) The function to quantify the distance to singularity $\kappa^{-1}(\mathcal{T}_{\boldsymbol{\Theta}})$.

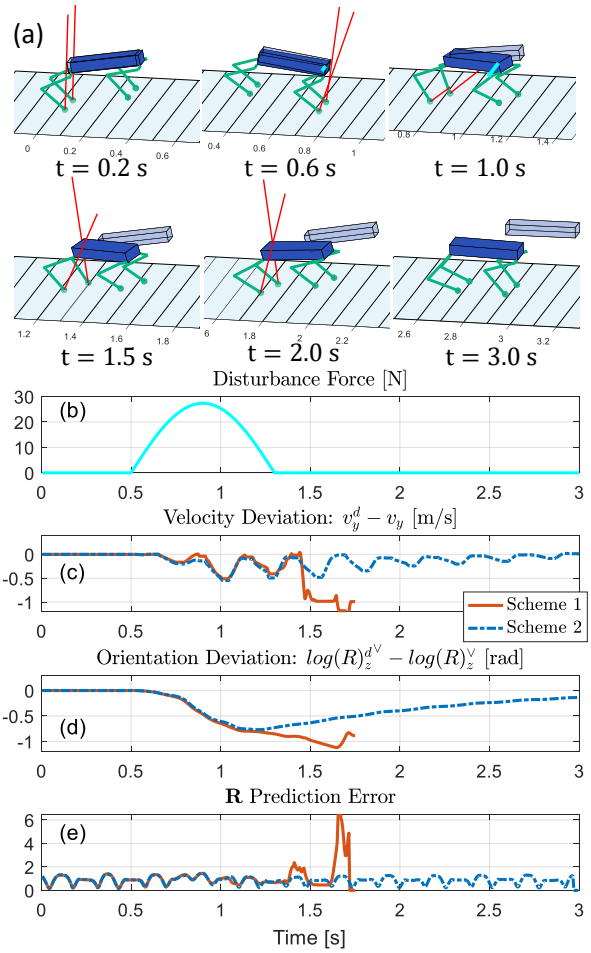

Fig. 7: Comparison between two linearization schemes. (a) A sequential snapshot of the simulation scenario for scheme 2. The GRFs are shown in red and the disturbance force is shown in cyan. The translucent box represents the reference pose of the robot. (b) Disturbance force profile. (c) Velocity deviation and (d) Orientation deviation in the y-direction from the reference trajectory, respectively. (e) Prediction error of the Rotation matrix.

defined in Section II. Constant matrices $\boldsymbol{A}|_{op}$ and $\boldsymbol{B}|_{op}$ are used to propagate the state through the prediction horizon using a Linear Time-Invariant (LTI) system.

The robustness of these two linearization schemes are qualitatively compared by examining how much external disturbance they can handle. The simulation is setup such that robot is bounding at a constant speed of 1.0 m/s in the $+x$ direction. The disturbance with a maximum force of 27 N is applied on the robot in the $+y$ direction, causing it to deviate from the reference trajectory. The reference trajectory, controller gain, gait timing and disturbance are same for both schemes, with only the linearization scheme being different.

Fig. 7 (a) shows a sequential snapshot of the simulated scenario for scheme 2. The GRFs are shown in red and the disturbance force (visible at t = 1.0 s and t = 1.5 s) is in cyan. Fig. 7 (b) shows the disturbance force profile, which is applied at the FR shoulder of the robot.

RF-MPC using scheme 1 fails at 1.5 s since the velocity and orientation start to diverge from the reference, as shown in Fig. 7 (c) and (d), respectively. In comparison, the RF-MPC using scheme 2 recovers from the disturbance and successfully tracks the reference trajectory. To investigate the reason for the discrepancy, we defined a quantity that measures the prediction quality of the rotation matrices,

$$\Phi(t) = \sum_{k=1}^{N} ||\widetilde{\boldsymbol{R}}_k - \overline{\boldsymbol{R}}_k||_F + ||log(\boldsymbol{R}_{op,t+k}^T \widetilde{\boldsymbol{R}}_k)^{\vee}||, \quad (45)$$

where $\boldsymbol{R}_{op,t+k} \in SO(3)$ is the rotation matrix at time $t + k \cdot dt$; $\overline{\boldsymbol{R}}_k$ is the k^{th} predicted rotation matrix at time t, whose projection to the $SO(3)$ manifold is denoted as $\widetilde{\boldsymbol{R}}_k \in SO(3)$; $||\cdot||_F$ is the Frobenius norm of a matrix. The prediction error of rotation matrices is plotted in Fig. 7 (e), where the error value of scheme 1 became high when the system started to deviate from the reference trajectory. This numerical study serves as an empirical validation of the robustness of scheme 2 in comparison to scheme 1 in disturbance rejection.

F. Reference Trajectory Generation

1) ***Trotting****:* The reference control trajectory $\boldsymbol{u}_{k,d}$ for trotting is defined based on the heuristic that the total weight of the robot should be supported evenly by all the legs that are in contact:

$$\begin{aligned} u^{x/y}_{i,k,d} &= 0 \\ u^{z}_{i,k,d} &= \frac{b_{i,k}}{\sum_{i=1}^{4} b_{i,k}} Mg, \end{aligned} \tag{46}$$

where $b_{i,k} \in \{0, 1\}$ is a binary variable that indicates the contact condition of leg i at instance $t + k$, where $b_{i,k} = 1$ indicates contact phase and 0 otherwise. The value of the binary variable $b_{i,k}$ is defined according to the time schedule of a Finite State Machine (FSM), which is introduced in Section IV-A. The reference state $\boldsymbol{x}_{k,d}$ is constructed by simply assuming the robot accelerate from static pose with constant acceleration until reaching the maximal velocity. Both walking trot and running trot use the same reference trajectory.

2) ***Aperiodic Complex Dynamic Maneuver****:* The reference trajectory is generated by an off-line TO algorithm based on the single-shooting method. The twist jump motion is decomposed into three phases with fixed timing. Phase one is when four feet are in contact, which lasts for 0.1 s; phase two is when front legs lift off and hind legs are in contact, which lasts for 0.1 s; phase three is when the robot is airborne, which lasts for 0.3 s. The optimization variables are the magnitude of the GRFs, which is assumed to be constant throughout phases one and two. The cost function is the weighted sum of the control effort and the deviation from the desired landing pose. The constraints imposed in the optimizations are

- Fixed initial state and bounded final state
- Fixed contact sequence and timing
- Kinematic reachability of each leg
- Collision avoidance with the environment
- Unilateral GRF stay within friction cone.

The TO is formulated as a nonlinear program with 24 variables, representing the force magnitude of 4 GRFs (each has 3 components) in two stance phases. The optimization problem is solved by the MATLAB NLP solver *fmincon*.

3) ***Bounding****:* The periodic trajectory for bounding gait is generated by considering the robot as a single rigid body in 2D, which has 3 DoFs (x, z, θ). The contact sequence and timing is pre-specified as front-stance, aerial phase I, hind-stance and aerial phase II. The shapes of vertical GRF and pitch torque profiles are parametrized by Bézier polynomials. Periodicity in the z and θ directions is achieved by finding the scaling factors and initial condition based on the principle of impulse-scaling [8] analytically.

4) ***Controlled Backflip****:* To generate the reference trajectory of the controlled backflip in Section V-D, we used the open-source *OptimTraj* library [57] to set up the TO problem using the direct collocation method. The optimization is done on a 2-D single rigid body model of the robot. The convex quadratic cost function penalizes large GRF and rewards smooth force profiles. In addition to the constraints mentioned in *2)*, the following constraints are also imposed

- The constraint that enforces feasible dynamics
- Path constraints on the state and control.

The above problem setup has 27 time steps, which results in an optimization problem with 272 variables and 366 constraints solved by MATLAB NLP solver *fmincon*.

IV. Controller Implementation Details

The MPC framework in Section II is combined with other components such as state estimation and swing leg controllers to give rise to various motions implemented on the robot hardware platform. This section presents the implementation details that are required to realize the MPC control design on the hardware.

Fig. 8 shows the schematics of the overall control system. The Finite State Machine (FSM) sends the desired state and control trajectories $\boldsymbol{X}_d, \boldsymbol{U}_d$ to the MPC, which formulates a Quadratic Program (QP). The QP is solved by the custom QP solver qpSWIFT [58] to obtain the optimal solution $\delta\boldsymbol{u}$, which is added to the control at the operating point $\boldsymbol{u}^{-}_{op}$ to get the GRF $\boldsymbol{u}_{op}$. A swing leg controller calculates the swing force $\boldsymbol{u}_{sw}$ to track the swing foot trajectories. The commanded torque is modified by a lower-level joint controller, which compensates for friction and motor dynamics. The Brush-Less Direct Current (BLDC) motors actuate the robot to interact with the environment.

