Journal of Bionic Engineering (2023) 20:1449–1466
https://doi.org/10.1007/s42235-023-00347-9

RESEARCH ARTICLE

A Real-Time Planning and Control Framework for Robust and Dynamic Quadrupedal Locomotion

Jun Li[1,2] · Haibo Gao[1] · Yuhui Wan[2] · Haitao Yu[1] · Chengxu Zhou[2]

Received: 13 October 2022 / Revised: 7 January 2023 / Accepted: 10 January 2023 / Published online: 9 February 2023

Abstract
Legged locomotion poses significant challenges due to its nonlinear, underactuated and hybrid dynamic properties. These challenges are exacerbated by the high-speed motion and presence of aerial phases in dynamic legged locomotion, which highlights the requirement for online planning based on current states to cope with uncertainty and disturbances. This article proposes a real-time planning and control framework integrating motion planning and whole-body control. In the framework, the designed motion planner allows a wider body rotation range and fast reactive behaviors based on the 3-D single rigid body model. In addition, the combination of a Bézier curve based trajectory interpolator and a heuristic-based foothold planner helps generate continuous and smooth foot trajectories. The developed whole-body controller uses hierarchical quadratic optimization coupled with the full system dynamics, which ensures tasks are prioritized based on importance and joint commands are physically feasible. The performance of the framework is successfully validated in experiments with a torque-controlled quadrupedal robot for generating dynamic motions.

Keywords Bionic robot · Legged locomotion · Motion planning · Whole-body control

1 Introduction

Legged robots offer significant advantages over wheeled or tracked robots, as these advantages help traverse challenging terrain and unstructured environments. The ability to perform dynamic locomotion over unstructured terrain in unpredictable environments is extremely essential in real-world deployments. However, control of dynamic legged locomotion remains a challenging problem due to the difficulty in dynamical stabilization and the presence of unknown external disturbances. To achieve robust planning and control of dynamic locomotion, it is critical and necessary to adjust the motion plan according to the current robot state.

In recent years, some progresses have been achieved in dynamic legged locomotion with bipedal [1–5] and quadrupedal [6–15] robots. These solutions can be classified into optimization-based approaches [1, 2, 7, 10, 11] and learning-based approaches [4, 12, 13]. The optimization-based approach has been widely used, but it often needs to solve computationally expensive nonlinear optimization and suffers convergence issues, which makes them challenging to be implemented in real-time. The learning-based approach builds a neural network directly mapping sensor measurements to desired motor control signals but often requires high-quality training data and can not be adjusted partially. Compared with learning-based approaches that lack safety guarantees, the optimization-based approach is more suitable for the motion control of robots because it has the potential to take full-body dynamics and all physical constraints into consideration.

The optimization-based approach can be divided into trajectory optimization approaches [3, 16, 17] and predictive

✉ Haitao Yu
yht@hit.edu.cn

✉ Chengxu Zhou
c.x.zhou@leeds.ac.uk

Jun Li
junli@hit.edu.cn

Haibo Gao
gaohaibo@hit.edu.cn

Yuhui Wan
mnywa@leeds.ac.uk

1 State Key Laboratory of Robotics and System, Harbin Institute of Technology, Harbin 150001, China

2 School of Mechanical Engineering, University of Leeds, Leeds LS2 9JT, UK

control approaches [7, 10, 11] according to whether they can be run online. The trajectory optimization approach solves the locomotion problem offline over the long-time horizon of a specific motion while optimizing over a large number of variables consisting of contact sequences, contact timings, contact locations and the whole-body trajectory. These variables can be obtained automatically while taking into account physical constraints such as full-body dynamics, joint motion limits, unilateral contact constraints, and friction cone constraints. Conversely, the predictive control approach emphasizes solving the locomotion problem online over the short-time horizon and correcting motions to cope with model mismatches and unpredictable disturbances.

In the last few years, the trajectory optimization approach has been widely used in legged locomotion to generate complex trajectories with whole-body dynamics to explore more possible behaviors, solving the locomotion problem over a long-time horizon at once [3, 16–19]. For example, a method has been developed for whole-body trajectory generation of multi-limbed robotic systems with unilateral constraints introduced by contact with the environment, which can eliminate the requirement for a priori contact mode ordering and satisfy all dynamic and contact constraints [16]. Although many impressive results are presented, the trajectory optimization approach is too slow to be applied online on the real robot because of the high computational cost of nonlinear optimization problems.

Instead, some works in the predictive control approach focus on the online motion generation and execution on the real robot [6, 10, 11, 20]. For instance, The locomotion problem is formulated as a convex optimization by simplifying the robot dynamics and expressing the robot's orientation as Euler angles. While realizing the online planning of robot motions, some limitations are also introduced, such as the base cannot have a large range of roll and pitch motions, and the joint torque limits cannot be considered. A representation-free model predictive control framework is proposed to control various 3-D motions of a quadrupedal robot with the rotation matrix representation [11]. Since the rotation matrix is the natural representation of the special orthogonal group SO(3), representing orientation with the rotation matrix instead of Euler angles and quaternions can avoid the singularity and ambiguity issues [21, 22]. Although the orientation limitation is removed, the joint torque limit still can not be imposed. The use of simplified models and the omission of important constraints may make the generated trajectories unrealizable in the real robot.

To combine the advantages of the trajectory optimization approach and predictive control approach, there are some methods [7, 23, 24] trying to solve the locomotion problem online with differential dynamic programming (DDP), considering the whole-body dynamics and other physical constraints. These DDP-like methods only optimize a local control policy rather than control-state trajectories by iteratively solving a low-order Taylor approximation of the nonlinear locomotion problem, which makes them computationally efficient, but also limits their applicability as state constraints cannot be easily incorporated.

Most of the above works use only one model, some use a reduced-order model [6, 10, 11, 25], and some use a full-order model [7, 16, 23, 24, 26]. Actually, the whole locomotion control problem can be decomposed into a motion planning module and a motion tracking module [27–30]. The latter serves as a trajectory stabilizer calculating joint commands to track the task-space and joint-space trajectory references generated by the motion planner compliantly. The decomposition offers the possibility to use different models in motion generation and execution.

Thanks to the decomposition, this paper proposes a real-time planning and control framework shown in Fig. 1 for robust, dynamic quadrupedal locomotion containing two main modules: motion planning and whole-body control. The motion planner generates task-space references such as base motions, center-of-mass (CoM) motions, feet contact positions and forces, as well as limb motion reference trajectories. The tracking controller can execute various tasks and accomplish multiple control objectives, accounting for whole-body dynamics, manipulating contact forces to

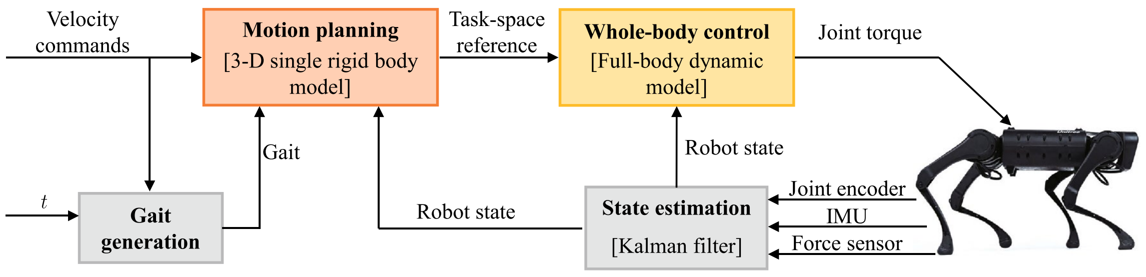


Fig. 1 Planning and control framework. Body velocity commands are sent to the motion planning module, which computes desired body pose, foot positions, and ground reaction forces. The whole-body control module generates desired generalized accelerations and external forces by solving a hierarchical quadratic programming problem. Then the joint torque, velocity and position commands of each joint are computed from the acceleration and force

control the floating base, and transforming task space references into joint-space commands. Although the gait scheduler and the state estimator are also included in the proposed framework, they will not be explained in detail here since they are out of the scope of this paper. Actually, many gait generation approaches [31, 32] and body pose estimation methods [33–36] can be used to schedule gait and estimate the base state of quadrupedal robots.

The contributions of this research can be summarized below:

1. A real-time planning and control framework is developed for quadrupedal robots capable of performing robust and dynamic locomotion, mainly consisting of a motion planner and a whole-body controller.
2. The 3-D single rigid body model (3D-SRBM) achieves a balance of expressivity and simplicity in the planning phase, allowing to conduct the planning and tracking of base orientation on SO(3) manifold to realize a wider body rotation range while allowing the planner to run in a receding horizon fashion to cope with disturbances.
3. A trajectory interpolator based on the Bézier curve is integrated with a heuristic-based foothold planner, which provides continuous and smooth Cartesian velocity and position of foot-ends simultaneously.
4. The developed whole-body controller uses hierarchical quadratic optimization coupled with the full system dynamics and physical constraints, which ensures tasks are prioritized based on importance, and joint commands sent to actuators are physically feasible.
5. The effectiveness of the framework is validated in real-world experiments on the *A1* robot.

The rest of this paper is organized as follows. The following section presents the motion planner generating body trajectory commands, desired foot reference trajectories, and contact forces. Section 3 presents the whole-body controller formulating the control problem as a series of quadratic programs. Section 4 describes the control framework's use on the physical robot. Finally, Sect. 5 gives discussions and concludes the paper.

2 Motion Planning

The motion planner receives operator input and generates body motion reference, foot motion trajectories, and optional reaction force profiles. Planning is achieved via three components, detailed in the following sections and illustrated in Fig. 2. The body motion reference sub-module is responsible for generating desired body velocity and pose that are used in the foot placement planning and contact force optimization sub-modules with the scheduled gait and estimated robot state to plan foot positions and reaction forces.

