2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
October 14-18, 2024. Abu Dhabi, UAE

A Fast Online Omnidirectional Quadrupedal Jumping Framework Via Virtual-Model Control and Minimum Jerk Trajectory Generation

Linzhu Yue[1], Lingwei Zhang[1], Zhitao Song[1], Hongbo Zhang[1],
Jinhu Dong[1], Xuanqi Zeng[1], and Yun-Hui Liu[1]

Abstract— Exploring the limits of quadruped robot agility, particularly in the context of rapid and real-time planning and execution of omnidirectional jump trajectories, presents significant challenges due to the complex dynamics involved, especially when considering significant impulse contacts. This paper introduces a new framework to enable fast, omnidirectional jumping capabilities for quadruped robots. Utilizing minimum jerk technology, the proposed framework efficiently generates jump trajectories that exploit its analytical solutions, ensuring numerical stability and dynamic compatibility with minimal computational resources. The virtual model control is employed to formulate a Quadratic Programming (QP) optimization problem to accurately track the Center of Mass (CoM) trajectories during the jump phase. The whole-body control strategies facilitate precise and compliant landing motion. Moreover, the different jumping phase is triggered by time-schedule. The framework's efficacy is demonstrated through its implementation on an enhanced version of the open-source Mini Cheetah robot. Omnidirectional jumps-including forward, backward, and other directional-were successfully executed, showcasing the robot's capability to perform rapid and consecutive jumps with an average trajectory generation and tracking solution time of merely 50 microseconds.

I. INTRODUCTION

Robotics jumping is critical for quadruped robots to traverse between complicated unstructured terrains. Although quasi-static jumping algorithms empower quadrupedal robots to cross mild irregular terrain, when aimed at avoiding unknown obstacles, it is necessary for robots to perform omnidirectional jumping promptly, which requires fast, skillful planning and dynamical executing motions. We strive to propose a framework with minimal computational resources that enables quadrupedal robots to make rapid omnidirectional jump responses to avoid sudden dangers while exploring unknown environments. This could be significantly helpful in inspection scenarios where quadrupedal robots are equipped with large, heavy instruments and the onboard computer has other heavy-load tasks. The real-time omnidirectional jumping framework could improve the robot's ability to traverse rough terrains and ensure the safety of robots.

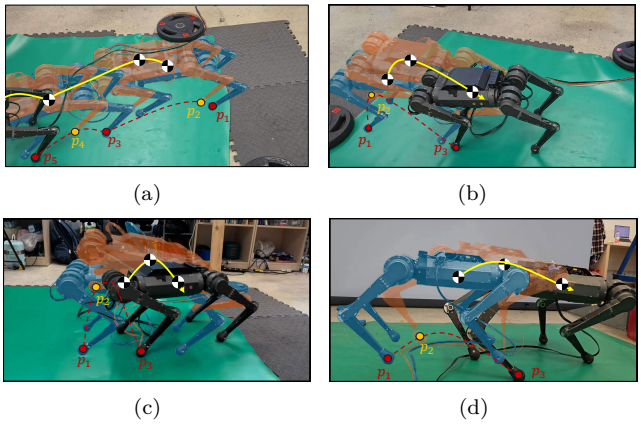


Fig. 1. Various jumping motion experiments to validate the proposed omnidirectional jumping framework. (a) Front-left jumping consecutively. (b) Single rear-right jumping. (c) Single rear jumping. (d) Single front jumping. The yellow line shows the trajectory of CoM, and the red dashed line represents the trajectory of the selected foot. The red dots indicate the foot's contact with the ground, and the orange dots indicate the foot in the flight phase.

Various experimental conductions demonstrate the great potential of optimal control to provoke robots to achieve robust jumping motions. Robots such as ANYmal [1], SALTO-1P [2] and Mini Cheetah [3] can perform remarkable jumping motions and navigating in the local environment. A traditional method simplifies the robot model as a 2-D planar model by combining left and right legs to accelerate reference trajectory optimization speed. Model Predictive Control (MPC) [4], Whole-Body Impulse Control (WBIC) [5], and Virtual Model Control (VMC) [6] are implemented to track the reference trajectory. The MIT Cheetah 2 is able to autonomously jump over obstacles up to 40 cm in height during bounding gait through Nonlinear MPC [7]. Except for single rigid body models, with kino-dynamic and novel torque-speed limitation constraints, the Mini Cheetah can reliably produce successful aerial motions such as flips and barrel rolls [8]. The cost on the order of seconds rather than milliseconds confines the performance of the kino-dynamic planners to generate a motion plan, which implies the real-time limitation of running jumping that starts from the non-static initial position. Though our previous work [9], [10] based on offline Differential

[1] L. Z. Yue, L. W. Zhang, Z. T. Song, H. B. Zhang, J. H. Dong, X. Q. Zeng, and Y.-H. Liu is with the Department of Mechanical and Automation Engineering at the Chinese University of Hong Kong. lzyue@mae.cuhk.edu.hk
* Corresponding author: Y.-H. Liu yhliu@cuhk.edu.hk
This work is supported by the InnoHK of the Government of Hong Kong via the Hong Kong Centre for Logistics Robotics, the CUHK T Stone Robotics Institute, and the Shenzhen Portion of Shenzhen-Hong Kong Science and Technology Innovation Cooperation Zone under HZQB-KCZYB-20200089.

Evolution (DE) and online evolutionary algorithms with pre-motion library also endow quadrupedal robots with the ability to execute complicated jumping motions, the trajectory optimization result is not reliable in some extreme cases.