A. Finite State Machine

Various gaits are generated by a finite state machine (FSM). Fig. 9 shows the schematics of the FSM where the timing schedules are sent from the gait planner to each leg. A leg independent phase variable s_i quantifies the percentile completion of either stance or swing state. The phase variables are defined as $s_i := \{\bar{t}/T_j \text{ s.t. } j \in \{st, sw\}\}$, where $\bar{t}$ represents the current dwell time, T_{st}, T_{sw} are stance and swing times, respectively. The period of the gait is the sum of the dwell times $T = T_{st} + T_{sw}$. The guard sets G_i and reset maps Δ_j define the transition between states. The guard sets are given as $G_i := \{\bar{t} \text{ s.t. } \bar{t} = T_j\}$. The reset map is defined as $\Delta_j(\bar{t}) = 0$ such that it resets the phase variable and current dwell time. This framework allows the implementation of any gait sequence by changing the timing schedules. The contact detection algorithm could be incorporated to adjust the gait timings and extend the time-based FSM to event-based FSM. Using the FSM scheme, trotting, bounding and aperiodic motions could be realized.

It is worth noting that the prediction horizon could cover multiple phases. Hence, in motions with aerial phases such as bounding and acrobatic jump, the RF-MPC could take into consideration of the upcoming phase change and plan the current control accordingly.

B. Platform Description

The hardware platform used for the experiments is a 5.5 kg fully torque controllable, electrical quadruped robot named *Panther*. Three custom-made BLDC motor units are assembled into a leg module [59] which is capable of executing highly dynamic maneuvers [60] [61]. The body of *Panther* is

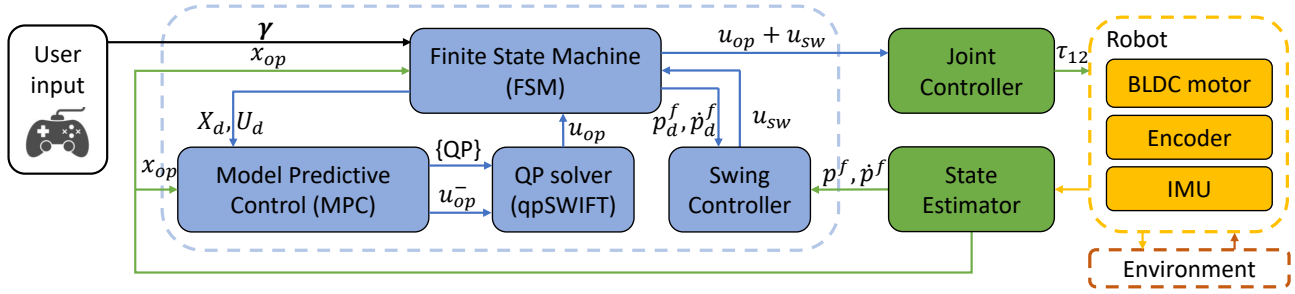

Fig. 8: Overview of the control system. The user sends commands to the on-board computer (blue), where the finite state machine schedules the gait and sends desired trajectories to the MPC block to formulate the QP. The custom QP solver qpSWIFT solves for the $\boldsymbol{u}_{op}$ and send it to the FSM. The FSM combines the stance and swing forces and send to the joint controller (green), which maps leg forces to joint torque ormulation of the next cycle. The previous

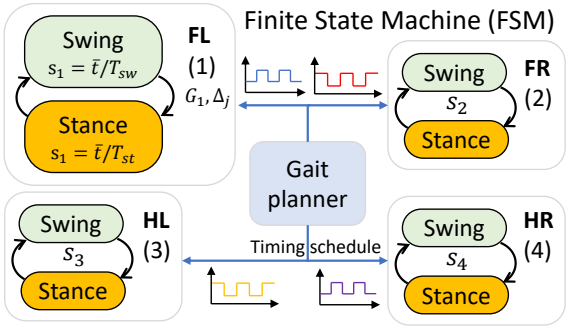

Fig. 9: Schematics of the Finite State Machine (FSM). The gait planner sends to all legs the timing schedules; the normalized variable s_i is the percentile completion of the current state. $\Delta_j, j \in \{st, sw\}$

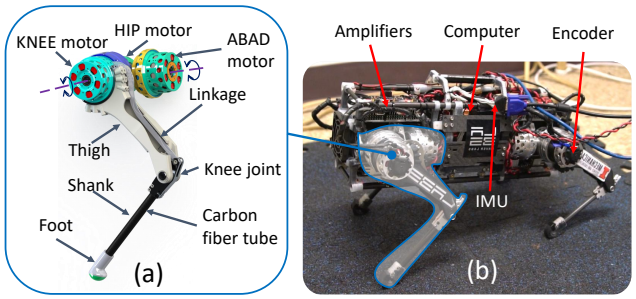

Fig. 10: An illustration of the hardware platform. (a) CAD model of the mechanical components of the leg module, which includes ABAD, HIP and KNEE modules and linkages of the leg. (b) A picture of the assembled quadruped platform, which integrates a computer, sensors including an IMU and 12 encoders.

assembled from carbon fiber reinforced 3D printed parts that connect carbon fiber tubes and plates for higher strength to weight ratio. The point foot is cushioned with sorbothane for its shock absorption capability. The electronic system consists of an on-board computer PC104 with Intel i7-3517UE at 1.70 GHz, Elmo Gold Twitter amplifiers, RLS-RMB20 magnetic encoders on each joint, and an inertial measuring unit (IMU) VN-100. Fig. 10 shows the CAD model of the mechanical components of a leg module and the picture of the assembled quadruped platform with all the parts integrated.

C. Computation Setup

The MPC framework is implemented using Simulink Real-Time (SLRT). The encoder readings and lower-level kinematics calculations are carried out at a base rate of 4 kHz, while the IMU signals are received and state estimation is performed at 1 kHz. The user input from the joystick is updated at 23 Hz, and the QP is solved at between 160 Hz to 250 Hz depending on the size of the problem. The proposed QP (39) is solved by a custom QP solver qpSWIFT [58] for all the experiments. Written in ANSI C, the solver is library-free while and it interfaces with SLRT through a gateway *s-function*. A RF-MPC with prediction horizon of 6 entails solving a QP with 144 variables, 72 inequality and 72 equality constraints.

D. Swing Leg Control

Since the leg mass is less than 10% of the total mass of the robot and the motor inertia is much smaller than body inertia, the inertia effect of the legs could be neglected during stance. Nevertheless, leg inertia is considered when designing the swing leg controller. The swing leg is modeled as a 3-link serial manipulator attached to a stationary base. The swing leg controller consists of feed-forward and feedback terms, where the former is based on the workspace inverse dynamics,

$$\boldsymbol{\tau}_{sw}^{ff} = \boldsymbol{D}(\boldsymbol{q})\boldsymbol{J}^{-1}(\boldsymbol{a}_x^f - \dot{\boldsymbol{J}}\dot{\boldsymbol{q}}) + \boldsymbol{h}(\boldsymbol{q}, \dot{\boldsymbol{q}}), \tag{47}$$

where $\boldsymbol{\tau}_{sw}^{ff}$ is the feed-forward torque; $\boldsymbol{D}(\boldsymbol{q})$ is the Inertia matrix and $\boldsymbol{h}(\boldsymbol{q}, \dot{\boldsymbol{q}})$ includes the centrifugal, Corolis and gravitational terms of the swing leg; $\boldsymbol{q}, \dot{\boldsymbol{q}}$ are the joint angle and velocity vectors; $\boldsymbol{J}$ is the leg Jacobian matrix and $\dot{\boldsymbol{J}}$ is its time derivative; $\boldsymbol{a}_x^f$ is workspace acceleration vector, which is defined as

$$\boldsymbol{a}_x^f = \ddot{\boldsymbol{p}}_d^f + \boldsymbol{K}_p^{ff}(\boldsymbol{p}_d^f - \boldsymbol{p}^f) + \boldsymbol{K}_d^{ff}(\dot{\boldsymbol{p}}_d^f - \dot{\boldsymbol{p}}^f), \tag{48}$$

where $\ddot{\boldsymbol{p}}_d^f$ is the desired foot workspace acceleration; $\boldsymbol{p}_d^f, \dot{\boldsymbol{p}}_d^f$ are the desired foot position and velocity; $\boldsymbol{K}_p^{ff}$, $\boldsymbol{K}_d^{ff}$ are the position and velocity gain matrices. The full swing leg controller consists of both feed-forward and feedback terms,

$$\boldsymbol{\tau}_{sw} = \boldsymbol{\tau}_{sw}^{ff} + \boldsymbol{K}_p^{fb}(\boldsymbol{p}_d^f - \boldsymbol{p}^f) + \boldsymbol{K}_d^{fb}(\dot{\boldsymbol{p}}_d^f - \dot{\boldsymbol{p}}^f), \tag{49}$$

where $\boldsymbol{K}_p^{fb}$ and $\boldsymbol{K}_d^{fb}$ are the position and velocity gain matrices for the feedback term of the swing leg controller.