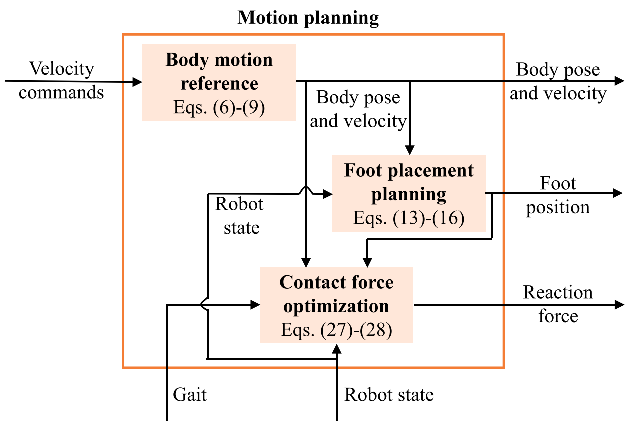


Fig. 2 A block diagram illustrates the interaction between the body motion reference, foot placement planning, and contact force optimization submodules

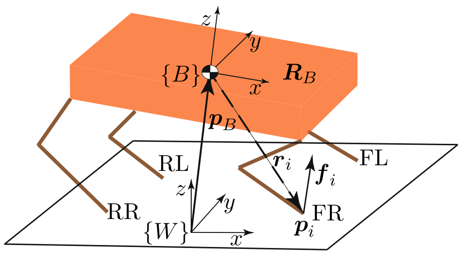


Fig. 3 3D-SRBM. Its CoM coincides with the CoM of the robot generated based on the mass and state of the whole robot. However, its orientation equals the base orientation of the robot, and its inertia is generated from a simplified robot body, where each leg is lumped to a point on the corresponding hip

2.1 3D-SRBM

The 3D-SRBM shown in Fig. 3 especially fits those robots with lightweight legs in which the robot base makes the largest contribution to the total mass of the whole robot. The underlying assumptions are: (1) their negligible limb masses do not contribute significantly to the momentum of the whole robotic system, and (2) the leg configuration does not influence the body inertia. This slightly more complex model approximates the robot as a single rigid body subject to external forces at the contact patches, thus it is still possible to consider the angular momentum in the planning phase even though it ignores limb dynamics.

The state of the 3D-SRBM can be collected into a tuple as

$$\boldsymbol{x} := \left(\boldsymbol{p}_B, \dot{\boldsymbol{p}}_B, \boldsymbol{R}_B, {}^B\boldsymbol{\omega}_B\right) \tag{1}$$

where $\boldsymbol{p}_B \in \mathbb{R}^3$ and $\dot{\boldsymbol{p}}_B \in \mathbb{R}^3$ are the CoM position and velocity, respectively. $\boldsymbol{R}_B \in \mathrm{SO}(3) = \{\boldsymbol{R} \in \mathbb{R}^{3\times3} | \boldsymbol{R}^{\mathrm{T}}\boldsymbol{R} = \mathbb{I}, \det(\boldsymbol{R}) = +1\}$ is the body orientation evolving on the SO(3) manifold. $\det(\cdot)$ calculates the determinant of a rotation matrix and $\mathbb{I}$ indicates the 3×3 identity matrix. ${}^{B}\boldsymbol{\omega}_B \in \mathbb{R}^3$ represents the angular velocity of the robot body expressed in the body frame $\{B\}$.

Equations of motion of the 3D-SRBM are given by

$$m\ddot{\boldsymbol{p}}_B = \sum_{i=1}^{n} \boldsymbol{f}_i + m\boldsymbol{a}_{\mathrm{g}}, \tag{2}$$

$$\frac{\mathrm{d}}{\mathrm{d}t}\left({}^{B}\boldsymbol{I}^{B}\boldsymbol{\omega}_B\right) = \sum_{i=1}^{n} \boldsymbol{R}_B^{\mathrm{T}}\boldsymbol{r}_i \times \boldsymbol{f}_i, \tag{3}$$

where m is the body mass; $\boldsymbol{f}_i$ and $\boldsymbol{a}_{\mathrm{g}}$ are three dimensional vectors representing the contact force, and gravitational acceleration; ${}^{B}\boldsymbol{I} \in \mathbb{R}^{3\times3}$ is the equivalent rotational inertia tensor expressed in the body frame $\{B\}$; n is the number of contacts; $\boldsymbol{r}_i$ is the location of i-th foot $\boldsymbol{p}_i$ relative to the CoM of the robot$\boldsymbol{p}_B$, which is equivalent to the moment arm of the contact force; $\boldsymbol{f}_i$ is the contact force of the i-th foot. By default, and unless otherwise stated, all of the above quantities are relative to the inertial frame $\{W\}$.

In the dynamics shown in Eqs. (2) and (3), $\boldsymbol{r}_i$ is known when the foothold location $\boldsymbol{p}_i$ is planned in advance. The contact force $\boldsymbol{f}_i \in \mathbb{R}^3$ is chosen as the system input. These contact forces create an external wrench

$$\mathcal{F} = \begin{bmatrix} \boldsymbol{f} \\ {}^{B}\boldsymbol{\tau} \end{bmatrix} = \sum_{i=1}^{n} \begin{bmatrix} \mathbf{1} \\ \boldsymbol{R}_B^{\mathrm{T}}\hat{\boldsymbol{r}}_i \end{bmatrix} \boldsymbol{f}_i, \tag{4}$$

where $i \in \{1, 2, 3, 4\}$ is the foot index for the front left (FL), front right (FR), rear left (RL), and rear right (RR), respectively. $\boldsymbol{f}$ and ${}^{B}\boldsymbol{\tau}$ are the total force and torque, respectively; the hat operator $\hat{(\cdot)}$ represents the cross-product operation.

Therefore, The dynamics of the rigid body Eqs. (2) and (3) can be rewritten as

$$\dot{\boldsymbol{x}} = \begin{bmatrix} \dot{\boldsymbol{p}}_B \\ \ddot{\boldsymbol{p}}_B \\ \dot{\boldsymbol{R}}_B \\ {}^{B}\dot{\boldsymbol{\omega}}_B \end{bmatrix} = \begin{bmatrix} \dot{\boldsymbol{p}}_B \\ \frac{1}{m}\boldsymbol{f} + \boldsymbol{a}_{\mathrm{g}} \\ \boldsymbol{R}_B \cdot {}^{B}\boldsymbol{\omega}_B \\ {}^{B}\boldsymbol{I}^{-1}\left({}^{B}\boldsymbol{\tau} - {}^{B}\hat{\boldsymbol{\omega}}_B{}^{B}\boldsymbol{I}^{B}\boldsymbol{\omega}_B\right) \end{bmatrix}. \tag{5}$$

2.2 Body Motion Reference

The body pose trajectory evolves on the special Euclidean SE(3) group. Although its motion can be divided as a translational motion in vector space and a rotational motion in special orthogonal group SO(3), it is still impossible to directly apply vector operations to the trajectory due to the special structure of SO(3) manifold. Unlike other representations such as the Euler angles and the unit quaternions, the natural nine parameters representation of the body orientation as an element of SO(3) is unique and non-singular. Therefore, it is more appropriate to use the rotation matrix representation.

The desired velocity $\boldsymbol{V}_{B,k} = \left({}^{W}\boldsymbol{\omega}_{B,k}, {}^{W}\boldsymbol{v}_{B,k}\right)$ and pose $\boldsymbol{X}_{B,k} = \left(\boldsymbol{R}_{B,k}, \boldsymbol{p}_{B,k}\right)$ are updated at every time step k. The above values are calculated with a reference velocity input $\boldsymbol{V} = \left(\boldsymbol{\omega}_{k+1}^{\mathrm{cmd}}, \boldsymbol{v}_{k+1}^{\mathrm{cmd}}\right)$ expressed in the body coordinate from a complex path planner or a user-operated joystick. The desired body velocity and pose are updated as

$${}^{W}\boldsymbol{v}_{B,k+1} = \boldsymbol{R}_{B,k}\boldsymbol{v}_{k+1}^{\mathrm{cmd}}, \tag{6}$$

$$\boldsymbol{p}_{B,k+1} = \boldsymbol{p}_{B,k} + {}^{W}\boldsymbol{v}_{B,k+1}\Delta t, \tag{7}$$

$${}^{W}\boldsymbol{\omega}_{B,k+1} = \boldsymbol{R}_{B,k}\boldsymbol{\omega}_{k+1}^{\mathrm{cmd}}, \tag{8}$$

$$\boldsymbol{R}_{B,k+1} = \boldsymbol{R}_{B,k}\mathrm{Exp}\left(\boldsymbol{\omega}_{k+1}^{\mathrm{cmd}}\Delta t\right), \tag{9}$$

where Δt is the timestep, and the mapping function $\mathrm{Exp} : \mathbb{R}^3 \to \mathrm{SO}(3)$ is the retraction for SO(3).

2.3 Foot Placement Planning

The desired foothold location is determined by a foot placement planner designed based on the Raibert Heuristic [37] and Capture Point [38]. The main idea is to force the leg's landing angle to be the same as its leaving angle if the robot moves at the commanded speed. the location of foot i is calculated with the following equation:

$$\boldsymbol{p}_i = \boldsymbol{p}_B + \boldsymbol{R}_z(\psi)^{B}\boldsymbol{l}_i + \frac{T_{\mathrm{st}}}{2}\boldsymbol{v}_i + K\left(\boldsymbol{v}_i - \boldsymbol{v}_i^{\mathrm{cmd}}\right) \tag{10}$$

where $\boldsymbol{p}_B$ is the body position, $\boldsymbol{R}_z(\psi)$ is the body orientation expressed as a rotation matrix, only the rotation along z-axis is considered. ${}^{B}\boldsymbol{l}_i$ is the position of hip i relative to the body position $\boldsymbol{p}_B$, T_{st} is the stance time for each foot, $\boldsymbol{v}_i$ is the velocity of hip i, $\boldsymbol{v}_i^{\mathrm{cmd}}$ is the desired velocity of hip i which can be calculated by the body linear and angular velocity commands, and K represents a gain, usually taking the value 0.03.

Although the expected foothold location is given by Eq. (10), the foot still needs to lift off and touch down periodically to make the robot move. During the swing phase, when the foot lifts off in the air, the planned velocity and position of the foot should be smooth enough for the whole-body controller to track. The Bézier curve is chosen to generate the swing foot trajectory, for it just needs a set of discrete control points to define a smooth and continuous curve.