Reinforcement Learning (RL) is another broad way to achieve accurate and aggressive quadrupedal jumping motions. In [11], with policies considering the robot's total power limits and torque-speed relationships, Unitree A1 [12] achieves aggressive and accurate jumping motions. Cat-like jumping exploits the possibility of adjusting robot body gestures by jumping motions in a low gravity environment [13]. However, few researchers attach great importance to using a single policy to make robots perform complicated or omnidirectional jumping motions.

For real-time omnidirectional jumping frameworks controlled by a microcontroller unit, the robot Moobot [14] is capable of traversing different platforms from all directions. However, the insect-scale robot cannot jump consecutively and must be reloaded by hand after each jump. As for quadruped robots, a hierarchical planning and control framework [15] is proposed to enable Mini Cheetah to traverse complex multi-layered terrain. A novel high-level jump selection controller is implemented to ensure the most robustness guarantees. In our work, enabling quadruped robots to perform omnidirectional jumping consecutively, we proposed a framework that minimizes the computing time of the trajectory planner and the reference tracking controller to improve the agility of quadruped robots significantly. Due to analytic solutions, the minimum-jerk trajectory planner efficiently generates a CoM reference path in the jumping phase with numerical stability and dynamic compatibility. Aiming to track reference motions, an intuitive virtual model controller is deployed to compute the force of feet with low computational resources. Then, in the landing phase, a whole-body controller guarantees the CoM inside the support polygon of four feet, ensuring the stability and compatibility of the robot's body.

This paper makes the following contributions:

- We propose an omnidirectional jumping framework that generates and tracks aerial motions for quadrupedal robots, made up of a minimum jerk CoM trajectory planner, a VMC reference tracking controller, and a WBC landing controller.
- The average cost of generating trajectory and computing tracking motor torque commands to perform omnidirectional jumping motion is within 50 us, implying the remarkable real-time performance of the framework.
- The efficacy of the framework is verified on open-source Mini Cheetah [24], and the robot succeeds in generating multiple reference CoM trajectories and performing omnidirectional jumping behaviors consecutively (see Fig. 1)

The remaining content of this paper is structured as follows. Section II briefly introduces the models of robot and omnidirectional jumping. Section III details the formulation of the minimum jerk trajectory planner and the VMC controller, and section IV is the implementation details and experiments, including the hardware setup, real-time performance, and, verification of the omnidirectional jumping.

II. Models and dynamics

A. Robot Model

The reduced-order dynamic model of jumping motion treats the robot as a single rigid body (SRB) with a specified moment of inertia. The robot state $\boldsymbol{x}$ can be written as:

$$\boldsymbol{x} := [\boldsymbol{P}_{com}^T \quad \boldsymbol{\Theta}^T \quad \boldsymbol{V}_C^T \quad \boldsymbol{\omega}_B^T]^T \in \mathbb{R}^{12} \tag{1a}$$

$$\boldsymbol{Q} := [\boldsymbol{q}_i \quad \dot{\boldsymbol{q}}_i] \in \mathbb{R}^{24} \tag{1b}$$

Where $\boldsymbol{P}_{com} \in \mathbb{R}^3$ is the position of the robot's body center of mass (CoM) with respect to (w.r.t.) inertial frame (see Fig.2); $\boldsymbol{\Theta} = [\psi \ \phi \ \theta]$ represents the Euler angles of the robot's body; $\boldsymbol{V}_C \in \mathbb{R}^3$ is the velocity of the CoM. $\boldsymbol{\omega}_B \in \mathbb{R}^3$ is the angular velocity of CoM w.r.t. the robot frame $\{B\}$. $\boldsymbol{q}_i \in \mathbb{R}^3$ and $\dot{\boldsymbol{q}}_i \in \mathbb{R}^3$ are the joint angles and velocities of each leg. i is the number of feet. The ground reaction force (GRF) $\boldsymbol{f}_i \in \mathbb{R}^3$ at each contact point consists of the dynamic system control input $\boldsymbol{u} := [\boldsymbol{f}_i]$. $\boldsymbol{r}_i$ is the vector from CoM to the robot foot. Then, The linear acceleration of CoM and the angular acceleration of the base are shown:

$$m\ddot{\boldsymbol{P}}_{com} = \sum_{i=1}^{n_c} \boldsymbol{f}_i - \boldsymbol{g} \tag{2a}$$

$$\frac{\mathrm{d}}{\mathrm{d}t}(\boldsymbol{I}_B\boldsymbol{\omega}_B) = \sum_{1}^{n_c} \boldsymbol{r}_i \times \boldsymbol{f}_i + \boldsymbol{\omega}_B \times \boldsymbol{I}_B\boldsymbol{\omega}_B, \tag{2b}$$

where $\boldsymbol{g} \in \mathbb{R}^3$ represents gravitational acceleration. n_c represents the number of contacts. ${}^B\boldsymbol{I} \in \mathbb{R}^{3\times3}$ is the robot's rotation inertial tensor, which is assumed as a constant in this work, $\mathrm{diag}({}^B\boldsymbol{I}) = [0.07, 0.26, 0.242]^T$.

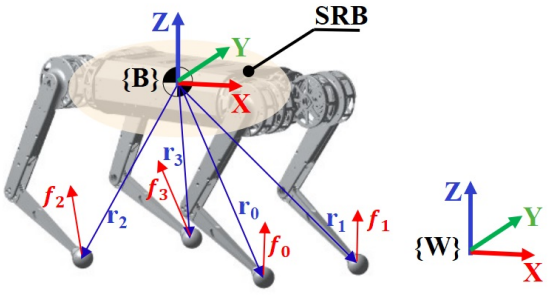


Fig. 2. A model of a single rigid body (SRB) utilized in the framework for VMC. The blue arrow represents the CoM to the plantar position vector, while the red arrow represents the Ground Reaction Forces (GRFs).