The desired foot placement is a linear combination of a velocity-based feed-forward term and a capture-point [62] based feedback term.

$$\boldsymbol{p}_{step}^{f} = \boldsymbol{p}^{h} + \frac{T_{st}}{2}\dot{\boldsymbol{p}}_{d}^{h} + \sqrt{\frac{z_0^h}{g}}(\dot{\boldsymbol{p}}^{h} - \dot{\boldsymbol{p}}_{d}^{h}), \tag{50}$$

where $\boldsymbol{p}_{step}^{f}$ is the desired step location on the ground plane; $\boldsymbol{p}^{h}$ is the projection of the hip joint on the ground plane and $\dot{\boldsymbol{p}}^{h}$ is the corresponding velocity; $\dot{\boldsymbol{p}}_{d}^{h}$ is the desired hip velocity projected on the ground plane; T_{st} is the stance time; g is the gravitational acceleration constant; z_0^h is the nominal hip height.

E. State Estimation

Kalman Filter [63] has been applied for a range of applications in legged robots. Meanwhile, simple linear single-input single-output (SISO) complementary filters [64] has been proven to work robustly in practice [65] [66]. The complementary filter performs low-pass filtering on a low-frequency estimation and high-pass filtering on a biased high-frequency estimation. For instance, the CoM velocity $\dot{\boldsymbol{p}}$ is obtained by combining the CoM velocity estimate from leg kinematics data $\dot{\boldsymbol{p}}^{enc}$ and the accelerometer readings $\boldsymbol{a}^{acc}$ from the on-board IMU.

$$\begin{aligned} \dot{\boldsymbol{p}}_{k+1} &= \dot{\boldsymbol{p}}_k + \boldsymbol{a}_k \cdot \Delta t \\ \boldsymbol{a}_k &= \boldsymbol{a}_k^{acc} - \boldsymbol{K}_p^{v}(\dot{\boldsymbol{p}}_k - \dot{\boldsymbol{p}}_k^{enc}), \end{aligned} \tag{51}$$

where $\dot{\boldsymbol{p}}_k$ is the estimated velocity from the previous iteration; the subscript $(\cdot)_k$ is the discrete time index; Δt is the IMU sampling period; $\boldsymbol{K}_p^{v}$ is a positive diagonal gain matrix; $\boldsymbol{a}_k^{acc}$ is the accelerometer reading; $\dot{\boldsymbol{p}}_k^{enc}$ is the average of all the velocities from contact feet to CoM based on kinematic calculations. Similarly, the CoM position $\boldsymbol{p} \in \mathbb{R}^3$ is estimated by fusing the CoM position estimate from leg position kinematics $\boldsymbol{p}^{enc}$ and the estimated CoM velocity $\dot{\boldsymbol{p}}$.

F. Contact Detection

Contact sensing plays a crucial role in legged locomotion. However, conventional force estimation is fragile and noisy, which is not suitable for dynamic locomotion applications. Proprioceptive sensing [67] is utilized in this work because of the highly-transparent actuation design. We use the generalized momenta based disturbance observer [68], which only requires proprioceptive measurements $\boldsymbol{q}, \dot{\boldsymbol{q}}$ and the commanded torque $\boldsymbol{\tau}$. In this work, only the knee joints are considered in contact detection based on the assumption that the knee joint momentum is changed the most by the contact impact. The residual vector $\boldsymbol{r}_k$ is defined as,

$$\boldsymbol{r}_k = \boldsymbol{K}_I \cdot [\boldsymbol{I}^{kn}\dot{\boldsymbol{q}}_k^{kn} - \sum_{i=1}^{k}(\boldsymbol{\tau}_i^{kn} + \boldsymbol{r}_{i-1})\Delta t], \boldsymbol{r}_0 = 0, \tag{52}$$

where $\boldsymbol{r}_k \in \mathbb{R}^4$ is the residual vector for the four legs. k is the index for the current instance; $\boldsymbol{K}_I$ is a diagonal gain matrix; $\boldsymbol{I}^{kn}$ is the diagonal inertia matrix for all the knee joints; $\dot{\boldsymbol{q}}_k^{kn}$ is the vector of knee joint velocity; $\boldsymbol{\tau}_k^{kn}$ is the commanded knee torque; $\boldsymbol{r}_0$ is the initial value of the residual. The summation accumulates all the previous residuals and the commanded torque. Contact is declared when the residual vector $\boldsymbol{r}_k$ exceeds a threshold value r_{th}.

G. System Identification

*1) **Friction Compensation**:* The gear ratio of the planetary gearbox in each motor module is 23.36:1, which is higher than other quadruped robots with proprioceptive actuation scheme, including MIT Cheetah 3 (7.67:1) [9], Mini Cheetah (6:1) [13] and Minitaur (1:1) [4]. Due to the relatively higher gear ratio, the friction induced by the gearbox and bearing is modeled and compensated for more accurate force control. Following [13], the friction is modeled by the expression

$$\tau_{friction} = c_1 \cdot \text{sat}(\omega) + c_2 \cdot \tau_{motor} \cdot \text{sat}(\omega), \tag{53}$$

where ω is the output angular velocity; τ_{motor} is the commanded motor torque amplified by the gear ratio; c_1, c_2 are tunable constants that are motor-specific. $\tau_{friction}$ is the friction compensation term and the output torque $\tau_{output} = \tau_{motor} + \tau_{friction}$. The saturation function is defined as

$$\text{sat}(\omega) = \begin{cases} -1 & \omega \leq -\omega_{thr} \\ 1/\omega_{thr} & -\omega_{thr} < \omega \leq \omega_{thr} \\ 1 & \omega_{thr} < \omega, \end{cases} \tag{54}$$

which serves as a relaxed version of the sign function. The threshold value ω_{thr} could be tuned to prevent chattering around the equilibrium point.

*2) **Center of Mass Location**:* The CoM location estimation from the CAD model of small robots is less accurate than that of larger robots. That is because for small robots, a large portion of the body mass is occupied by the electronics, whose mass distribution cannot be exactly captured by the CAD model. Instead, we obtained the CoM location by suspending the robot by a string. When robot is stationary, the accelerometer reading is recorded. This procedure is repeated for several other known attachment points on the robot. A bundle of lines could be constructed from the readings of the accelerometer and the position of the attachment points obtained from the CAD model. The CoM location could be obtained by solving a least-square problem,

$$\underset{\boldsymbol{p}_{CoM}}{\text{argmin}} \sum_i ||\boldsymbol{p}_{CoM} - l_i||^2, \tag{55}$$

where $\boldsymbol{p}_{CoM}$ is the CoM location; l_i is the bundle of lines constructed from the accelerometer readings. The norm takes the smallest distance from the point to the line. The legs are commanded to a stationary nominal position throughout the experiment.

*3) **Mass Moment of Inertia**:* Mass moment of inertia ${}^{B}\boldsymbol{I}$ is an important parameter for the dynamic modeling of the robot. However, the value directly obtained from the CAD model for a small robot may not be as accurate due to the the unknown mass distribution of electronics. Therefore, a linear version of the bifilar (two-wire) torsional pendulum [69] is used to obtain the mass moment of inertia.

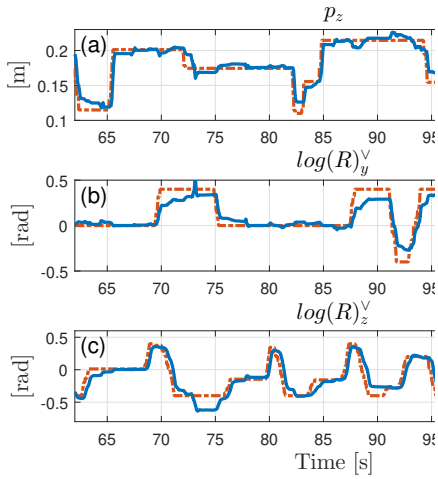

Fig. 11: Pose control experiment data. (a) performance for p_z, (b) and (c) present th performance in the y and z directions.