Given the start foot-end position $\boldsymbol{p}_{i,k}^{\mathrm{s}}$ recorded when the foot lifts off, the desired foot-end position $\boldsymbol{p}_{i,k}^{\mathrm{e}}$ calculated by

Eq. (10), the phase signal $\phi_{i,k}$ from the gait scheduler, and the maximum foot clearance h_f, the quadratic Bézier curve that requires only three control points is naturally suitable for interpolating out the reference velocity and position of foot i in swing-phase at the k-th timestep. Figure 4 illustrates the diagram of generating swing foot trajectory with the quadratic Bézier curve.

The conventional procedure to use the quadratic Bézier curve is to first determine the coordinates of the control point $\boldsymbol{p}^{\mathrm{c}}_{i,k}$. Then it inputs the coordinates of start point $\boldsymbol{p}^{\mathrm{s}}_{i,k}$, control point $\boldsymbol{p}^{\mathrm{c}}_{i,k}$ and end point $\boldsymbol{p}^{\mathrm{e}}_{i,k}$ into standard Bézier curve algorithms. If the projection of control point $\boldsymbol{p}^{\mathrm{c}}_{i,k}$ on the x–y plane coincides with the middle point $\boldsymbol{p}^{\mathrm{m}}_{i,k}$, the conventional procedure can be simplified to firstly obtain foot position and velocity with only the start and end points, and then correct the z-component with the desired foot clearance. These two steps are shown as follows

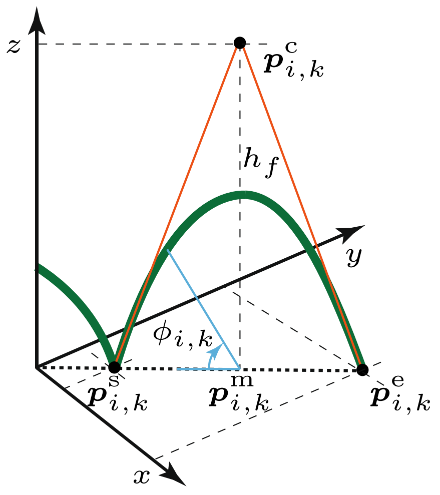
$$B_k(\phi_{i,k}) = \phi_{i,k}^3 + 3\phi_{i,k}^2\left(1 - \phi_{i,k}\right) \tag{11}$$

$$B_k^{'}(\phi_{i,k}) = 6\phi_{i,k}\left(1 - \phi_{i,k}\right) \tag{12}$$

$$\boldsymbol{p}^{\mathrm{ref}}_{i,k} = \boldsymbol{p}^{\mathrm{s}}_{i,k} + B_k(\phi_{i,k}) \cdot \left(\boldsymbol{p}^{\mathrm{e}}_{i,k} - \boldsymbol{p}^{\mathrm{s}}_{i,k}\right)/T_{\mathrm{sw}} \tag{13}$$

$$\boldsymbol{v}^{\mathrm{ref}}_{i,k} = B_k^{'}(\phi_{i,k}) \cdot \left(\boldsymbol{p}^{\mathrm{e}}_{i,k} - \boldsymbol{p}^{\mathrm{s}}_{i,k}\right)/T_{\mathrm{sw}}^2 \tag{14}$$

where, B_k is the Bézier curve, $B_k^{'}$ is the derivative of the Bézier curve, T_{sw} is the swing time for each foot, $\boldsymbol{p}^{\mathrm{ref}}_{i,k}$ and $\boldsymbol{v}^{\mathrm{ref}}_{i,k}$ are interpolated foot-end position and velocity. The foot height and vertical velocity are corrected with



z
$\boldsymbol{p}^{\mathrm{c}}_{i,k}$
h_f
y
$\phi_{i,k}$
$\boldsymbol{p}^{\mathrm{s}}_{i,k}$
$\boldsymbol{p}^{\mathrm{m}}_{i,k}$
$\boldsymbol{p}^{\mathrm{e}}_{i,k}$
x

Fig. 4 A schematic diagram showing how to generate swing foot trajectory with the quadratic Bézier curve. Three black circles represent the start point, control point, and end point, respectively. The green thick line represents the generated swing foot trajectory. The light blue denotes the phase signal

$$\begin{aligned}\boldsymbol{p}^{\mathrm{ref}}_{i,k}(2) &= \boldsymbol{p}^{\mathrm{s}}_{i,k}(2)\\ &+ \begin{cases} B_k\left(2\phi_{i,k}\right) \cdot \left(\boldsymbol{p}^{\mathrm{s}}_{i,k}(2) + h_f - \boldsymbol{p}^{\mathrm{s}}_{i,k}(2)\right)\frac{2}{T_{\mathrm{sw}}}, & \phi_{i,k} \le 0.5\\ B_k\left(2\phi_{i,k} - 1\right) \cdot \left(\boldsymbol{p}^{\mathrm{e}}_{i,k}(2) - \left(\boldsymbol{p}^{\mathrm{s}}_{i,k}(2) + h_f\right)\right)\frac{2}{T_{\mathrm{sw}}}, & \phi_{i,k} > 0.5\end{cases}\end{aligned} \tag{15}$$

$$\begin{aligned}&\boldsymbol{v}^{\mathrm{ref}}_{i,k}(2)\\ &= \begin{cases} B_k^{'}(2\phi_{i,k}) \cdot \left(\boldsymbol{p}^{\mathrm{s}}_{i,k}(2) + h_f - \boldsymbol{p}^{\mathrm{s}}_{i,k}(2)\right)\frac{4}{T_{\mathrm{sw}}^2}, & \phi_{i,k} \le 0.5\\ B_k^{'}(2\phi_{i,k} - 1) \cdot \left(\boldsymbol{p}^{\mathrm{e}}_{i,k}(2) - \left(\boldsymbol{p}^{\mathrm{s}}_{i,k}(2) + h_f\right)\right)\frac{4}{T_{\mathrm{sw}}^2}, & \phi_{i,k} > 0.5\end{cases}.\end{aligned} \tag{16}$$

2.4 Contact Force Optimization

The desired contact forces are calculated by an optimization problem making the robot follow the given body trajectory updated by equations from (6) to (9). To make the optimization convex, some approximations [6] can be made. To be specific, the system dynamics Eq. (5) and the contact friction cone constraint should be linearized.

The body orientation can be parametrized as Z–Y–X Euler angles $\boldsymbol{\Theta} = \left[\phi \;\; \theta \;\; \psi\right]^{\mathrm{T}}$, where ϕ, θ and ψ are the roll, pitch and yaw angles, respectively. Therefore, the transformation from the body frame to the inertial frame can be decomposed as

$$\boldsymbol{R}_B = \boldsymbol{R}_z(\psi)\boldsymbol{R}_y(\theta)\boldsymbol{R}_x(\phi), \tag{17}$$

where $\boldsymbol{R}_x(\phi)$, $\boldsymbol{R}_y(\theta)$ and $\boldsymbol{R}_z(\psi)$ are rotations of ϕ, θ and ψ about the X-, Y-, and Z-axis, respectively.

With this parametrization, the relationship between the body angular velocity and the rate of change of Euler angles can be expressed as

$$^{W}\boldsymbol{\omega}_B = \begin{bmatrix}\cos(\theta)\cos(\psi) & -\sin(\psi) & 0\\ \cos(\theta)\sin(\psi) & \cos(\psi) & 0\\ 0 & 0 & 1\end{bmatrix}\begin{bmatrix}\dot{\phi}\\ \dot{\theta}\\ \dot{\psi}\end{bmatrix}. \tag{18}$$

If roll and pitch angles are small and the body is not erected or inverted, the rate of Euler angles is

$$\begin{aligned}\dot{\boldsymbol{\Theta}} &\approx \begin{bmatrix}\cos(\psi) & \sin(\psi) & 0\\ -\sin(\psi) & \cos(\psi) & 0\\ 0 & 0 & 1\end{bmatrix} {}^{W}\boldsymbol{\omega}_B\\ &= \boldsymbol{R}_z^{\mathrm{T}}(\psi)^{W}\boldsymbol{\omega}_B = {}^{B}\boldsymbol{\omega}_B.\end{aligned} \tag{19}$$

It is worth noting that the rotation order is critical, with other order, the above approximation will not hold.

If the angular velocity is small, the term $\boldsymbol{\omega} \times (\boldsymbol{I}\boldsymbol{\omega})$ will not contribute significantly to the body orientation dynamics. Then, the orientation dynamics can be approximated with

$$\frac{\mathrm{d}}{\mathrm{d}t}(\boldsymbol{I}\boldsymbol{\omega}) = \boldsymbol{I}\dot{\boldsymbol{\omega}} + \boldsymbol{\omega} \times (\boldsymbol{I}\boldsymbol{\omega}) \approx \boldsymbol{I}\dot{\boldsymbol{\omega}}. \tag{20}$$

With small roll and pitch angles assumption, the inertia tensor and the orientation dynamics become

$$^{W}\boldsymbol{I} = \boldsymbol{R}_B {}^{B}\boldsymbol{I}\boldsymbol{R}_B^{\mathrm{T}} = \boldsymbol{R}_z(\psi){}^{B}\boldsymbol{I}\boldsymbol{R}_z^{\mathrm{T}}(\psi), \tag{21}$$

$$\begin{aligned} {}^{W}\boldsymbol{I}{}^{W}\dot{\boldsymbol{\omega}}_B &= \boldsymbol{R}_z(\psi){}^{B}\boldsymbol{I}\boldsymbol{R}_z^{\mathrm{T}}(\psi)\boldsymbol{R}_z(\psi){}^{B}\dot{\boldsymbol{\omega}}_B \\ &= \sum_{i=1}^{n} \boldsymbol{r}_i \times \boldsymbol{f}_i \end{aligned} \quad . \tag{22}$$