B. Omnidirectional Jumping Model

Omnidirectional jumping significantly increases quadruped robots' capability to access terrains instead of traditional assumptions that robots jump in 2D. The motion of omnidirectional jumping comprises three phases: preparing, flight, and landing (see Fig. 3). In the preparing phase, given desired $[\boldsymbol{P}_{CoM}\ \dot{\boldsymbol{P}}_{CoM}\ \ddot{\boldsymbol{P}}_{CoM}]$, mini jerk planner and motor torques generate a smooth CoM trajectory are calculated the motor torque commands τ_{cmd} to track the reference path. In the flight phase, a proportional and derivative controller ensures the preparation of the robot landing. In the landing phase, given reference GRFs and desired base angles, a whole-body controller maintains the stability of the robot's base regardless of the impact of the ground on the robot's legs.

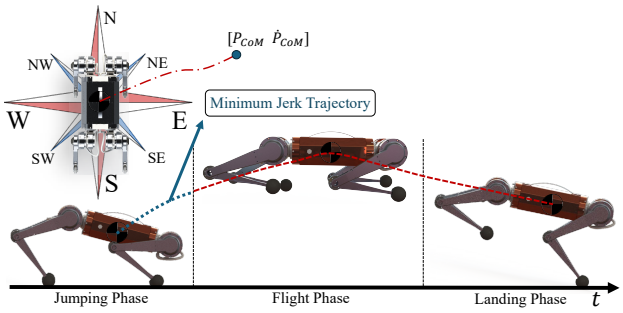


Fig. 3. The Jumping phase, Flight phase, and Landing phase consist of the omnidirectional jumping motion.

III. Framework

This section introduces the online omnidirectional jumping framework with low computational requirements, comprising three primary components: a minimum jerk trajectory generator, a virtual model controller, and a whole-body landing controller. With an analytic solution of relative coefficients, the five-order polynomial trajectory generator rapidly generates a smooth and dynamic feasible path for the CoM of robots based on its current states. Through optimizing output GRFs, the VMC enables the robot to track the CoM reference trajectory using a quadratic program and a Cartesian PD controller. Finally, given reference GRFs and reference base Euler angles, the whole-body landing controller ensures the stability of the robot's landing, and a PD motor torque controller is implemented to improve the robot's performance of tracking references in the landing phase. An overview is shown in Fig 4. The first subsection shows details of the mini-jerk trajectory planner, the second section reveals the process of generating motor torque commands by VMC, and the last subsection briefly introduces the WBC and motor torque control methods in the landing phase.

A. Minimum Jerk Trajectory Planner

The minimum jerk/snap algorithm has been extensively utilized in UAV systems [16]. UAVs possess unique dynamic features that satisfy the criteria of differential flatness, demonstrating that UAV control can be used for trajectory generation. When the UAV system employs mini-jerk trajectory creation, the UAV experiences minimal thrust, while using mini-snap results in the minimum differential thrust. For the quadrupedal robot, considering the SRB model, when all four feet are in contact with the ground, only the force in the Z direction and the acceleration of the CoM are taken into account, similar to trajectory generation in a UAV system. The system with a fixed inclination satisfies the differential flatness property even if the force direction does not align perfectly with the vector leading to the center of mass along the sole of the quadruped robot [17].

Currently, the trajectory produced by the mini-jerk allows for the generation of relatively reasonable GRFs in a short amount of time. (2a) the connection between the CoM acceleration and the contact force helps create a smooth trajectory for the quadruped robot's jumping motion. This trajectory is then sent to the VMC for tracking and optimization to achieve a 12-dimensional contact force that meets kino-dynamics requirements. In this work, reference Euler angles and angular velocities are selected as hyperparameters, and the trajectory outputs are chosen as:

$$s(t) = [x \quad y \quad z] = \boldsymbol{\psi} \tag{3}$$

To make a quadruped robot well track the space the flatness outputs, the polynomial order of the smooth trajectory is selected as five, and it is convenient to formulate it as three-piecewise segments:

$$s(t) = \begin{cases} s_1(t) = \sum_{i=0}^{5} a_{1,i} t^i & T_0 \le t \le T_1 \\ s_2(t) = \sum_{i=0}^{5} a_{2,i} t^i & T_1 \le t \le T_2 \\ s_3(t) = \sum_{i=0}^{5} a_{3,i} t^i & T_2 \le t \le T_3 \end{cases} \tag{4a}$$

$$s_j(t)^{'''} = \sum_{i\ge 3}^{5} i(i-1)(i-2)t^{i-3} a_{j,i} \tag{4b}$$

Where $s(t)$ represents the whole trajectory of CoM, $s_j(t)$ is each segment's trajectory. $a_{j,i}$ is the coefficient of each 5-order polynomial segment path. $s_j(t)^{'''}$ is the jerk of each segment.