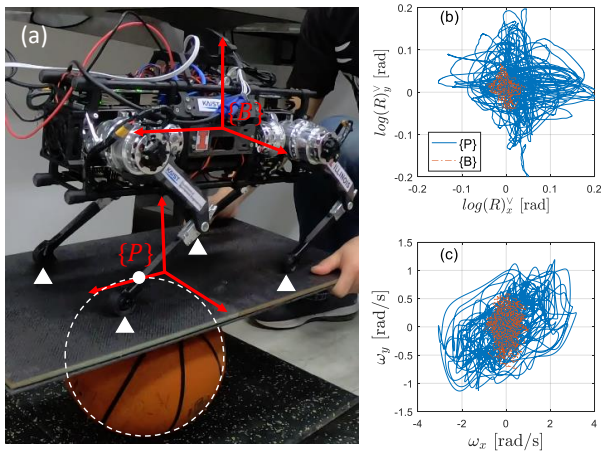

Fig. 12: The balancing control experiment (a) Experimental setup. The robot stands on a platform pivoted on a sphere, the pivot point is shown as a solid circle. The four triangles indicate the foot contact points. (b) The orientation deviation of the platform coordinate $\{P\}$ (blue) and the body coordinate $\{B\}$ (red). (c) The body angular velocity of the platform coordinate (blue) and the body coordinate (red).

V. Experiment Results

The proposed RF-MPC controller is a general motion control framework which could be used to achieve multiple motion objectives. This section presents the experimental results of some common tasks for quadrupedal robots, including pose control, balancing on a moving platform, and periodic locomotive gaits such as walking trot, running trot and bounding. In addition, a controlled backflip experiment is presented to show that the RF-MPC framework is capable of controlling dynamic motions previously hard to achieve because of the presence of singularity. The gain values and the gait timing for all experiments could be found in Table II. Clips of all the experiments could be found in the supplementary video.

A. Pose and Balancing Control

To exhibit the tracking performance of the MPC controller, the pose control experiment is conducted. The experimenter sends the desired CoM vertical height and orientation commands in y and z directions to the robot from the joystick. The MPC controller continuously solves for the desired GRFs at the four feet, which are in contact with the ground throughout the experiment. The position and orientation reference tracking data shown in Fig. 11 suggests that RF-MPC could closely track the reference command. To demonstrate the capability of RF-MPC to balance its body under large disturbances, the balancing experiment is presented. The experimental setup is shown in Fig. 12 (a). The robot stands on a pivoted platform, attempting to maintain the balance at the nominal standing pose when the platform is perturbed by the operator. The robot body coordinate $\{B\}$ and the coordinate of the platform $\{P\}$ are both plotted in Fig. 12 (a). The origin of $\{P\}$ is set at the center of the four feet. Fig. 12 (b) shows the orientation deviation in the x and y directions for $\{P\}$ in blue and $\{B\}$ in red; Fig. 12 (c) shows the angular velocity in the x and y directions. As shown in Fig. 12, the balancing controller significantly reduces the movement of the robot's body frame $\{B\}$ compared to that of the platform-fixed frame $\{P\}$.

B. Walking Trot

To demonstrate that RF-MPC can stabilize basic locomotion gaits, the walking trot experiment is presented. The robot could move in any direction parallel to the ground while maintaining the body orientation. Fig. 13 (a) and (b) exhibit the velocity tracking performance of the controller, and Fig. 13 (c) shows that the orientation deviation is kept small (within ± 0.06 rad) during the walking trot experiment; Fig. 13 (d) presents the vertical GRF during the walking trot. The velocities are measured from the state estimation.

C. Running Trot and Bounding

To investigate the performance of RF-MPC for dynamic gaits, experiments of running trot and bounding gaits with full aerial phases are conducted. Fig. 14 presents the running trot experiment data, where Fig. 14 (a) shows that the vertical CoM velocity experiences 40 ms free fall during the aerial phase. Fig. 14 (b) shows that the robot could produce abruptly changing GRF as the contact condition changes. During the trot running experiment, the vertical GRF could reach as high as 60 N while the knee torque goes up to 6.3 Nm, as could be observed in Fig. 14(b) and (c), respectively.

The bounding gait leverages the full dynamics of the robot and involves extensive body pitch oscillation. A sequential snapshots of the bounding experiment could be found in Fig. 15 (a). The robot starts from a static pose and the MPC stabilizes the robot to follow the desired GRF and state trajectories. More details about reference trajectory generation could be found in Section III-F. The reference and measured trajectories of orientation and angular velocity in the y-direction are shown in Fig. 15 (b), (c). Since the robot started from a static pose, there is an initial offset. The vertical GRF profile is shown in Fig. 15 (d). The transition from swing to stance phase occurs when a touchdown event is declared by the contact detection algorithm described in Section IV-F.

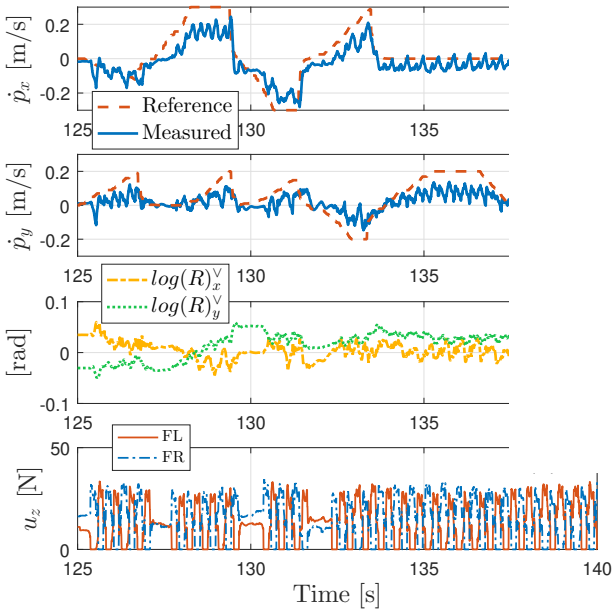

Fig. 13: Walking trot experiment data. (a) Velocity tracking in the x-direction. (b) Velocity tracking in the y-direction (c) Orientation deviations along the x and y-axes, where the reference is 0. (d) Vertical GRF u_z for front legs.

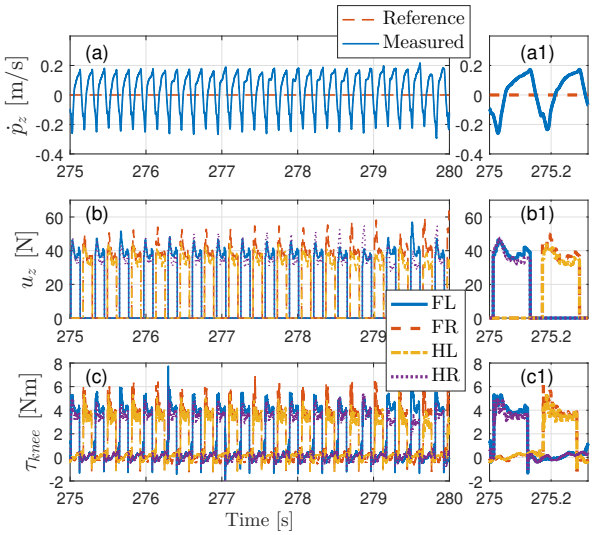

Fig. 14: Running trot experiment data. Zoomed-in views placed at the right of the figure show the details of the signals. (a) Reference and measured CoM vertical velocity in the z-direction (b) Vertical GRF (c) Knee torque.

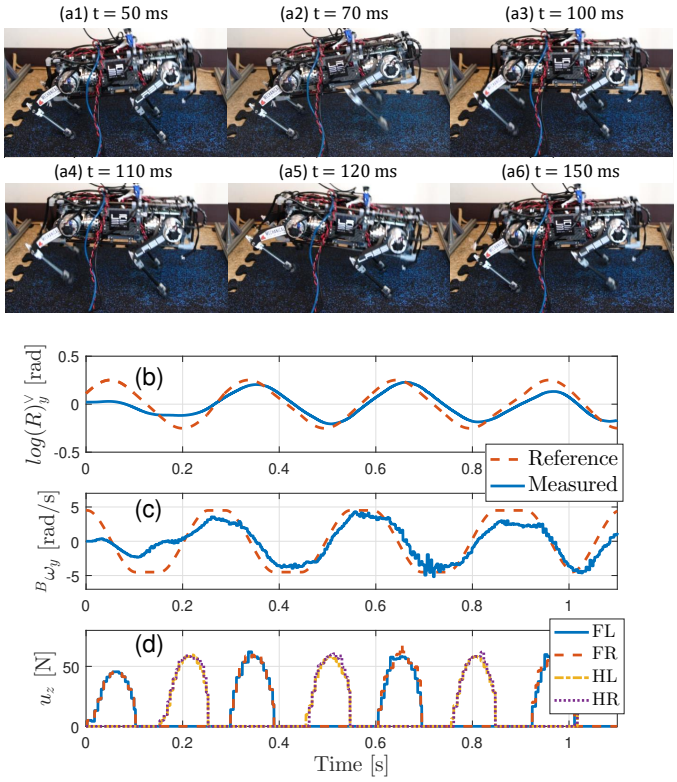

Fig. 15: Bounding experiment data (a) Sequential snapshots of the robot in a bounding experiment. (b) Orientation tracking in the y-direction $log(R)^\vee_y$ (c) Angular velocity tracking in the y-direction $^B\omega_y$ (d) Vertical GRF.