After combining Eqs. (2), (19) and (20), the linearized robot dynamics can be expressed as

$$\ddot{\boldsymbol{q}}_B = \boldsymbol{M}\boldsymbol{f} + \bar{\boldsymbol{g}} \tag{23}$$

where

$$\begin{aligned} \ddot{\boldsymbol{q}}_B &= \left[\ddot{\boldsymbol{p}}_B^{\mathrm{T}}\ {}^{B}\dot{\boldsymbol{\omega}}_B^{\mathrm{T}}\right]^{\mathrm{T}}, \\ \boldsymbol{M} &= \begin{bmatrix} \boldsymbol{1}_3/m & \cdots & \boldsymbol{1}_3/m \\ {}^{B}\boldsymbol{I}^{-1}\boldsymbol{R}_z^{\mathrm{T}}(\psi)\hat{\boldsymbol{r}}_1 & \cdots & {}^{B}\boldsymbol{I}^{-1}\boldsymbol{R}_z^{\mathrm{T}}(\psi)\hat{\boldsymbol{r}}_n \end{bmatrix}, \\ \boldsymbol{f} &= \left[\boldsymbol{f}_1{}^{\mathrm{T}} \cdots \boldsymbol{f}_n{}^{\mathrm{T}}\right]^{\mathrm{T}}, \\ \bar{\boldsymbol{g}} &= \left[\boldsymbol{a}_{\mathrm{g}}^{\mathrm{T}}\ \boldsymbol{0}_3^{\mathrm{T}}\right]^{\mathrm{T}}. \end{aligned} \tag{24}$$

When the foot is in stance phase, a friction cone constraint must be imposed on the contact force $\boldsymbol{f} \in \mathbb{R}^3$ as follows

$$\sqrt{f_x^2 + f_y^2} \leqslant \mu f_z, \quad f_z \geqslant 0. \tag{25}$$

Since the friction cone constraint poses some computational challenges, a better option is to consider its conservative pyramid approximation

$$|f_x| \leqslant \mu f_z, \quad |f_y| \leqslant \mu f_z, \quad f_z \geqslant 0. \tag{26}$$

Given a desired body pose $\boldsymbol{q}_B^{\mathrm{des}} = \left[\boldsymbol{p}_B^{\mathrm{T}}\ \boldsymbol{\Theta}^{\mathrm{T}}\right]^{\mathrm{T}}$ and velocity $\dot{\boldsymbol{q}}_B^{\mathrm{des}} = \left[\dot{\boldsymbol{p}}_B^{\mathrm{T}}\ \dot{\boldsymbol{\Theta}}^{\mathrm{T}}\right]^{\mathrm{T}}$, we use PD control to compute the desired acceleration

$$\ddot{\boldsymbol{q}}_B^{\mathrm{des}} = K_{\mathrm{p}}\left(\boldsymbol{q}_B^{\mathrm{des}} - \boldsymbol{q}_B\right) + K_{\mathrm{d}}\left(\dot{\boldsymbol{q}}_B^{\mathrm{des}} - \dot{\boldsymbol{q}}_B\right) \tag{27}$$

where K_{p} is position gain, and K_{d} is velocity gain. $\boldsymbol{q}_B$ and $\dot{\boldsymbol{q}}_B$ are the current body pose and velocity. With equations from (23) to (24) and Eq. (27), a Quadratic Program (QP) can be constructed to find contact forces, which includes an acceleration error cost, contact selection constraints and linearized friction cone constraints.

$$\begin{aligned} \min_{\boldsymbol{f}} \quad & \frac{1}{2}\|\boldsymbol{M}\boldsymbol{f} + \bar{\boldsymbol{g}} - \ddot{\boldsymbol{q}}_B^{\mathrm{des}}\|_{\mathbf{Q}} + \frac{1}{2}\|\boldsymbol{f}\|_{\mathbf{R}} \\ \text{s.t.} \quad & \boldsymbol{H}\boldsymbol{f} \leqslant 0 \quad \text{(friction pyramid)} \\ & \boldsymbol{D}\boldsymbol{f} = 0 \quad \text{(contact selection)} \end{aligned} \tag{28}$$

where $\mathbf{Q}$ and $\mathbf{R}$ represent diagonal weight matrices, and $\boldsymbol{H}$ is the friction pyramid matrix corresponding to friction cone constraints Eq. (26), and $\boldsymbol{D}$ is the contact selection matrix for swing feet.

3 Whole-Body Control

After the motion planner generates reference motions and reaction forces, the whole-body controller computes joint torque, velocity and position commands. Motion tracking is achieved through three submodules and the interaction between them is illustrated in Fig. 5. First, the task formulation sub-module formulates task-space tasks as linear functions of generalized accelerations and external forces and puts them in a task hierarchy. Then, the hierarchy is solved by the hierarchical optimization sub-module to get the optimal generalized accelerations and external forces. Finally, the desired joint torques is calculated from the optimal accelerations and forces in the hybrid joint controller sub-module.

3.1 System Underactuation

The quadruped robot shown in Fig. 6 belongs to the multi-body dynamic system, and its equations of motion is relatively complicated, written as

$$\boldsymbol{M}(\boldsymbol{q})\dot{\boldsymbol{v}} + \boldsymbol{h}(\boldsymbol{q}, \boldsymbol{v}) = \boldsymbol{S}^{\mathrm{T}}\boldsymbol{\tau} + \boldsymbol{J}_c^{\mathrm{T}}(\boldsymbol{q})\boldsymbol{\lambda}, \tag{29}$$

where $\boldsymbol{q} \in \mathbb{R}^{n_q}$ and $\boldsymbol{v} \in \mathbb{R}^{n_v}$ are generalized coordinates and velocities, and for those robotic systems with floating-bases,

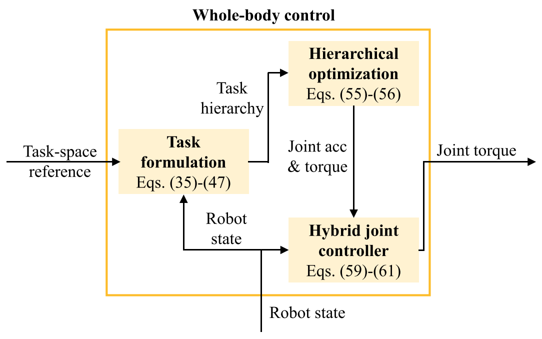

Whole-body control
Hierarchical optimization Eqs. (55)-(56)
Task hierarchy
Task-space reference
Task formulation Eqs. (35)-(47)
Joint acc & torque
Joint torque
Robot state
Hybrid joint controller Eqs. (59)-(61)
Robot state

Fig. 5 A block diagram illustrates the interaction between the task formulation, hierarchical optimization, and hybrid joint controller submodules

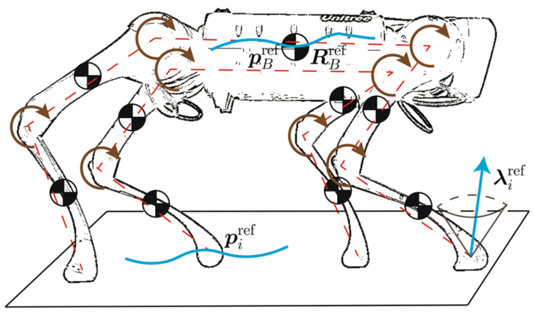


Fig. 6 Full body dynamic model. Blue represents task-space motion and force references, red indicates the full-body dynamics, and brown represents joint commands

n_q is different with n_v. $\boldsymbol{M}(\boldsymbol{q}) \in \mathbb{R}^{n_v \times n_v}$ stands for the generalized inertia matrix of the robot, and $\boldsymbol{h}(\boldsymbol{q}, \boldsymbol{v}) \in \mathbb{R}^{n_v}$ is the nonlinear term (including Coriolis, centrifugal and gravity forces). $\boldsymbol{S} = \left[\boldsymbol{0}_{n_j \times (n_v - n_j)}\ \boldsymbol{I}_{n_j \times n_j}\right]$ is the selection matrix, representing the system under-actuation that the floating-base is not directly actuated by joint torques $\tau \in \mathbb{R}^{n_j}$. $\boldsymbol{J}_c = \left[\boldsymbol{J}_{c_1}^{\mathrm{T}} \ \cdots \ \boldsymbol{J}_{c_{n_c}}^{\mathrm{T}}\right]^{\mathrm{T}} \in \mathbb{R}^{3n_c \times n_v}$ is the Jacobian matrix mapping contact forces $\boldsymbol{\lambda} \in \mathbb{R}^{3n_c}$ to the joint-space torques.

The equations of motion Eq. (29) of a physical system describes the relationship between the accelerations $\dot{\boldsymbol{v}}$, the external contact forces $\boldsymbol{\lambda}$ and the joint torques $\boldsymbol{\tau}$. For a quadruped robot, its equations of motion can be decomposed into underactuated and actuated parts

$$\boldsymbol{M}_{\mathrm{u}}(\boldsymbol{q})\dot{\boldsymbol{v}} + \boldsymbol{h}_{\mathrm{u}}(\boldsymbol{q}, \boldsymbol{v}) = \boldsymbol{J}_{c,\mathrm{u}}^{\mathrm{T}} \boldsymbol{\lambda}, \tag{30}$$

$$\boldsymbol{M}_{\mathrm{a}}(\boldsymbol{q})\dot{\boldsymbol{v}} + \boldsymbol{h}_{\mathrm{a}}(\boldsymbol{q}, \boldsymbol{v}) = \boldsymbol{\tau} + \boldsymbol{J}_{c,\mathrm{a}}^{\mathrm{T}} \boldsymbol{\lambda}, \tag{31}$$

where Eq. (30) excluding joint torques is the first 6 equations of Eq. (29) related to the floating base, and Eq. (31) is the rest of the equations. Equation (30) expresses the relationship between the centroidal momentum of the whole robot and external contact forces. The joint torques $\boldsymbol{\tau}$ only appear in Eq. (31) and is linearly dependent on $\dot{\boldsymbol{v}}$, so $\boldsymbol{\lambda}$ can be expressed as

$$\boldsymbol{\tau} = \boldsymbol{M}_{\mathrm{a}}(\boldsymbol{q})\dot{\boldsymbol{v}} + \boldsymbol{h}_{\mathrm{a}}(\boldsymbol{q}, \boldsymbol{v}) - \boldsymbol{J}_{c,\mathrm{a}}^{\mathrm{T}} \boldsymbol{\lambda}. \tag{32}$$

For any combination of generalized accelerations $\dot{\boldsymbol{v}}$ and external contact forces $\boldsymbol{\lambda}$, there always exists a solution for joint torques $\boldsymbol{\tau}$. All occurrences of the joint torques can be replaced with Eq. (32), so it is enough to use Eq. (30) instead of Eq. (29) as the dynamic constraint during the optimization. The optimization variable $\boldsymbol{z}$ can be chosen as

$$\boldsymbol{z} = \left[\dot{\boldsymbol{v}}^{\mathrm{T}} \ \boldsymbol{\lambda}^{\mathrm{T}}\right]^{\mathrm{T}} \in \mathbb{R}^{n_v + 3n_c}. \tag{33}$$

This decomposition largely reduces the number of optimization variables. As a result, it reduces solution time and allows the controller to run at a frequency of 500 Hz in real test.