Under the precondition of tracking reference trajectory, smaller reference GRF values in (2a) ensure motors' accuracy of torque and speed. Therefore, the optimization program of figuring out a trajectory equipped with the minimal acceleration reference can be formulated as follows:

$$J_j(T) = \int_{T_{j-1}}^{T_j} (s_j(t)^{'''})^2 dt = \boldsymbol{p}_j^T \boldsymbol{Q}_j \boldsymbol{p}_j \tag{5a}$$

$$s.t. \quad s_j^k(T_j) = \boldsymbol{\psi}_{T,j}^k \tag{5b}$$

$$s_j^k(T_j) = s_{j+1}^k(T_j) \tag{5c}$$

where (5b) is the derivative constraint for one polynomial segment, (5c) is the continuity constraint between two

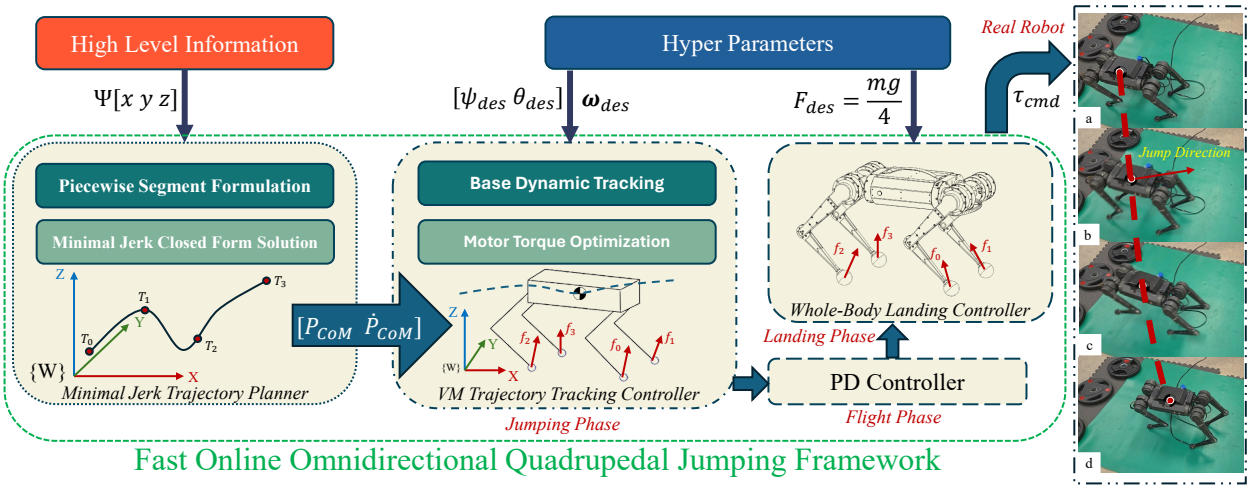


Fig. 4. The overview of the fast online omnidirectional quadrupedal jumping framework, made of 3 primary components.

segments, and k is the derivative order. Then the solution of this problem can be solved in closed form ref to [18]:

$$J = \begin{bmatrix} \boldsymbol{a}_f \\ \boldsymbol{a}_p \end{bmatrix}^T \underbrace{\begin{bmatrix} \boldsymbol{R}_{ff} & \boldsymbol{R}_{fp} \\ \boldsymbol{R}_{pf} & \boldsymbol{R}_{pp} \end{bmatrix}}_{\boldsymbol{R}} \begin{bmatrix} \boldsymbol{a}_f \\ \boldsymbol{a}_p \end{bmatrix} \tag{6a}$$

$$= \boldsymbol{a}_f^T \boldsymbol{R}_{ff} \boldsymbol{a}_f + \boldsymbol{a}_f^T \boldsymbol{R}_{fp} \boldsymbol{d}_p + \boldsymbol{a}_p^T \boldsymbol{R}_{pf} \boldsymbol{a}_f + \boldsymbol{a}_p^T \boldsymbol{R}_{pp} \boldsymbol{a}_p \tag{6b}$$

where $\boldsymbol{a} = \boldsymbol{C}^T \begin{bmatrix} \boldsymbol{a}_f & \boldsymbol{a}_p \end{bmatrix}^T$, $\boldsymbol{C}$ is the selecting matrix that divides $\boldsymbol{a}$ to free $\boldsymbol{a}_f$ and constrained $\boldsymbol{a}_p$. $\boldsymbol{R} = \boldsymbol{C}\boldsymbol{M}^{-T}\boldsymbol{Q}\boldsymbol{M}^{-1}\boldsymbol{C}^T$, $\boldsymbol{M}$ is the decision variable mapping matrix. The derivative of (6b) gives the solution: $\boldsymbol{a}_p^* = -\boldsymbol{R}_{pp}^{-1}\boldsymbol{R}_{fp}^T \boldsymbol{a}_f$.

Given a final position target w.r.t body frame, piecewise trajectories depend on the piecewise time allocation. We assume the robot body's trapezoidal velocity to simplify the time allocation process. Every piece accelerates to max velocity, keeps its velocity, and decreases to $0m/s^2$.