D. Controlled Backflip

To demonstrate the capability of RF-MPC to control dynamic maneuvers that involve singularity poses, a controlled backflip experiment is presented. As shown in Fig. 1, the robot flips backwards around the y-axis, passing through the pose where the robot is upright, before it lands with the upside-down orientation. Note that even though the reference trajectory is generated based on a 2-D model of the robot as presented in Section III-F, RF-MPC controls the 3D robot in the controlled backflip experiment without resorting to decomposition of sagittal plane motion and out-of-plane motion.

The image sequence in Fig. 16 (a) is plotted along with the body orientation, which is reconstructed from the experiment data as shown in Fig. 16 (b). The rotation matrix is represented by the body coordinate frame axes (x-blue, y-red, z-green); the three color-coded rings correspond to the Euler angles with the roll-pitch-yaw sequence convention (roll-blue, pitch-red, yaw-green). Note that the robot is at the singular pose at around 300 ms as shown in Fig. 16 (a3). In the corresponding body orientation plot, the axes of the rings for Euler angles almost coincide. Therefore, the robot indeed passes through the singularity pose when the RF-MPC is actively controlling the hind legs to track the reference trajectory. To the best of the authors' knowledge, this is the first instance of hardware experimental implementation of MPC to control acrobatic motion which involves singularity.

Fig. 17 presents the data from the controlled backflip

experiment, where the robot goes through three phases. The deep-shaded area corresponds to the phase when all four legs are in contact; the light-shaded area indicates the phase when only the hind legs are in contact; the non-shaded area corresponds to the landing phase. The RF-MPC controller is activated during the first two phases, and an impedance control is utilized in the third phase. Experiment data gathered from 10 backflip trials are shown in Fig. 17, where the solid lines are the mean values of all the tests, and the shaded tube is the value within one standard deviation.

As could be observed from Fig. 17 (d), the robot passes through the neighborhood of singularity as the number introduced in Section III-A drops below the threshold 10^{-1}. Fig. 17 (c) shows that the pitch angle θ is not monotonic throughout the controlled backflip while $log(R)_y^\vee$ decreases monotonically. The dash-dot curves in Fig. 17 (a) and (c) are from the experiment trial where the initial state of the robot is perturbed. Specifically, the height of the stage on which the front legs are positioned is increased from 80 mm to 130 mm. It could be observed that while in this case the trajectory of the robot deviates more than one standard deviation from the average, RF-MPC could still stabilize the motion and land safely.

VI. Discussion

In this section we briefly discuss some of the important aspects of this work. That includes the interpretation of some of the experiment results, discussion of the findings, and limitations of this work.

In this work, the proposed RF-MPC uses the rotation matrix to represent the orientation, which is capable of stabilizing dynamic motion in 3D that involves singularity in the Euler angle formulation. Specifically, Section V-D presents the controlled backflip experiment, where RF-MPC stabilized the robot to perform an acrobatic maneuver that passes through the singularity. The simulation result in Section III-D suggests that the function value dropping below the threshold would result in the failure of the EA-MPC. Another problem EA-MPC has is shown in Fig. 17 (c), where the pitch angle θ decreased until $-\frac{\pi}{2}$ and went back to 0 rad. Notice how the monotonicity changed as the value of $\kappa^{-1}(\mathcal{T}_{\boldsymbol{\Theta}})$ went below the threshold of 10^{-1} s shown in Fig. 17 (d). In contrast, $log(R)_y^\vee$ monotonically decreased to $-\pi$. Note that though the motion of the controlled backflip remains in the sagittal plane, RF-MPC is stabilizing the 3D motion without decomposing the motion into in-plane and out-of-plane parts. This experiment is our initial demonstration of the RF-MPC framework that could potentially open up the possibilities of controlling legged robots to perform the extremely agile motions as shown in [3].

One of the findings is that, a simulation case study shown in Section III-E suggests that within the RF-MPC framework, linearizing around the operating point (scheme 2) provides more robust behavior compared with linearizing around the reference trajectory (scheme 1). The result is counter-intuitive because scheme 1 uses time-varying Jacobian matrices $\boldsymbol{A}_k, \boldsymbol{B}_k$ parametrized by the reference trajectory within the prediction horizon, while scheme 2 uses matrices parameterized only by the operating point $\boldsymbol{A}|_{op}, \boldsymbol{B}|_{op}$ throughout the prediction horizon. Namely, scheme 1 utilizes more information than scheme 2. Our conjecture for the reason why scheme 2 provides more robustness is stated as follows. Since RF-MPC represents orientation using the rotation matrix, which presumes $SO(3)$ structure, linearizing around the operating point guarantees an accurate dynamics model for predicted states that are close to the operating point. In comparison, when the orientation deviation from the reference trajectory is large, the dynamics linearized about the reference no longer provide a realistic local approximation of the original nonlinear dynamics. This phenomenon is shown in Fig. 7(e), where the prediction error of the rotation matrix in scheme 1 became large, which leads to the failure of the controller. The decay rate γ is used in both schemes to discount the effect of states that are farther in the future, where the linearized model is less accurate.

A limitation of the proposed RF-MPC formulation is that the predicted rotation matrices constructed from $\boldsymbol{\xi}$ are not elements of the $SO(3)$ manifold since the first order approximation is unable to fully capture the $SO(3)$ structure. Hence, the prediction error is more pronounced for a longer prediction horizon. Currently, prediction horizon remains an important design parameter with a trade-off between the predictive ability of MPC and the accuracy of the linearized dynamics model. Prediction horizon being too long leads to inaccurate rotational dynamics, while being too short leads to myopic behaviors. To mitigate this issue, we envision a hierarchical framework with multiple MPCs running at different rates. Specifically, an MPC with simpler model and longer prediction horizon could be running at lower update rate, while the RF-MPC with shorter prediction horizon could be running at a higher rate.

The larger deviation towards the end of the bounding motion may be caused by the simple state estimation and contact detection algorithms presented in Sections IV-E and IV-F, respectively. The tracking error in Fig. 11, 13, and 15 could also be affected by these reasons since the velocities were measured from state estimation instead of external sensors.

VII. Conclusion and Future Work

In this work we presented a representation-free model predictive control framework that directly represents orientation using the rotation matrix instead of using other orientation representations. Despite the local validity of linearized dynamics on the rotation matrix, this approach introduces the possibility to stabilize 3D complex acrobatic maneuvers that involve singularities in the Euler angles formulation. By directly working on the rotation matrix, this method avoids issues arising from the usage of other representations such as unwinding phenomenon (quaternion) or singularity (Euler angles). The application of a variation-based linearization scheme and a vectorization routine linearized the nonlinear dynamics and transformed the matrix variables into vector variables. The deliberate construction of the orientation error function enabled us to formulate the MPC into the standard QP form.

We reported both simulation and experiment results of the RF-MPC controller applied on the quadruped *Panther* robot.

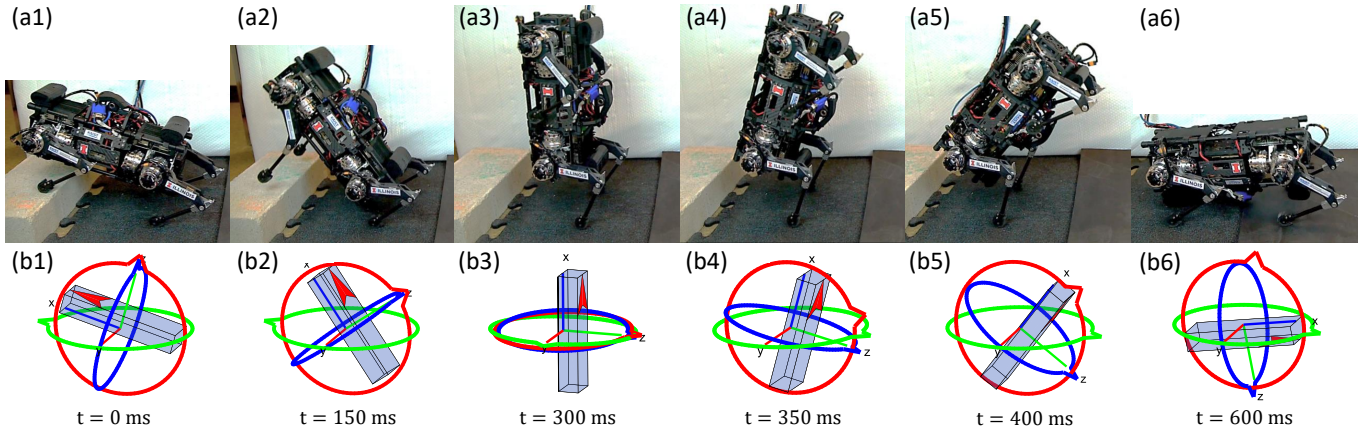

Fig. 16: Quadruped robot *Panther* performing a controlled backflip that passes through singularity pose. (a) An image sequence of the robot executing the controlled backflip, with its front legs launching from a 80 mm high platform. (b) The body orientation reconstructed from the experimental data. The rotation matrix is represented by the body coordinate frame axes (x-blue, y-red, z-green); the Euler angles are visualized by the three colored rings with arrow (ϕ-blue, θ-red, ψ-green). The robot passes through the upright pose (a3) while the hind legs are in contact with the ground. The Gimbal lock effect is shown in (b3) where axes of Euler angles are aligned.