3.2 Task Formulation

All robot tasks are often modelled in the form of a general task $\mathbf{T}$ expressed as constraints on the optimization variable $\boldsymbol{z}$, including linear equality and/or inequality:

$$\mathbf{T}: \begin{cases} \boldsymbol{A}\boldsymbol{z} - \boldsymbol{b} = \boldsymbol{s}_{\mathrm{eq}} \\ \boldsymbol{C}\boldsymbol{z} - \boldsymbol{d} \leqslant \boldsymbol{s}_{\mathrm{in}} \end{cases} \tag{34}$$

where $\boldsymbol{A} \in \mathbb{R}^{m \times (n_v + 3n_c)}$ is the equality constraint matrix, $\boldsymbol{b} \in \mathbb{R}^m$ is the equality constraint vector, $\boldsymbol{C} \in \mathbb{R}^{k \times (n_v + 3n_c)}$ is the inequality constraint matrix, and $\boldsymbol{d} \in \mathbb{R}^k$ is the inequality constraint vector. $\boldsymbol{s}_{\mathrm{eq}} \in \mathbb{R}^m$ and $\boldsymbol{s}_{\mathrm{in}} \in \mathbb{R}^k$ are slack variables. It should be noted that, for a specific task, the above equality constraint and inequality constraint may exist together or independently.

(1) *Dynamic consistency* To ensure physical consistency, the underactuated part Eq. (30) can be expressed in the general task form Eq. (34) with

$$\begin{aligned} \boldsymbol{A}_{\mathrm{dyn}} &= \left[\boldsymbol{M}_{\mathrm{u}}(\boldsymbol{q}) \ \boldsymbol{J}_{c,\mathrm{u}}^{\mathrm{T}}\right], \\ \boldsymbol{b}_{\mathrm{dyn}} &= \boldsymbol{h}_{\mathrm{u}}(\boldsymbol{q}, \boldsymbol{v}), \end{aligned} \tag{35}$$

which serves as the dynamic consistency constraint.

(2) *Torque saturation limits* To ensure the solved joint torques are physically consistent, the joint torque limitations should be considered. The torque limitation constraint can be formulated in the general task form Eq. (34) with

$$\begin{aligned} \boldsymbol{C}_{\mathrm{tau}} &= \begin{bmatrix} -\boldsymbol{M}_{\mathrm{a}}(\boldsymbol{q}) & \boldsymbol{J}_{c,\mathrm{a}}^{\mathrm{T}} \\ \boldsymbol{M}_{\mathrm{a}}(\boldsymbol{q}) & -\boldsymbol{J}_{c,\mathrm{a}}^{\mathrm{T}} \end{bmatrix}, \\ \boldsymbol{d}_{\mathrm{tau}} &= \begin{bmatrix} \boldsymbol{h}_{\mathrm{a}}(\boldsymbol{q}, \boldsymbol{v}) - \boldsymbol{\tau}^{\min} \\ -\boldsymbol{h}_{\mathrm{a}}(\boldsymbol{q}, \dot{\boldsymbol{q}}) + \boldsymbol{\tau}^{\max} \end{bmatrix}. \end{aligned} \tag{36}$$

where $\boldsymbol{\tau}^{\min}$ and $\boldsymbol{\tau}^{\max}$ are the lower and upper torque limits, respectively.

(3) *Contact force limits* When the point feet are in contact with the ground, friction cone constraints on the resulting reaction forces $\boldsymbol{\lambda} \in \mathbb{R}^{3n_c}$ should be enforced. All contact force constraints for active contacts have been derived from Eq. (25) to Eq. (26), so the contact force limits constraint is expressed as

$$\begin{aligned} \boldsymbol{C}_{\mathrm{cf}} &= \left[\boldsymbol{0} \ \boldsymbol{H}\right], \\ \boldsymbol{d}_{\mathrm{cf}} &= \boldsymbol{0}. \end{aligned} \tag{37}$$

(4) *Contact motion task* When the point feet are in stance phase, three constraint equations are introduced

$\boldsymbol{p}_i(t) = \text{const}$. Then, the constraint is differentiated twice to generate the following:

$$\dot{\boldsymbol{p}}_i = \boldsymbol{J}_{\mathrm{c}_i}\boldsymbol{v} = \boldsymbol{0}, \quad \ddot{\boldsymbol{p}}_i = \boldsymbol{J}_{\mathrm{c}_i}\dot{\boldsymbol{v}} + \dot{\boldsymbol{J}}_{\mathrm{c}_i}\boldsymbol{v} = \boldsymbol{0}, \tag{38}$$

where $\boldsymbol{J}_{\mathrm{c}_i}$ is the corresponding end-effector Jacobian. By stacking the above constraints for all active contacts together, the contact motion task can be formulated as

$$\begin{aligned} \boldsymbol{A}_{\mathrm{cm}} &= \left[\boldsymbol{J}_{\mathrm{c}}\ \boldsymbol{0}\right], \\ \boldsymbol{b}_{\mathrm{cm}} &= -\dot{\boldsymbol{J}}_{\mathrm{c}}\boldsymbol{v}. \end{aligned} \tag{39}$$

(5) *Contact force regularization task* Given reference contact forces $\boldsymbol{\lambda}^{\mathrm{ref}}$ generated by Eq. (28), we can also directly penalize the optimization variable $\boldsymbol{z}$ from reference values, with

$$\begin{aligned} \boldsymbol{A}_{\mathrm{cfr}} &= \left[\boldsymbol{0}\ \boldsymbol{I}\right], \\ \boldsymbol{b}_{\mathrm{cfr}} &= \boldsymbol{\lambda}^{\mathrm{ref}}. \end{aligned} \tag{40}$$

This contact force regularization task is helpful for directly controlling specific contact forces. It also provides regularization on the optimization variable $\boldsymbol{z}$ to solve the QP problem faster.

(6) *Cartesian space position control task* For the body with position $\boldsymbol{p} \in \mathbb{R}^3$ and linear velocity $\dot{\boldsymbol{p}} \in \mathbb{R}^3$, the relationship between linear velocity in Cartesian space and the generalized velocity is

$$\dot{\boldsymbol{p}} = \boldsymbol{J}_{\mathrm{pos}}(\boldsymbol{q})\boldsymbol{v}, \tag{41}$$

which can be differentiated as

$$\ddot{\boldsymbol{p}} = \boldsymbol{J}_{\mathrm{pos}}(\boldsymbol{q})\dot{\boldsymbol{v}} + \dot{\boldsymbol{J}}_{\mathrm{pos}}(\boldsymbol{q}, \boldsymbol{v})\boldsymbol{v}. \tag{42}$$

The deviation from the desired Cartesian linear acceleration can be penalized with

$$\begin{aligned} \boldsymbol{A}_{\mathrm{pos}} &= \left[\boldsymbol{J}_{\mathrm{pos}}(\boldsymbol{q})\ \boldsymbol{0}\right], \\ \boldsymbol{b}_{\mathrm{pos}} &= \ddot{\boldsymbol{p}}^{\mathrm{ref}} + \boldsymbol{K}_{\mathrm{p}}^{\mathrm{pos}}(\boldsymbol{p}^{\mathrm{ref}} - \boldsymbol{p}) \\ &\quad + \boldsymbol{K}_{\mathrm{d}}^{\mathrm{pos}}(\dot{\boldsymbol{p}}^{\mathrm{ref}} - \dot{\boldsymbol{p}}) - \dot{\boldsymbol{J}}_{\mathrm{pos}}(\boldsymbol{q}, \boldsymbol{v})\boldsymbol{v}. \end{aligned} \tag{43}$$

where $\boldsymbol{K}_{\mathrm{p}}^{\mathrm{pos}}$ and $\boldsymbol{K}_{\mathrm{d}}^{\mathrm{pos}}$ are proportional gain and derivative gain, usually taking diagonal positive definite matrices. $\boldsymbol{p}^{\mathrm{ref}}$, $\dot{\boldsymbol{p}}^{\mathrm{ref}}$ and $\ddot{\boldsymbol{p}}^{\mathrm{ref}}$ are specified by a high level motion planner, $\boldsymbol{p}$ and $\dot{\boldsymbol{p}}$ are computed with forward kinematics using the estimated state. Many tasks, such as *CoM position control task*, *Swing foot position control task*, are specified with this form.