B. Virtual Model Trajectory Tracking Controller

The last subsection generates the robot's base's reference position, velocity, and acceleration. Given the reference acceleration and reference angular acceleration, the GRFs can be solved rapidly with a virtual model controller. The virtual components producing virtual forces do not exist in the real world. Choose spring and damper as the robot's virtual components, and the demonstration is shown as Fig. 4 and the relationship between $\ddot{\boldsymbol{P}}_{com}$, $\dot{\boldsymbol{\omega}}_B$ and the reference trajectory from minimum jerk is:

$$\ddot{\boldsymbol{P}}^r_{com} = \boldsymbol{k}_p^p(\boldsymbol{P}^r_{com} - \boldsymbol{P}_{com}) + \boldsymbol{k}_d^p(\dot{\boldsymbol{P}}^r_{com} - \dot{\boldsymbol{P}}_{com}) \tag{7a}$$

$$\dot{\boldsymbol{\omega}}^r_B = \boldsymbol{k}_p^{\omega}\boldsymbol{e}_R + \boldsymbol{k}_d^{\omega}(\boldsymbol{\omega}^r_B - \boldsymbol{\omega}_B) \tag{7b}$$

where $\ddot{\boldsymbol{P}}^r_{com}$ is the reference acceleration, and $\dot{\boldsymbol{P}}^r_{com}$, $\boldsymbol{P}^r_{com}$ are reference base's velocity and position separately from the trajectory planner. $\boldsymbol{k}_p^p \in \mathbb{R}^{3\times3}$ and $\boldsymbol{k}_d^p \in \mathbb{R}^{3\times3}$ are the proportional and derivative gains matrices. $\boldsymbol{\omega}^r_B$ is a constant, and $\boldsymbol{e}_R$ is the error function for the rotation matrix, given by [19] as $e_R = log(\boldsymbol{R}_r^T \cdot \boldsymbol{R})^v$. $\boldsymbol{k}_p^{\omega}$ and $\boldsymbol{k}_d^{\omega}$ are proportional and derivative gain matrices. The reason for not considering the acceleration generated from the trajectory planner is to get a better trajectory tracking performance while robots prepare to jump. The accelerations produced by the virtual model promote the robot's dynamic efficiency and tracking performance with rarely any additional computational cost.

Rewrite (2a) and (2b) in matrix formulation [20]:

$$\underbrace{\begin{bmatrix} \mathbb{I} & \cdots & \mathbb{I} \\ \boldsymbol{r}_1\times & \cdots & \boldsymbol{r}_4\times \end{bmatrix}}_{\boldsymbol{M}\in\mathbb{R}^{6\times12}} \underbrace{\begin{bmatrix} \boldsymbol{f}_1 \\ \vdots \\ \boldsymbol{f}_4 \end{bmatrix}}_{\boldsymbol{f}\in\mathbb{R}^{12\times1}} = \underbrace{\begin{bmatrix} m\ddot{\boldsymbol{P}}_{com} + \boldsymbol{g} \\ \boldsymbol{I}_B\ \dot{\boldsymbol{\omega}}_B \end{bmatrix}}_{\boldsymbol{N}\in\mathbb{R}^{6\times1}} \tag{8}$$

With known Cartesian and angular accelerations of the base, the constraint of (8) is 6-dimension, but f is 12-dimension. Thus, an optimization program can be formed as follows:

$$\boldsymbol{f} = \arg\min_{f}(\boldsymbol{M}\boldsymbol{f} - \boldsymbol{N})^T\boldsymbol{Q}(\boldsymbol{M}\boldsymbol{f} - \boldsymbol{N}) + (\boldsymbol{J}_c\boldsymbol{f})^T\boldsymbol{R}\boldsymbol{J}_c\boldsymbol{f} \tag{9a}$$

$$s.t. \qquad \boldsymbol{c}_l < \boldsymbol{G}\boldsymbol{f} < \boldsymbol{c}_u \tag{9b}$$

Where (9b) is the friction cone constraint of contact feet, whose matrices are shown as:

$$\boldsymbol{G}_i = \begin{bmatrix} -(\mu_i\boldsymbol{n}_i)^T \\ (\mu_i\boldsymbol{n}_i)^T \\ \boldsymbol{n}_i^T \end{bmatrix}, \boldsymbol{c}_l = \begin{bmatrix} -\infty \\ 0 \\ \boldsymbol{f}_{min_i} \end{bmatrix}, \boldsymbol{c}_u = \begin{bmatrix} 0 \\ \infty \\ \boldsymbol{f}_{max_i} \end{bmatrix} \tag{10}$$

$\mu_i \in \mathbb{R}$ is the friction parameter between the contact point and the ground. $\boldsymbol{n}_i \in \mathbb{R}^3$ is the directional normal vector to the ground. $\boldsymbol{Q} \in \mathbb{R}^{6\times6}$ and $\boldsymbol{R} \in \mathbb{R}^{6\times6}$ is weight matrices and the term $(\boldsymbol{J}_c\boldsymbol{f}^T)\boldsymbol{R}\boldsymbol{J}_c\boldsymbol{f}$ penalizes large motor torque output when robot's base tracking the references. $\boldsymbol{J}_c \in \mathbb{R}^{6\times12}$ is the contact Jacobian Matrix. $\boldsymbol{c}_l \in \mathbb{R}^{20\times1}$ is the lower bound of friction cone constrain and $\boldsymbol{c}_u \in \mathbb{R}^{20\times1}$ is the upper bound. $\boldsymbol{G} \in \mathbb{R}^{20\times12}$ is the mapping matrix.