TABLE II: Cost function weights for the simulations and experiments. The values in parenthesis represent weights on the terminal costs.

	Sim. Pose	Sim. TrotWalk	Sim. Bound	Acro. Mnvr.	Exp. Pose/ Balance	Exp. TrotWalk	Exp. TrotRun	Exp. Bound	Exp. Backflip
Q_{p_x}	3e5 (1e5)	1e5	8e4	5e6	3e5 (1e5)	1e5	1e5	2e5 (1.2e5)	1e5
Q_{p_y}	5e5 (1e5)	2e5	5e4	5e6	5e5 (1e5)	1e5 (1.5e5)	1e5 (1.5e5)	4e5	1e5 (2e5)
Q_{p_z}	2e5 (1e5)	3e5	3e6	5e6	2e5 (1e5)	1.5e5 (2.2e5)	2e4	1.5e5 (2e5)	1.5e5 (2.2e5)
$Q_{\dot{p}_x}$	10 (30)	5e2	4e3(5e2)	5e3	10 (30)	1e3 (1.5e3)	1e3 (1.5e3)	50	1e3 (1.5e3)
$Q_{\dot{p}_y}$	8 (30)	1e3	5e2	5e3	8 (30)	1e3	1e3	200 (150)	1e3
$Q_{\dot{p}_z}$	10 (30)	1e3	7e2(5e2)	5e3	10 (30)	150	100	30	150
Q_{R_x}	5e2	1e3	8e3	1e6	5e2	2e3	1e3 (2e3)	3e3 (1e3)	4e3 (6e3)
Q_{R_y}	2e3 (3e3)	1e4	5e5 (5e4)	1e6	2e3 (3e3)	2e3	2e3	4e3 (8e3)	0 (10)
Q_{R_z}	1e3	8e2	8e3	1e6	1e3	8e2	8e2	1e3 (3e3)	8e2
Q_{ω_x}	2	40	2e2	5e3	2	60 (100)	60 (100)	3 (2)	60 (100)
Q_{ω_y}	4	40	1e2	5e3	4	40 (45)	40 (45)	6 (2)	0 (1)
Q_{ω_z}	3	10	2e2	5e3	3	10	10	5 (8)	10
R_{u_x}	0.1	0.1	0.2	0.1	0.1	0.1	0.1	0.1	0.1
R_{u_y}	0.1	0.2	0.2	0.1	0.1	0.18	0.18	0.18	0.12
R_{u_z}	0.1	0.1	0.2	0.1	0.1	0.08	0.08	0.2	0.1
T_{st}	N/A	0.3	0.1	0.1 (0.2)	N/A	0.3	0.12 / 0.2	0.1	0.12 (0.3)*
T_{sw}	N/A	0.15	0.16	N/A	N/A	0.15	0.2 / 0.1	0.2	0.3
N_{hor}	7	6	6	7	6	6	6	7	6
γ	1.0	1.0	0.9	0.9	1.0	1.0	1.0	0.8	0.8
T_{pred}	0.05	0.08	0.01	0.01	0.02	0.08	0.05	0.01	0.02
f_{MPC}	100	100	100	100	250	250	250	160	200

Note: T_{st}, T_{sw} and T_{pred} all have the unit of [s]; N_{hor} is the MPC prediction horizon; T_{pred} is the prediction time step; f_{MPC} is the MPC control frequency with the unit of [Hz].
*0.13 s is the front stance time, and 0.3 s is the hind stance time.

In the simulation case study presented in Section III-E we found out that in the RF-MPC framework, linearizing around the operating point provides a more robust control strategy compared with linearizing around the reference trajectory. Experiments including pose/balance control, walking/running trot and bounding were conducted on the robot. In addition, the controlled backflip experiment demonstrated that RF-MPC controller can stabilize dynamic motions that involve the singularity. By utilizing a custom QP solver qpSWIFT, the MPC could reach control frequency as high as 250 Hz.

This novel RF-MPC framework is likely to open up possibilities for quadruped robots and legged robots in general to realize extremely dynamic 3D motions. We also envision to equip the robot with special end-effectors ($e.g.$, climbing robot with claws [70] or magnetic grippers), enabling it to climb up vertical surfaces and walk on the ceiling. Moreover, with the emergence of powerful and light-weight computing units, the variation-based formulation could potentially be applied to stabilizing acrobatic maneuvers in UAVs.

ACKNOWLEDGMENT

The authors would like to thank Prof. Patrick Wensing for the insightful discussion, Prof. João Ramos for his advice and support, and Jaejun Park for his help in hardware assembly.

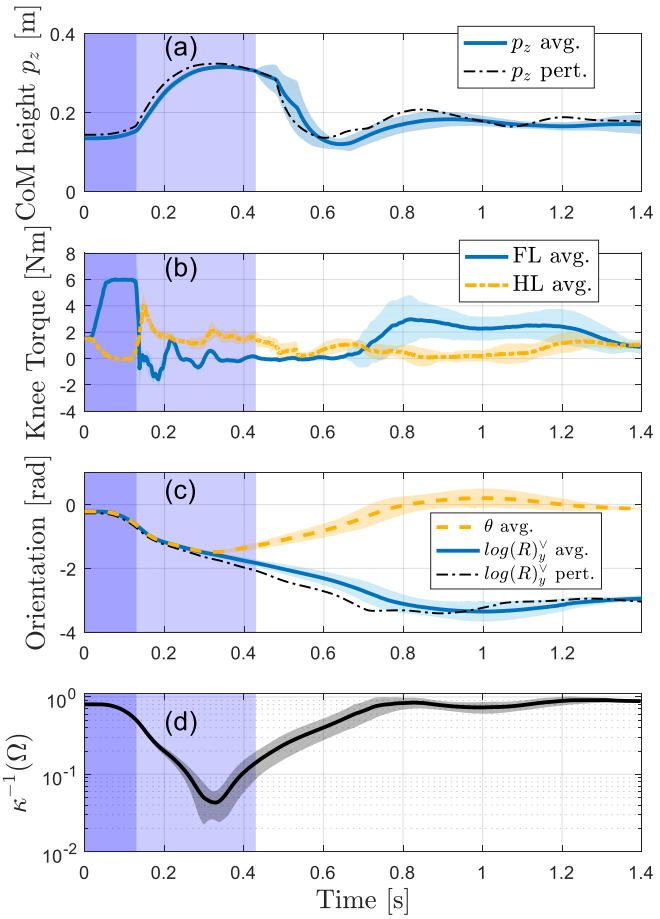

Fig. 17: Experimental data from 10 controlled backflip trials. The solid lines are the average (avg.) of all the tests, and the shaded tube is the range within one standard deviation. The black dash-dot curves in (a) and (c) are from the case where the initial condition of the backflip is perturbed (pert.). The deep-shaded area is when four legs are in contact; the light-shaded area is when hind legs are in contact, and the non-shaded area is when all legs are in the impedance control phase. (a) The CoM height. (b) The knee torque of legs FL and HL. (c) Comparison between the pitch angle θ and rotation matrix $log(R)_y^\vee$. (d) The function $\kappa^{-1}(\mathcal{T}_\Theta)$ indicates that the robot indeed encountered the singular pose in the controlled backflip experiment.

REFERENCES

[1] Y. Ding, A. Pandala, and H.-W. Park, "Real-time model predictive control for versatile dynamic motions in quadrupedal robots," in *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, 2019, pp. 8484–8490.

[2] L. Aronson, "The ibex—king of the cliffs," *Massada, Tel Aviv,(in Hebrew)*, 1982.