(7) *Cartesian space orientation control task* For the body with orientation $\boldsymbol{R} \in \mathrm{SO}(3)$ and angular velocity $\boldsymbol{\omega} \in \mathbb{R}^3$, the relationship between angular velocity in Cartesian space and the generalized velocity is

$$\boldsymbol{\omega} = \boldsymbol{J}_{\mathrm{ori}}(\boldsymbol{q})\boldsymbol{v}, \tag{44}$$

which can be differentiated as

$$\dot{\boldsymbol{\omega}} = \boldsymbol{J}_{\mathrm{ori}}(\boldsymbol{q})\dot{\boldsymbol{v}} + \dot{\boldsymbol{J}}_{\mathrm{ori}}(\boldsymbol{q}, \boldsymbol{v})\boldsymbol{v}. \tag{45}$$

The deviation from the desired Cartesian angular acceleration can be penalized with

$$\begin{aligned} \boldsymbol{A}_{\mathrm{ori}} &= \left[\boldsymbol{J}_{\mathrm{ori}}(\boldsymbol{q})\ \boldsymbol{0}\right], \\ \boldsymbol{b}_{\mathrm{ori}} &= \dot{\boldsymbol{\omega}}^{\mathrm{ref}} + \boldsymbol{K}_{\mathrm{p}}^{\mathrm{ori}}\boldsymbol{R}\mathrm{Log}\left(\boldsymbol{R}^{\mathrm{T}}\boldsymbol{R}^{\mathrm{ref}}\right) \\ &\quad + \boldsymbol{K}_{\mathrm{d}}^{\mathrm{ori}}(\boldsymbol{\omega}^{\mathrm{ref}} - \boldsymbol{\omega}) - \dot{\boldsymbol{J}}_{\mathrm{ori}}(\boldsymbol{q}, \boldsymbol{v})\boldsymbol{v}. \end{aligned} \tag{46}$$

where $\boldsymbol{K}_{\mathrm{p}}^{\mathrm{ori}}$ and $\boldsymbol{K}_{\mathrm{d}}^{\mathrm{ori}}$ proportional gain and derivative gain, usually taking diagonal positive definite matrices. $\boldsymbol{R}^{\mathrm{ref}}$, $\boldsymbol{\omega}^{\mathrm{ref}}$ and $\dot{\boldsymbol{\omega}}^{\mathrm{ref}}$ are specified by a high level motion planner, $\boldsymbol{R}$ and $\boldsymbol{\omega}$ are computed by forward kinematics with the estimated state, and $\mathrm{Log} : \mathrm{SO}(3) \to \mathbb{R}^3$ is the logarithm map. Some tasks such as *Base orientation control task* are specified with this form.

(8) *Minimum motion task* In order to determine all joint states and avoid internal drift, the research also introduces a task spanning the configuration space. For example, the deviation can be penalised from the nominal configuration and velocity with

$$\begin{aligned} \boldsymbol{A}_{\mathrm{nom}} &= \left[\left[\boldsymbol{0}\ \boldsymbol{I}\right]\ \boldsymbol{0}\right], \\ \boldsymbol{b}_{\mathrm{nom}} &= \boldsymbol{K}_{\mathrm{p}}^{\mathrm{nom}}\left(\boldsymbol{q}_{n_\tau}^{\mathrm{nom}} - \boldsymbol{q}_{n_\tau}\right) \\ &\quad + \boldsymbol{K}_{\mathrm{d}}^{\mathrm{nom}}\left(\boldsymbol{v}_{n_\tau}^{\mathrm{nom}} - \boldsymbol{v}_{n_\tau}\right). \end{aligned} \tag{47}$$

where $\boldsymbol{q}_{n_\tau}^{\mathrm{nom}}$ and $\boldsymbol{v}_{n_\tau}^{\mathrm{nom}}$ are the nominal generalized positions and velocities of actuated joints, $\boldsymbol{K}_{\mathrm{p}}^{\mathrm{nom}}$ and $\boldsymbol{K}_{\mathrm{d}}^{\mathrm{nom}}$ are proportional and derivative gains. It is worth mentioning that these tasks are generally assigned the lowest priority.

3.3 Hierarchical Optimization

The general task Eq. (34) can be formulated as solving a QP problem:

$$\begin{aligned} \min_{\boldsymbol{z}, \boldsymbol{s}_{\mathrm{eq}}, \boldsymbol{s}_{\mathrm{in}}} \quad & \frac{1}{2}\|\boldsymbol{s}_{\mathrm{eq}}\|^2 + \frac{1}{2}\|\boldsymbol{s}_{\mathrm{in}}\|^2 \\ \text{s.t.} \quad & \boldsymbol{A}\boldsymbol{z} - \boldsymbol{b} = \boldsymbol{s}_{\mathrm{eq}} \\ & \boldsymbol{C}\boldsymbol{z} - \boldsymbol{d} \leqslant \boldsymbol{s}_{\mathrm{in}} \end{aligned} \quad . \tag{48}$$

The equivalent QP problem is

$$\begin{aligned} \min_{\boldsymbol{Z}} \quad & \frac{1}{2}\boldsymbol{Z}^{\mathrm{T}}\boldsymbol{P}\boldsymbol{Z} + \boldsymbol{c}^{\mathrm{T}}\boldsymbol{Z} \\ s.t. \quad & \bar{\boldsymbol{C}}\boldsymbol{Z} \leqslant \bar{\boldsymbol{d}} \end{aligned} , \tag{49}$$

where $\boldsymbol{Z} = \left[\boldsymbol{z}^{\mathrm{T}}\ \boldsymbol{s}_{\mathrm{in}}^{\mathrm{T}}\right]^{\mathrm{T}}$ is the equivalent optimization variable, the Hessian matrix $\boldsymbol{P}$ and the gradient vector $\boldsymbol{c}$ can be calculated from $\boldsymbol{P} = \bar{\boldsymbol{A}}^{\mathrm{T}}\bar{\boldsymbol{A}}$ and $\boldsymbol{c} = -\bar{\boldsymbol{A}}^{\mathrm{T}}\bar{\boldsymbol{b}}$. The expressions of the augmented equality constraint matrix $\bar{\boldsymbol{A}}$, augmented equality constraint vector $\bar{\boldsymbol{b}}$, augmented inequality constraint matrix $\bar{\boldsymbol{C}}$, augmented inequality constraint vector $\bar{\boldsymbol{d}}$ are

$$\bar{\boldsymbol{A}} = \begin{bmatrix} \boldsymbol{A} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{I} \end{bmatrix}, \bar{\boldsymbol{b}} = \begin{bmatrix} -\boldsymbol{b} \\ \boldsymbol{0} \end{bmatrix}, \bar{\boldsymbol{C}} = \begin{bmatrix} \boldsymbol{C} & -\boldsymbol{I} \end{bmatrix}, \bar{\boldsymbol{d}} = \boldsymbol{d}.$$

For a set of tasks with priorities $\mathbf{T}_1, \dots, \mathbf{T}_n$, they can be solved under strict prioritization. For the task with priority p, it is defined as:

$$\mathbf{T}_p : \begin{cases} \boldsymbol{A}_p\boldsymbol{z} - \boldsymbol{b}_p = \boldsymbol{s}_{\mathrm{eq},p} \\ \boldsymbol{C}_p\boldsymbol{z} - \boldsymbol{d}_p \leqslant \boldsymbol{s}_{\mathrm{in},p} \end{cases}. \tag{50}$$

The null space projector of this task is defined as:

$$\boldsymbol{N}_p = \mathrm{Null}(\boldsymbol{A}_p) = \boldsymbol{I} - \boldsymbol{A}_p^{\dagger}\boldsymbol{A}_p, \tag{51}$$

where $\{\cdot\}^{\dagger}$ is the singular value decomposition (SVD) based pseudo-inverse. The augmented task Jacobian $\bar{\boldsymbol{A}}_p$ is defined as:

$$\bar{\boldsymbol{A}}_p = \begin{bmatrix} \boldsymbol{A}_1^{\mathrm{T}} & \boldsymbol{A}_2^{\mathrm{T}} & \cdots & \boldsymbol{A}_p^{\mathrm{T}} \end{bmatrix}^{\mathrm{T}}. \tag{52}$$

The null space projector of the augmented task Jacobian $\bar{\boldsymbol{A}}_p$ is

$$\bar{\boldsymbol{N}}_p = \mathrm{Null}(\bar{\boldsymbol{A}}_p) = \boldsymbol{I} - \bar{\boldsymbol{A}}_p^{\dagger}\bar{\boldsymbol{A}}_p. \tag{53}$$

An efficient recursive implementation is

$$\bar{\boldsymbol{N}}_p = \bar{\boldsymbol{N}}_{p-1}\mathrm{Null}(\boldsymbol{A}_p\bar{\boldsymbol{N}}_{p-1}). \tag{54}$$

Considering tasks $\mathbf{T}_1, \mathbf{T}_2, \dots, \mathbf{T}_{p+1}$, the augmented optional solution is

$$\bar{\boldsymbol{z}}_{p+1}^* = \bar{\boldsymbol{N}}_p\boldsymbol{z}_{p+1}^* + \bar{\boldsymbol{z}}_p^*, \quad p = 1, \dots, n-1, \tag{55}$$

where $\bar{\boldsymbol{N}}_1 = \boldsymbol{N}_1$ and $\bar{\boldsymbol{z}}_1 = \boldsymbol{z}_1$. $\boldsymbol{z}_{p+1}^*$ is solved from the optimization problem:

$$\begin{aligned} \min_{\boldsymbol{z},\boldsymbol{s}_{\mathrm{in}}} \quad & \frac{1}{2}\|\boldsymbol{A}_{p+1}(\bar{\boldsymbol{N}}_p\boldsymbol{z} + \bar{\boldsymbol{z}}_p^*) - \boldsymbol{b}_{p+1}\|^2 + \frac{1}{2}\|\boldsymbol{s}_{\mathrm{in}}\|^2 \\ \text{s.t.} \quad & \boldsymbol{C}_{p+1}(\bar{\boldsymbol{N}}_p\boldsymbol{z} + \bar{\boldsymbol{z}}_p^*) - \boldsymbol{d}_{p+1} \leqslant \boldsymbol{s}_{\mathrm{in}} \\ & \boldsymbol{C}_p(\bar{\boldsymbol{N}}_p\boldsymbol{z} + \bar{\boldsymbol{z}}_p^*) - \boldsymbol{d}_p \leqslant \boldsymbol{s}_{\mathrm{in},p}^* \\ & \dots \\ & \boldsymbol{C}_2(\bar{\boldsymbol{N}}_p\boldsymbol{z} + \bar{\boldsymbol{z}}_p^*) - \boldsymbol{d}_2 \leqslant \boldsymbol{s}_{\mathrm{in},2}^* \\ & \boldsymbol{C}_1(\bar{\boldsymbol{N}}_p\boldsymbol{z} + \bar{\boldsymbol{z}}_p^*) - \boldsymbol{d}_1 \leqslant \boldsymbol{s}_{\mathrm{in},1}^* \end{aligned} . \tag{56}$$