The 9a is a quadratic optimized problem and can be solved efficiently by QuadProg++ [21], [22]. The motor

torque command is calculated by:

$$\begin{aligned} \boldsymbol{\tau}_{ff} &= -\boldsymbol{S}_f \boldsymbol{J}_c \boldsymbol{f} \\ \boldsymbol{\tau}_{cmd} &= \boldsymbol{\tau}_{ff} + \boldsymbol{k}_{cp}(\boldsymbol{p}_f^r - \boldsymbol{p}_f) + \boldsymbol{k}_{cd}(\dot{\boldsymbol{p}}_f^r - \dot{\boldsymbol{p}}_f) \end{aligned} \tag{11}$$

where $\boldsymbol{\tau}_{ff} \in \mathbb{R}^{12}$ is the motor feed-forward torque vector and $\boldsymbol{S}_f \in \mathbb{R}^{12\times12}$ is the selecting matrix. $\boldsymbol{\tau}_{cmd} \in \mathbb{R}^{12}$ is the motor command torque. $\boldsymbol{k}_{cp} \in \mathbb{R}^{12\times12}$ and $\boldsymbol{k}_{cd} \in \mathbb{R}^{12\times12}$ are the Cartesian gain matrices separately. $\boldsymbol{p}_f^r$ and $\dot{\boldsymbol{p}}_f^r$ are the reference foot position and velocity vectors. $\boldsymbol{p}_f$ and $\dot{\boldsymbol{p}}_f$ are the actual foot position and velocity vectors.

C. WBC Landing Controller

As a quadratic program [23], the whole body controller with a controlling frequency of 500Hz ensures the robot's base stability in the landing phase. When the robot touches the ground, the summation of reference GRFs equals its gravity, and the reference base angles are all zeros. In this way, the robot can land steadily and safely. The whole framework algorithm is shown as algorithm 1. The motor torque commands from the whole-body controller come from as follows:

$$\boldsymbol{\tau}_{cmd} = \boldsymbol{\tau}_{ff} + \boldsymbol{k}_p(\boldsymbol{q}^r - \boldsymbol{q}) + \boldsymbol{k}_d(\dot{\boldsymbol{q}}^r - \dot{\boldsymbol{q}}) \tag{12}$$

where $\boldsymbol{k}_p$ and $\boldsymbol{k}_d$ are the PD gains matrices. $\boldsymbol{q}^r$ and $\dot{\boldsymbol{q}}^r$ are the reference motor position and velocity vectors. $\boldsymbol{q}$ and $\dot{\boldsymbol{q}}$ are the actual motor position and velocity vectors.

Algorithm 1: The Framework Algorithm

input : $\boldsymbol{P}^{\mathrm{e}}_{\mathrm{com}}, \dot{\boldsymbol{P}}^{\mathrm{e}}_{\mathrm{com}}, \ddot{\boldsymbol{P}}^{\mathrm{e}}_{\mathrm{com}}, \boldsymbol{\omega}^{\mathrm{e}}, \psi^{\mathrm{e}}, \phi^{\mathrm{e}}, k$
output: $\boldsymbol{\tau}_{\mathrm{cmd}}$

```
while j < k do
    s_j(t) = Σ_{i=0}^{5} a_{j,i} t^i    T_j ≤ t ≤ T_{j+1};
    s_j(t)''' = Σ_{i≥3}^{5} i(i-1)(i-2) t^{i-3} a_{j,i};
end
a_p^* = -R_pp^{-1} R_fp^T a_f;
P̈_com^r = k_p^p (P_com^r - P_com) + k_d^p (Ṗ_com^r - Ṗ_com);
ω̇_B^r = k_p^ω e_R + k_d^ω (ω_B^r - ω_B);
while n < maximum iterations do
    min_f (Mf - N)^T Q (Mf - N) + (J_c f)^T R J_c f
    if f < Tolerance then
        Terminate
    end
end
τ_cmd = -S_f J_c f
```

IV. Implementation Details and Experiments

This part focuses on the hardware implementation specifics of the online omnidirectional jumping framework and experiments conducted using the improved open-source Mini Cheetah [24]. Specifying hardware setup, we first offer solving time data of a particular omnidirectional leaping framework to illustrate the framework's capacity to function in real time on the robot. We showcase the framework's capacity to facilitate Mini-Cheetah executing omnidirectional jumps, such as front-left and rear-right jumps. Finally, comparing the robot performance between the minimum jerk trajectory planner and minimum snap trajectory planner illustrates the disparities between them.

TABLE I
HYPERPARAMETERS

Parameters	Symbol	Values
Minimum Jerk		
End Position Range	$\boldsymbol{P}_{CoM}[m]$	$[\pm0.15, \pm0.1, \pm0.05]$
End Velocity Range	$\dot{P}_{CoM}[m/s]$	$[\pm0.3, \pm0.16, [1.5, 3.5]]$
End Acceleration Range	$\ddot{\boldsymbol{P}}^z_{CoM}[m/s^2]$	$[\pm20, \pm20, [10, 40]]$
VMC		
Friction Coefficient	μ	0.5
Weights for CoM Wrench	$\boldsymbol{Q}$	$diag(1,1,10,20,10,25)$
Weights for Torque Wrench	$\boldsymbol{R}$	$diag(5,50,2)10^{-5}$
CoM Proportional Gains	$\boldsymbol{k}_p^p$	$diag(1070,1070,1070)$
CoM Derivative Gains	$\boldsymbol{k}_d^p$	$diag(12,12,10)$
Attitude Proportional Gains	$\boldsymbol{k}_p^\omega$	$diag(800,800,800)$
Attitude Derivative Gains	$\boldsymbol{k}_d^\omega$	$diag(20,10,20)$
GRFs Minimum	$\boldsymbol{f}_{min}[N]$	5
GRFs Maximum	$\boldsymbol{f}_{max}[N]$	250
Cartesian Proportional Gains	$\boldsymbol{k}_{cp}$	$diag(0,0,0)$
Cartesian Derivative Gains	$\boldsymbol{k}_{cd}$	$diag(15,15,15)$

A. Hardware Setup and Real-Time Performance

The framework developed with C++ is deployed in an enhanced Mini-Cheetah equipped with an Intel NUC i3-8145U@2.1G Hz. The trajectory planning, VMC, and whole-body controllers run on the NUC in real-time. Motor data is collected at 1000 Hz through a New-design USB to Can board. The Lightweight Communications and Marshalling package (LCM) is used for the asynchronous communication between the simulation, hardware bridge, and data collecting.