[3] Alex & jumpy - the parkour dog. YouTube:. [Online]. Available: https://www.youtube.com/watch?v=39oGCTAJ9Vw&t=117s

[4] A. De and D. E. Koditschek, "Vertical hopper compositions for preflexive and feedback-stabilized quadrupedal bounding, pacing, pronking, and trotting," *The International Journal of Robotics Research*, vol. 37, no. 7, pp. 743–778, 2018.

[5] M. Hutter, C. Gehring, D. Jud, A. Lauber, C. D. Bellicoso, V. Tsounis, J. Hwangbo, K. Bodie, P. Fankhauser, M. Bloesch *et al.*, "Anymal-a highly mobile and dynamic quadrupedal robot," in *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2016, pp. 38–44.

[6] C. Semini, N. G. Tsagarakis, E. Guglielmino, M. Focchi, F. Cannella, and D. G. Caldwell, "Design of HyQ–a hydraulically and electrically actuated quadruped robot," *Proceedings of the Institution of Mechanical Engineers, Part I: Journal of Systems and Control Engineering*, vol. 225, no. 6, pp. 831–849, 2011.

[7] S. Seok, A. Wang, M. Y. Chuah, D. Otten, J. Lang, and S. Kim, "Design principles for highly efficient quadrupeds and implementation on the MIT cheetah robot," in *2013 IEEE International Conference on Robotics and Automation*. IEEE, 2013, pp. 3307–3312.

[8] H.-W. Park, P. M. Wensing, and S. Kim, "High-speed bounding with the MIT cheetah 2: Control design and experiments," *The International Journal of Robotics Research*, vol. 36, no. 2, pp. 167–192, 2017.

[9] G. Bledt, M. J. Powell, B. Katz, J. Di Carlo, P. M. Wensing, and S. Kim, "MIT cheetah 3: Design and control of a robust, dynamic quadruped robot," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2018, pp. 2245–2252.

[10] P. Fankhauser, M. Bjelonic, C. Dario Bellicoso, T. Miki, and M. Hutter, "Robust rough-terrain locomotion with a quadrupedal robot," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, 2018, pp. 5761–5768.

[11] H.-W. Park, P. M. Wensing, and S. Kim, "Jumping over obstacles with mit cheetah 2," *Robotics and Autonomous Systems*, p. 103703, 2020.

[12] Q. Nguyen, M. J. Powell, B. Katz, J. D. Carlo, and S. Kim, "Optimized jumping on the mit cheetah 3 robot," in *2019 International Conference on Robotics and Automation (ICRA)*, 2019, pp. 7448–7454.

[13] B. Katz, J. Di Carlo, and S. Kim, "Mini cheetah: A platform for pushing the limits of dynamic quadruped control," in *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, 2019, pp. 6295–6301.

[14] M. H. Raibert, *Legged robots that balance*. MIT press, 1986.

[15] L. Righetti, J. Buchli, M. Mistry, M. Kalakrishnan, and S. Schaal, "Optimal distribution of contact forces with inverse-dynamics control," *The International Journal of Robotics Research*, vol. 32, no. 3, pp. 280–298, 2013.

[16] M. Hutter, C. Gehring, M. Bloesch, C. D. Remy, and R. Siegwart, "Hybrid operational space control for compliant legged systems," *Robotics*, p. 129, 2013.

[17] X. Xiong and A. Ames, "Sequential motion planning for bipedal somersault via flywheel slip and momentum transmission with task space control," *arXiv preprint arXiv:2008.02432*, 2020.

[18] J. Hwangbo, J. Lee, A. Dosovitskiy, D. Bellicoso, V. Tsounis, V. Koltun, and M. Hutter, "Learning agile and dynamic motor skills for legged robots," *Science Robotics*, vol. 4, no. 26, 2019.

[19] A. Herdt, H. Diedam, P.-B. Wieber, D. Dimitrov, K. Mombaur, and M. Diehl, "Online walking motion generation with automatic footstep placement," *Advanced Robotics*, vol. 24, no. 5-6, pp. 719–737, 2010.

[20] J. Koenemann, A. Del Prete, Y. Tassa, E. Todorov, O. Stasse, M. Bennewitz, and N. Mansard, "Whole-body model-predictive control applied to the HRP-2 humanoid," in *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2015, pp. 3346–3351.

[21] M. Neunert, M. Stäuble, M. Giftthaler, C. D. Bellicoso, J. Carius, C. Gehring, M. Hutter, and J. Buchli, "Whole-body nonlinear model predictive control through contacts for quadrupeds," *IEEE Robotics and Automation Letters*, vol. 3, no. 3, pp. 1458–1465, 2018.

[22] R. Grandia, F. Farshidian, A. Dosovitskiy, R. Ranftl, and M. Hutter, "Frequency-aware model predictive control," *IEEE Robotics and Automation Letters*, vol. 4, no. 2, pp. 1517–1524, 2019.

[23] R. Grandia, F. Farshidian, R. Ranftl, and M. Hutter, "Feedback mpc for torque-controlled legged robots," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2019, pp. 4730–4737.

[24] C. Mastalli, I. Havoutis, M. Focchi, D. G. Caldwell, and C. Semini, "Motion planning for quadrupedal locomotion: coupled planning, terrain mapping and whole-body control," *IEEE Transactions on Robotics*, 2020.

[25] S. Fahmi, C. Mastalli, M. Focchi, and C. Semini, "Passive whole-body control for quadruped robots: Experimental validation over challenging terrain," *IEEE Robotics and Automation Letters*, vol. 4, no. 3, pp. 2553–2560, 2019.

[26] M. Focchi, R. Orsolino, M. Camurri, V. Barasuol, C. Mastalli, D. G. Caldwell, and C. Semini, *Heuristic Planning for Rough Terrain Locomotion in Presence of External Disturbances and Variable Perception Quality*. Cham: Springer International Publishing, 2020, pp. 165–209.

[27] M. D. Shuster, "A survey of attitude representations," *Navigation*, vol. 8, no. 9, pp. 439–517, 1993.

[28] B. Siciliano and O. Khatib, *Springer handbook of robotics*. Springer, 2016.

[29] S. P. Bhat and D. S. Bernstein, "A topological obstruction to global asymptotic stabilization of rotational motion and the unwinding phe-