After collecting $\boldsymbol{z}$, $\boldsymbol{s}_{\mathrm{in}}$ together $\boldsymbol{Z} = \begin{bmatrix} \boldsymbol{z}^{\mathrm{T}} & \boldsymbol{s}_{\mathrm{in}}^{\mathrm{T}} \end{bmatrix}^{\mathrm{T}}$ and stacking inequality constraints to make the notation shorter, similar to the process from Eq. (48) to Eq. (49), the QP problem Eq. (56) becomes:

$$\begin{aligned} \min_{\boldsymbol{Z}} \quad & \frac{1}{2}\boldsymbol{Z}^{\mathrm{T}}\boldsymbol{P}\boldsymbol{Z} + \boldsymbol{c}^{\mathrm{T}}\boldsymbol{Z} \\ \text{s.t.} \quad & \begin{bmatrix} \boldsymbol{C}_{p+1}\bar{\boldsymbol{N}}_p & -\boldsymbol{I} \\ \bar{\boldsymbol{C}}_p\bar{\boldsymbol{N}}_p & \boldsymbol{0} \end{bmatrix}\boldsymbol{Z} \leqslant \begin{bmatrix} \boldsymbol{d}_{p+1} - \boldsymbol{C}_{p+1}\bar{\boldsymbol{z}}_p^* \\ \bar{\boldsymbol{d}}_p - \bar{\boldsymbol{C}}_p\bar{\boldsymbol{z}}_p^* + \bar{\boldsymbol{s}}_{\mathrm{in},p}^* \end{bmatrix}, \end{aligned} \tag{57}$$

where the Hessian matrix $\boldsymbol{P}$ and the gradient $\boldsymbol{c}$ can be calculated from $\boldsymbol{P} = \bar{\boldsymbol{A}}^{\mathrm{T}}\bar{\boldsymbol{A}}$ and $\boldsymbol{c} = -\bar{\boldsymbol{A}}^{\mathrm{T}}\bar{\boldsymbol{b}}$ with:

$$\bar{\boldsymbol{A}} = \begin{bmatrix} \boldsymbol{A}_{p+1}\bar{\boldsymbol{N}}_p & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{I} \end{bmatrix}, \bar{\boldsymbol{b}} = \begin{bmatrix} -(\boldsymbol{A}_{p+1}\bar{\boldsymbol{z}}_p^* - \boldsymbol{b}_{p+1}) \\ \boldsymbol{0} \end{bmatrix}. \tag{58}$$

3.4 Hybrid Joint Controller

After solving a set of QP problems, the optimal generalized accelerations $\dot{\boldsymbol{v}}^*$ can be obtained from Eqs. (56) and (33), and the optimal joint torques $\boldsymbol{\tau}^*$ can be calculated with Eq. (32). The desired joint velocity $\boldsymbol{v}_j^{\mathrm{des}}$ and position $\boldsymbol{q}_j^{\mathrm{des}}$ can be computed by adding the joint position and velocity portions to the measured joint velocity $\boldsymbol{v}_j$ and position $\boldsymbol{q}_j$ as

$$\boldsymbol{v}_j^{\mathrm{des}} = \boldsymbol{v}_j + \dot{\boldsymbol{v}}^*\Delta t, \tag{59}$$

$$\boldsymbol{q}_j^{\mathrm{des}} = \boldsymbol{q}_j + \boldsymbol{v}_j\Delta t + \frac{1}{2}\dot{\boldsymbol{v}}^*\Delta t^2. \tag{60}$$

The joint action $\left(\boldsymbol{q}_j^{\mathrm{des}}, K_{\mathrm{p},j}, \boldsymbol{v}_j^{\mathrm{des}}, K_{\mathrm{d},j}, \boldsymbol{\tau}^*\right)$ is sent the hybrid joint controller to compute motor torque commands as

$$\boldsymbol{\tau} = K_{\mathrm{p},j}\left(\boldsymbol{q}_j^{\mathrm{des}} - \boldsymbol{q}_j\right) + K_{\mathrm{d},j}\left(\dot{\boldsymbol{q}}_j^{\mathrm{des}} - \dot{\boldsymbol{q}}_j\right) + \boldsymbol{\tau}^*. \tag{61}$$

4 Experiment

The proposed planning and control framework is validated through experiments on a quadrupedal robot *A1* developed by Unitree Robotics. The robot has 18 degrees of freedom, a height of 0.4 m, and a weight of 14 kg. Four experiments were performed, namely body twisting, sharp turning, multi-gaited locomotion, and disturbance response. A video recording of the experiments is supported by the video submission.

The whole framework including gait generator, motion planner, whole-body controller and state estimator is executed on a dedicated computer with a frequency of 500 Hz. For modelling of kinematics and dynamics, the open-source library Pinocchio [39] is used, which is a C++ implementation of the modern rigid body algorithms for multibody systems [40]. To solve each QP problem numerically, a numerical optimization package OSQP [41] and its C++ interface OSQP-Eigen are used.

Table 1 Body Properties

Parameter	Symbol	Value	Units
Body mass	m	13.68	kg
Body inertia	I_{xx}	0.17	kg m^2
	I_{yy}	0.57	kg m^2
	I_{zz}	0.64	kg m^2

Table 2 Feedback gains

Parameter	Symbol	Value
Position gains	$\boldsymbol{K}_\mathrm{p}$	[1. 1. 420. 20. 10. 1.]
Velocity gains	$\boldsymbol{K}_\mathrm{d}$	[30. 30. 10. 0.05 0.05 15.]

Table 3 Task hierarchy (priority 0 is the highest)

Priority	Tasks	Weight
0	Dynamic consistency	1
0	Torque saturation limits	1
0	Contact force limits	1
1	Contact motion task	1
2	CoM position control task	1
2	Base orientation control task	0.50
2	Swing foot position control task	0.75
2	Contact force regularization task	0.15
3	Minimum motion task	0.20

All experiments use the same parameters, feedback gains, and task arrangements. The equivalent body property in Eq. (23) and feedback gains in Eq. (27) used in the motion planning module are collected in Tables 1 and 2, respectively. The task hierarchy used in the whole-body control module is summarized in Table 3. The hierarchy first maintains the stability of the robot and then plans movements to achieve task requirements. In detail, satisfying the dynamic constraints and hardware limitations has the highest priority. Then, satisfying the task enforces kinematic contact constraints as a second priority. After that, tasks tracking task-space motion and force references have the third priority. Last, the task of avoiding internal drift and keeping a better-looking posture has the lowest priority.

4.1 Body Twisting

To examine the body orientation tracking performance of the proposed framework, the body twisting experiment is conducted. Figure 7 presents video snapshots of the robot performing a body twisting motion. The robot is commanded to rotate its body in x-, y- and z-directions respectively while standing with all feet and its CoM position unchanged. The operator sends angular velocity commands, and then the motion planner calculates the reference body velocity and pose trajectories, which are tracked by a whole-body controller with equations from (6) to (9).

Figure 8 shows the body orientation reference tracking data. Comparing Fig. 8a, c, e, it can be found that

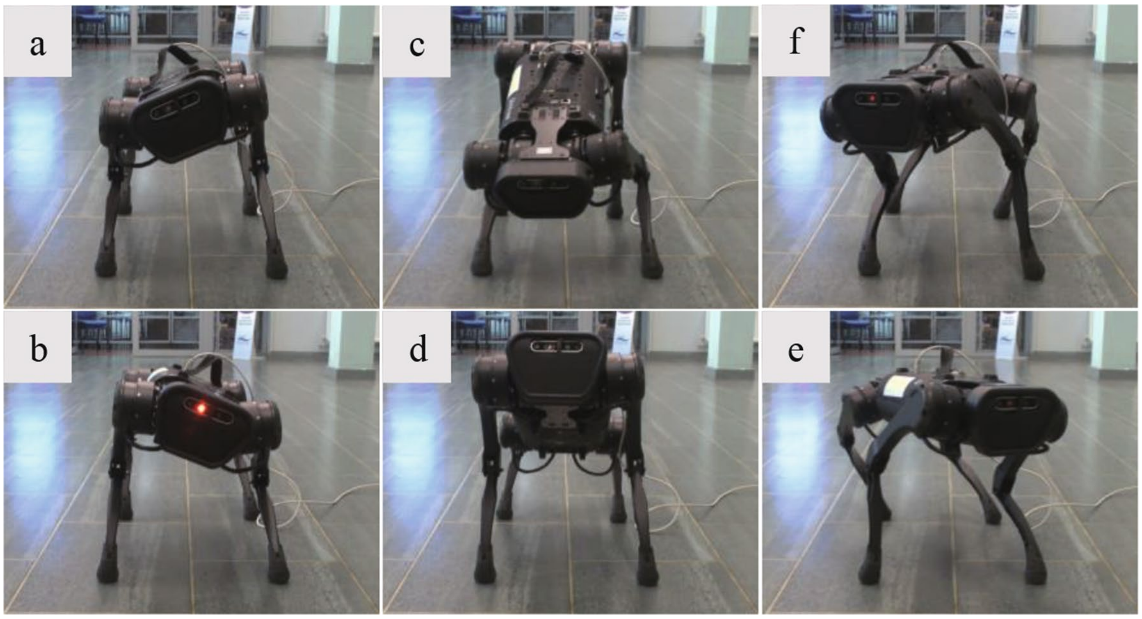


Fig. 7 Snapshots of the robot twisting its body. Pictures **a** and **b** show the robot twisting around the x-axis, pictures **c** and **d** show the robot twisting around the y-axis, and pictures **e** and **f** show the robot twisting around the z-axis

tracking errors in the x-direction are larger than those in the other two directions due to the asymmetry of the robot in the three directions. Also, the robot's internal frictional resistance in the x-direction is greater, making it more difficult to control. From Fig. 8b, d, f, it can be seen that the angular velocity tracking is more accurate, even when the angular velocity changes rapidly from -2 rad/s to +2 rad/s within 1 s.

The current maximum angle tracking error is about 0.12 rad, which causes by the robot prioritising the stability and limiting the *Base orientation control task*. It can be improved by adjusting the weight of the *Base orientation control task* in Table 3 and feedback gains $\boldsymbol{K}_{\mathrm{p}}^{\mathrm{ori}}$ and $\boldsymbol{K}_{\mathrm{d}}^{\mathrm{ori}}$ in Eq. (46). However, we believe that prioritising the stability of the robot itself is more important in a mission. It is not cost-effective to sacrifice stability and safety for better performance. Also, although the base orientations are presented as Euler angles for intuitive understanding, they are presented as rotation matrices in the planning and control processes, which won't cause singularity and ambiguity issues.