Hyper-parameters are shown in Tab. I, including hyperparameters of mini jerk trajectory planner and hyperparameters in VMC.

To verify the real-time performance of the fast omnidirectional jumping framework, with the random input reference end CoM position, velocity, and acceleration w.r.t the inertial frame, we record the time-consuming summation of the minimum jerk trajectory planner and VMC controller, which starts from the reference inputting and comes to an end with the generation of motor torque commands. Several time measurements are repeated on actual robots, and the average consuming time is compared with our previous work, the evolutionary-based jumping framework [10] and the offline jumping framework [9]. As Tab. II demonstrated, the average time resource required for trajectory planner and VMC is around 50 microseconds on NUC, depicting well the real-time efficacy of the framework.

TABLE II
JUMPING ALGORITHM COMPARISON

Algorithm	Average Time(s)
This work	0.00005
Evolution-Based Jumping Framework [10]	0.14
Offline Jumping Framework [9]	65

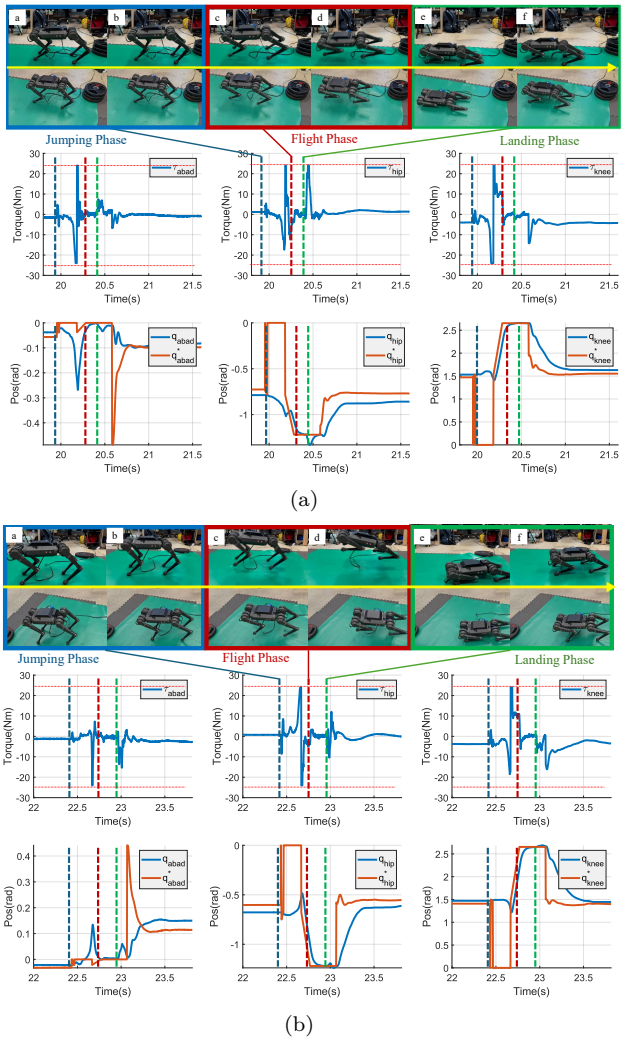
B. Ominidirectional Jumping

Made consisting of the jumping phase, flight phase, and landing phase, the omnidirectional jumping framework is capable of enabling Mini-Cheetah to jump in any direction w.r.t its base frame, such as the front side, the left side, the rear side, and angles between them. Fig. 5 illustrates the robot's ability to perform omnidirectional jumping well. As the figures show, the period between the start of the jumping phase and the beginning of the flight phase is within 0.5 seconds. The maximum of motor torques in the jumping phase is around 22 Nm. Because of the imprecision of the SRB model in dynamics, motor torques generated by the VMC are smaller than the actual desired torques. Thus, a Cartesian PD controller is implemented to improve the performance of foot position and speed tracking in the jumping phase, conducting a better jumping motion. The front-left jumping direction and the rear-right jumping direction are selected intentionally to show the robot's potential capability of performing omnidirectional jumping, which matches the result of the experiments.

V. CONCLUSIONS

In conclusion, this paper introduces a new omnidirectional jumping framework for quadruped robots, leveraging Virtual Model Control (VMC) and Minimum Jerk principles. This framework is distinguished by its rapid execution, achieving microsecond levels with impressive speed. By employing a 5th-order polynomial for trajectory generation alongside defined constraints for continuity, the minimum-jerk trajectory method proves highly effective for this application. Implementing a VMC-based trajectory tracking controller, utilizing PD control for centroid trajectory tracking and GRFs optimization, has shown remarkable efficiency, and in real-world jumping experiments, the solution time for trajectory tracking reached approximately 30 us, with an additional 20 us for trajectory generation. This significantly enhances the optimization time to 50 us, marking a substantial improvement over previous methods that relied on the differential evolution algorithm. Such computing speed is critical for fulfilling the real-time requirements of emergency obstacle avoidance tasks in robotic applications.

Looking ahead, our future work will explore improved time allocation techniques and enhanced precision in jump control, particularly in conjunction with external localization systems. We also aim to address the challenges observed with the current impulse-based WBC landing controller [5], which sometimes obtained failure results in landing phases. A primary focus will be developing a new landing controller that effectively manages horizontal and vertical speeds to preserve landing posture.