nomenon," in *American Control Conference, 1998. Proceedings of the 1998*, vol. 5. IEEE, 1998, pp. 2785–2789.
[30] X. Yang, Y. Chen, L. Chang, A. A. Calderón, and N. O. Pérez-Arancibia, "Bee+: A 95-mg four-winged insect-scale flying robot driven by twinned unimorph actuators," *IEEE Robotics and Automation Letters*, vol. 4, no. 4, pp. 4270–4277, 2019.
[31] C. G. Mayhew, R. G. Sanfelice, and A. R. Teel, "On quaternion-based attitude control and the unwinding phenomenon," in *Proceedings of the 2011 American Control Conference*, 2011, pp. 299–304.
[32] F. Bullo and A. D. Lewis, *Geometric control of mechanical systems: modeling, analysis, and design for simple mechanical control systems*. Springer Science & Business Media, 2004, vol. 49.
[33] G. Wu and K. Sreenath, "Variation-based linearization of nonlinear systems evolving on SO(3)and S2,," *IEEE Access*, vol. 3, pp. 1592–1604, 2015.
[34] T. Lee, M. Leok, and N. H. McClamroch, "Stable manifolds of saddle equilibria for pendulum dynamics on S2 and SO(3)," in *Decision and Control and European Control Conference (CDC-ECC), 2011 50th IEEE Conference on Decision and Control and European Control Conference (CDC-ECC)*. IEEE, 2011, pp. 3915–3921.
[35] E. S. Meadows, M. A. Henson, J. W. Eaton, and J. B. Rawlings, "Receding horizon control and discontinuous state feedback stabilization," *International Journal of Control*, vol. 62, no. 5, pp. 1217–1229, 1995.
[36] H. G. Bock and K.-J. Plitt, "A multiple shooting algorithm for direct solution of optimal control problems," *IFAC Proceedings Volumes*, vol. 17, no. 2, pp. 1603–1608, 1984.
[37] O. von Stryk, *Numerical Solution of Optimal Control Problems by Direct Collocation*. Basel: Birkhäuser Basel, 1993, pp. 129–143.
[38] R. J. Full and D. E. Koditschek, "Templates and anchors: neuromechanical hypotheses of legged locomotion on land," *Journal of experimental biology*, vol. 202, no. 23, pp. 3325–3332, 1999.
[39] S. Kajita, "Study of dynamic biped locomotion on rugged terrain-derivation and application of the linear inverted pendulum mode," in *Proc. IEEE Int. Conf. on Robotics and Automation, Sacramento, CA, 1991*, 1991, pp. 1405–1411.
[40] J. Ramos and S. Kim, "Humanoid dynamic synchronization through whole-body bilateral feedback teleoperation," *IEEE Transactions on Robotics*, vol. 34, no. 4, pp. 953–965, 2018.
[41] X. Xiong and A. D. Ames, "Orbit characterization, stabilization and composition on 3d underactuated bipedal walking via hybrid passive linear inverted pendulum model," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2019, pp. 4644–4651.
[42] D. E. Orin, A. Goswami, and S. H. Lee, "Centroidal dynamics of a humanoid robot," *Autonomous Robots*, vol. 35, no. 2-3, pp. 161–176, 2013.
[43] H. Dai, A. Valenzuela, and R. Tedrake, "Whole-body motion planning with simple dynamics and full kinematics," in *Proceedings of the IEEE-RAS international conference on humanoid robots*, 2014.
[44] P. M. Wensing and D. E. Orin, "Generation of dynamic humanoid behaviors through task-space control with conic optimization," in *2013 IEEE International Conference on Robotics and Automation*. IEEE, 2013, pp. 3103–3109.
[45] C. Li, Y. Ding, and H.-W. Park, "Centroidal-momentum-based trajectory generation for legged locomotion," *Mechatronics*, vol. 68, p. 102364, 2020.
[46] J. E. Marsden and T. S. Ratiu, "Introduction to mechanics and symmetry," *Physics Today*, vol. 48, no. 12, p. 65, 1995.
[47] M. Chignoli and P. M. Wensing, "Variational-based optimal control of underactuated balancing for dynamic quadrupeds," *IEEE Access*, vol. 8, pp. 49 785–49 797, 2020.
[48] T. Lee, M. Leoky, and N. H. McClamroch, "Geometric tracking control of a quadrotor uav on SE(3)," in *Decision and Control (CDC), 2010 49th IEEE Conference on*. IEEE, 2010, pp. 5420–5425.
[49] R. Featherstone, *Rigid body dynamics algorithms*. Springer, 2014.
[50] A. Graham, *Kronecker products and matrix calculus with applications*. Courier Dover Publications, 2018.
[51] J. C. Trinkle, J.-S. Pang, S. Sudarsky, and G. Lo, "On dynamic multi-rigid-body contact problems with coulomb friction," *ZAMM-Journal of Applied Mathematics and Mechanics/Zeitschrift für Angewandte Mathematik und Mechanik*, vol. 77, no. 4, pp. 267–279, 1997.
[52] Y. Wang and S. Boyd, "Fast model predictive control using online optimization," *IEEE Transactions on Control Systems Technology*, vol. 18, no. 2, pp. 267–278, March 2010.
[53] S. Boyd and L. Vandenberghe, *Convex optimization*. Cambridge university press, 2004.
[54] J. J. Craig, *Introduction to robotics: mechanics and control, 3/E*. Pearson Education India, 2009.
[55] J. Di Carlo, P. M. Wensing, B. Katz, G. Bledt, and S. Kim, "Dynamic locomotion in the MIT Cheetah 3 through convex model-predictive control," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2018, pp. 1–9.
[56] MIT-Biomimetics-Robotics-Lab, "Cheetah-software," https://github.com/charlespwd/project-title[Accessed 29 June 2020], 2019.
[57] M. Kelly, "An introduction to trajectory optimization: How to do your own direct collocation," *SIAM Review*, vol. 59, no. 4, pp. 849–904, 2017.
[58] A. G. Pandala, Y. Ding, and H.-W. Park, "qpSWIFT: A real-time sparse quadratic program solver for robotic applications," *IEEE Robotics and Automation Letters*, vol. 4, no. 4, pp. 3355–3362, 2019.
[59] Y. Ding and H.-W. Park, "Design and experimental implementation of a quasi-direct-drive leg for optimized jumping," in *Intelligent Robots and Systems (IROS), 2017 IEEE/RSJ International Conference*. IEEE, 2017, pp. 300–305.
[60] Y. Ding, C. Li, and H.-W. Park, "Single leg dynamic motion planning with mixed-integer convex optimization," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2018, pp. 1–6.
[61] ——, "Kinodynamic motion planning for multi-legged robot jumping via mixed-integer convex program," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*.
[62] J. Pratt, J. Carff, S. Drakunov, and A. Goswami, "Capture point: A step toward humanoid push recovery," in *Humanoid Robots, 2006 6th IEEE-RAS International Conference on*. IEEE, 2006, pp. 200–207.
[63] R. E. Kalman, "A new approach to linear filtering and prediction problems," *Transactions of the ASME–Journal of Basic Engineering*, vol. 82, no. Series D, pp. 35–45, 1960.
[64] S. Osder, W. Rouse, and L. Young, "Navigation, guidance, and control systems for V/STOL aircraft." *Sperry Tech*, vol. 1, no. 3, 1973.
[65] P. Corke, "An inertial and visual sensing system for a small autonomous helicopter," *Journal of robotic systems*, vol. 21, no. 2, pp. 43–51, 2004.
[66] S. Saripalli, J. M. Roberts, P. Corke, G. Buskey, and G. Sukhatme, "A tale of two helicopters," in *Proceedings 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems, 2003 (IROS 2003)*, vol. 1. IEEE, 2003, pp. 805–810.
[67] P. M. Wensing, A. Wang, S. Seok, D. Otten, J. Lang, and S. Kim, "Proprioceptive actuator design in the MIT Cheetah: Impact mitigation and high-bandwidth physical interaction for dynamic legged robots," *IEEE Transactions on Robotics*, vol. 33, no. 3, pp. 509–522, 2017.
[68] A. De Luca and R. Mattone, "Sensorless robot collision detection and hybrid force/motion control," in *Proceedings of the 2005 IEEE international conference on robotics and automation*. IEEE, 2005, pp. 999–1004.
[69] M. R. Jardin and E. R. Mueller, "Optimized measurements of unmanned-air-vehicle mass moment of inertia with a bifilar pendulum," *Journal of Aircraft*, vol. 46, no. 3, pp. 763–775, 2009.
[70] J. Park, D. H. Kong, and H. Park, "Design of anti-skid foot with passive slip detection mechanism for conditional utilization of heterogeneous foot pads," *IEEE Robotics and Automation Letters*, vol. 4, no. 2, pp. 1170–1177, 2019.

Yanran Ding (M'16) received his B.S. degree in Mechanical Engineering from Shanghai Jiao Tong University (SJTU), Shanghai, China, in 2015 and the M.S. degree from the Mechanical Science and Engineering Department, University of Illinois at Urbana-Champaign (UIUC), Champaign, in 2017. He is currently pursuing his Ph.D. degree at the Dynamic Robotics Laboratory in UIUC. His research interests include the design of agile robotic systems and optimization-based control for legged robots to achieve dynamic motions. He is one of the best student paper finalists in the International Conference on Intelligent Robots and Systems (IROS) 2017.

Abhishek Pandala received his Master of Science in Mechanical Engineering from the University of Illinois at Urbana-Champaign (UIUC) in 2019 and a Dual Degree (B.Tech and M.Tech) in Mechanical Engineering from the Indian Institute of Technology Madras (IIT-M) in 2017. He is currently pursuing his Ph.D. degree in Mechanical Engineering at the Virginia Polytechnic Institute and State University. His research interests include optimization-based control of dynamical systems with application to high degree of freedom robots.

Chuanzheng Li received his B.S. degree in Mechatronics from Zhejiang University, Hangzhou, China in 2014, and the M.S. degree from the Mechanical Science and Engineering Department, University of Illinois at Urbana-Champaign, Champaign, IL, USA in 2017. He is currently in the Ph.D. program at University of Illinois at Urbana-Champaign supervised by Dr. Hae-Won Park, working primarily on the design of mechatronic systems and the real-time control of legged robots.

Young-Ha Shin received his BS degrees in the Department of Mechanical Engineering from Korea Advanced Institute of Science and Technology (KAIST), Daejeon, South Korea in 2019. He is currently a graduate student in the MS course in the Department of Mechanical Engineering in Korea Advanced Institute of Science and Technology (KAIST), Daejeon, South Korea. His research interests include actuator design, model predictive control for quadruped robots.

Hae-Won Park is an Assistant Professor of Mechanical Engineering at the Korea Advanced Institute of Science and Technology (KAIST). He received B.S. and M.S. degrees from Yonsei University, Seoul, Korea, in 2005 and 2007, respectively, and the Ph.D. degree from the University of Michigan, in 2012, all in mechanical engineering. His research interests lie at the intersection of control, dynamics, and mechanical design of robotic systems, with special emphasis on legged locomotion robots. He is the recipient of the 2018 National Science Foundation (NSF) CAREER Award, NSF most prestigious awards in support of early-career faculty.