(a)

(b)

Fig. 5. Mini-Cheetah deployed with the omnidirectional jumping framework performs two jumping motions. Data charts in Fig. (a) and Fig. (b) are data records of the rear-right leg in front-left jumping and rear-right jumping, respectively. The blue line represents the start of the jumping phase, the dark red line indicates the start of the flight phase, the green line is the beginning of the landing phase, and the red dot lines represent the upper and lower bound of the torque limit with an absolute value of 24 Nm.

References

[1] M. Hutter, C. Gehring, A. Lauber, F. Gunther, C. D. Bellicoso, V. Tsounis, P. Fankhauser, R. Diethelm, S. Bachmann, M. Blösch, et al., "Anymal-toward legged robots for harsh environments," Advanced Robotics, vol. 31, no. 17, pp. 918–931, 2017

[2] J. K. Yim, B. R. P. Singh, E. K. Wang, R. Featherstone and R. S. Fearing, "Precision Robotic Leaping and Landing Using Stance-Phase Balance," Robotics and Automation Letters, vol. 5, no. 2, pp. 3422-3429, 2020

[3] H.-W. Park, M. Y. Chuah, and S. Kim, "Quadruped bounding control with variable duty cycle via vertical impulse scaling," in 2014 IEEE/RSJ International Conference on Intelligent Robots and Systems, 2014, pp. 3245–3252.

[4] 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," 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2018, pp. 1-9
[5] Kim, D., Carlo, J.D., Katz, B., Bledt, G., Kim, S. (2019). "Highly Dynamic Quadruped Locomotion via Whole-Body Impulse Control and Model Predictive Control". ArXiv, abs/1909.06586.
[6] Xie H, Ahmadi M, Shang J, Luo Z. An intuitive approach for quadruped robot trotting based on virtual model control. Proceedings of the Institution of Mechanical Engineers, Part I: Journal of Systems and Control Engineering. 2015;229(4):342-355
[7] Park H W, Wensing P M, Kim S. Jumping over obstacles with MIT Cheetah 2[J]. Robotics and Autonomous Systems, 2021, 136: 103703.
[8] M. Chignoli and S. Kim, "Online Trajectory Optimization for Dynamic Aerial Motions of a Quadruped Robot," 2021 IEEE International Conference on Robotics and Automation (ICRA), 2021, pp. 7693-7699
[9] Z. Song et al., "An Optimal Motion Planning Framework for Quadruped Jumping," 2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2022, pp. 11366-11373
[10] L. Yue, Z. Song, H. Zhang, X. Zeng, L. Zhang, and Y. -H. Liu, "Evolutionary-Based Online Motion Planning Framework for Quadruped Robot Jumping," 2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2023, pp. 767-773
[11] Bellegarda G, Nguyen C, Nguyen Q. Robust quadruped jumping via deep reinforcement learning[J]. arXiv preprint arXiv:2011.07089, 2020.
[12] "Unitree a1 robot," https://www.unitree.com/products/ a1/.
[13] Rudin N, Kolvenbach H, Tsounis V, et al. Cat-like jumping and landing of legged robots in low gravity using deep reinforcement learning[J]. IEEE Transactions on Robotics, 2021, 38(1): 317-328.
[14] L. Tang, Y. Li and B. Li, "Moobot: A Miniature Origami Omnidirectional Jumping Robot With High Trajectory Accuracy," IEEE Transactions on Industrial Electronics, vol. 71, no. 6, pp. 6032-6040
[15] Chignoli M, Morozov S, Kim S. "Rapid and reliable quadruped motion planning with omnidirectional jumping," 2022 International Conference on Robotics and Automation (ICRA). 2022: 6621-6627.
[16] Mellinger D, Kumar V. Minimum snap trajectory generation and control for quadrotors[C]//2011 IEEE international conference on robotics and automation. IEEE, 2011: 2520-2525.
[17] Mu B, Chirarattananon P. Trajectory generation for underactuated multirotor vehicles with tilted propellers via a flatness-based method[C]//2019 IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM). IEEE, 2019: 1365-1370.
[18] Richter C, Bry A, Roy N. Polynomial trajectory planning for aggressive quadrotor flight in dense indoor environments[C]//Robotics Research: The 16th International Symposium ISRR. Springer International Publishing, 2016: 649-666.
[19] F. Bullo and A. D. Lewis, Geometric Control of Mechanical Systems: Modeling, Analysis, and Design for Simple Mechanical Control Systems, vol. 49. Cham, Switzerland: Springer Science & Business Media, 2004.
[20] Focchi M, Del Prete A, Havoutis I, et al. High-slope terrain locomotion for torque-controlled quadruped robots[J]. Autonomous Robots, 2017, 41: 259-272.
[21] Goldfarb D, Idnani A. A numerically stable dual method for solving strictly convex quadratic programs[J]. Mathematical programming, 1983, 27(1): 1-33.
[22] https://github.com/liuq/QuadProgpp
[23] S. Fahmi, C. Mastalli, M. Focchi and C. Semini, "Passive Whole-Body Control for Quadruped Robots: Experimental Validation Over Challenging Terrain," in IEEE Robotics and Automation Letters, vol. 4, no. 3, pp. 2553-2560
[24] Katz B, Di Carlo J, Kim S. Mini cheetah: A platform for pushing the limits of dynamic quadruped control[C]. 2019 International Conference on Robotics and Automation (ICRA). , IEEE, 2019: 6295-6301.