4.2 Sharp Turning

A sharp turning experiment demonstrates the framework's ability to make the robot turn at a high angular velocity, which is essential when the robot traverses confined and crowded environments. Figure 9 presents video snapshots of the robot performing a sharp turning motion. In the experiment, the robot executes manoeuvres including moving forward, performing an S-Turn, and then moving straight again. During the whole process, the robot runs at a constant forward velocity of 0.8 m/s in the trot-run gait.

Figure 10 presents the sharp turning experiment data. Figure 10a shows a comparison between the commanded yaw velocity and the actual yaw velocity, Fig. 10b shows the reference yaw angle generated by the motion planner and the actual yaw angle extracted by the state estimator. As can be seen from these two sub-pictures, the reference is nicely tracked and the robot can still withstand

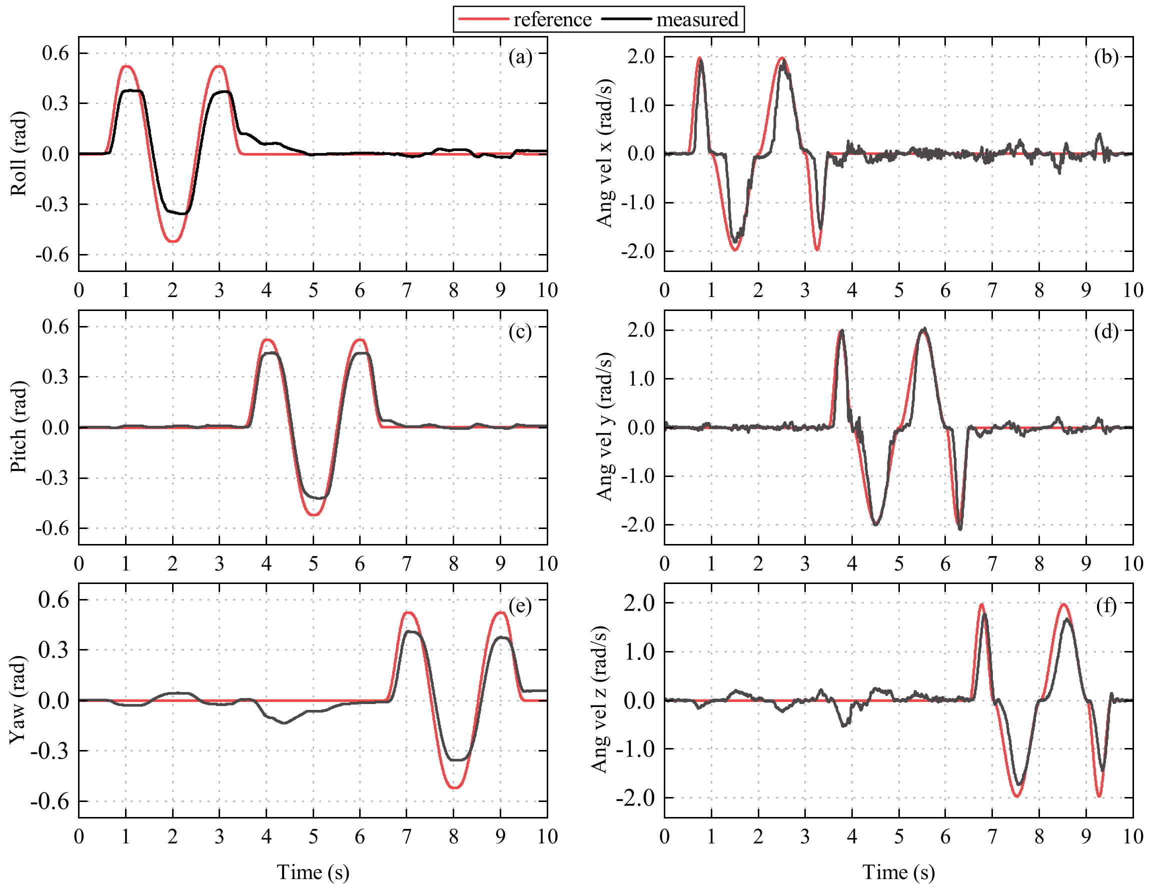


Fig. 8 Base orientation and angular velocity tracking performance. Graphs **a**, **c** and **e** are orientation tracking in the x-, y-, and z-direction. Graphs **b**, **d** and **f** are angular velocity tracking and these velocities are expressed in the body coordinate. The reference is generated by the motion planner, and the measured data is from the state estimator

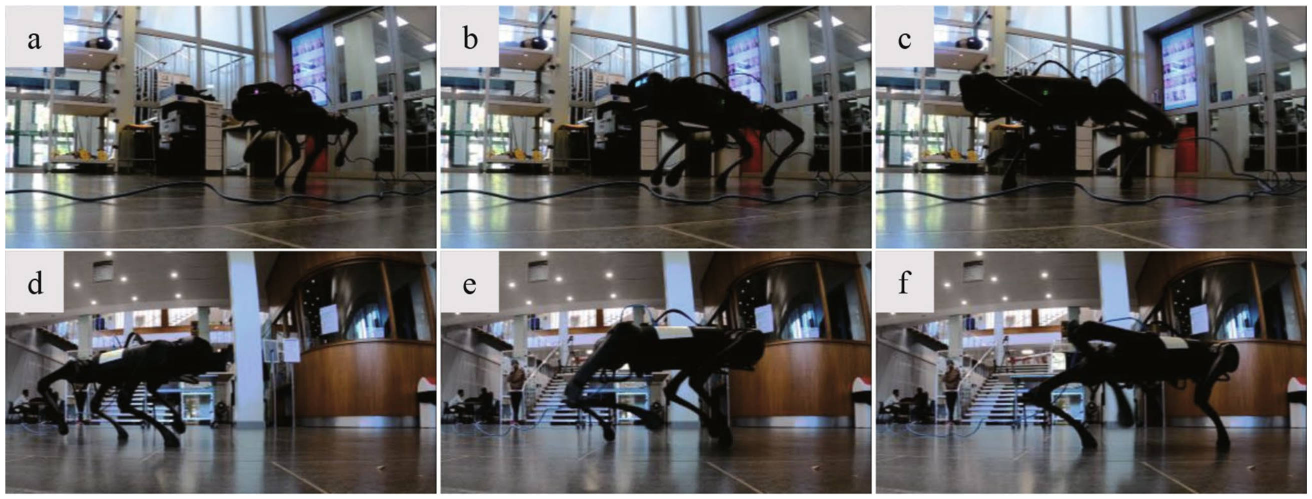


Fig. 9 Snapshots of the *A1* robot making sharp turns while running. Pictures **a**, **b**, and **c** show the robot turning clockwise, and pictures **d**, **e**, and **f** show the robot turning counterclockwise

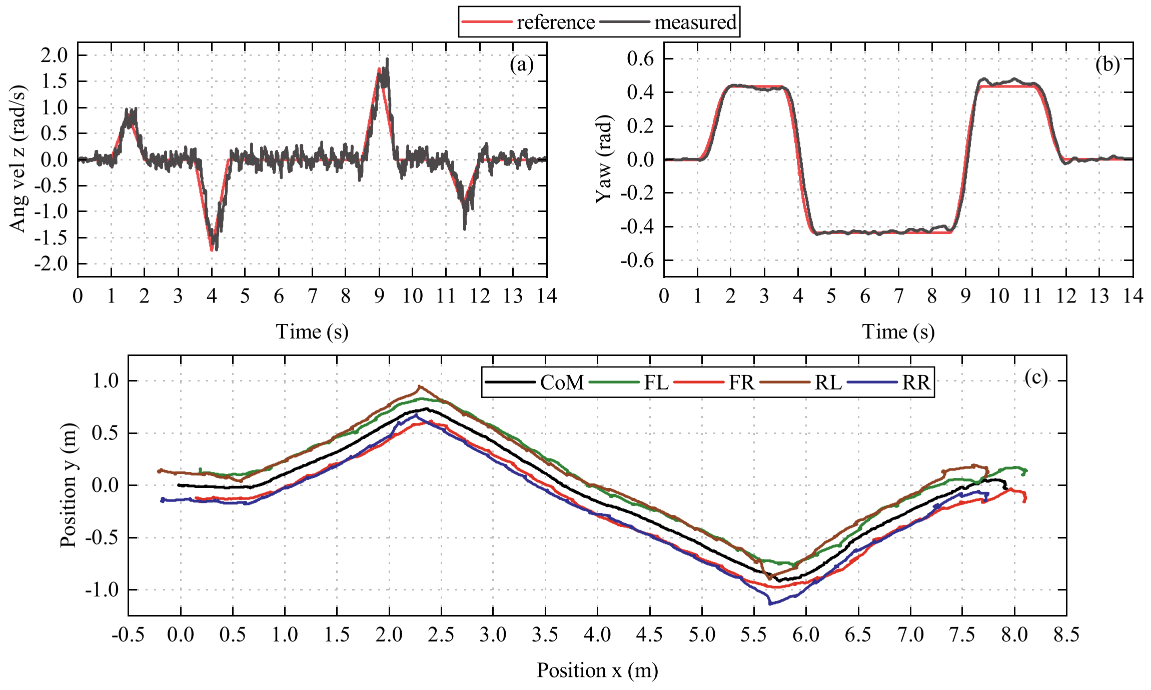


Fig. 10 Sharp turning experiment data. **a**, **b** Show the performance of angular velocity and angle in the yaw direction. **c** shows the actual CoM and foot trajectories

an instantaneous angular velocity of up to 1.8 rad/s when running fast. Figure 10c shows the top view of the motion in the ground plane, including the x–y positions of CoM and all feet extracted from the state estimator.

4.3 Multi-gaited Locomotion

To demonstrate that the planning and control framework can stabilize basic locomotion gaits, the multi-gaited locomotion

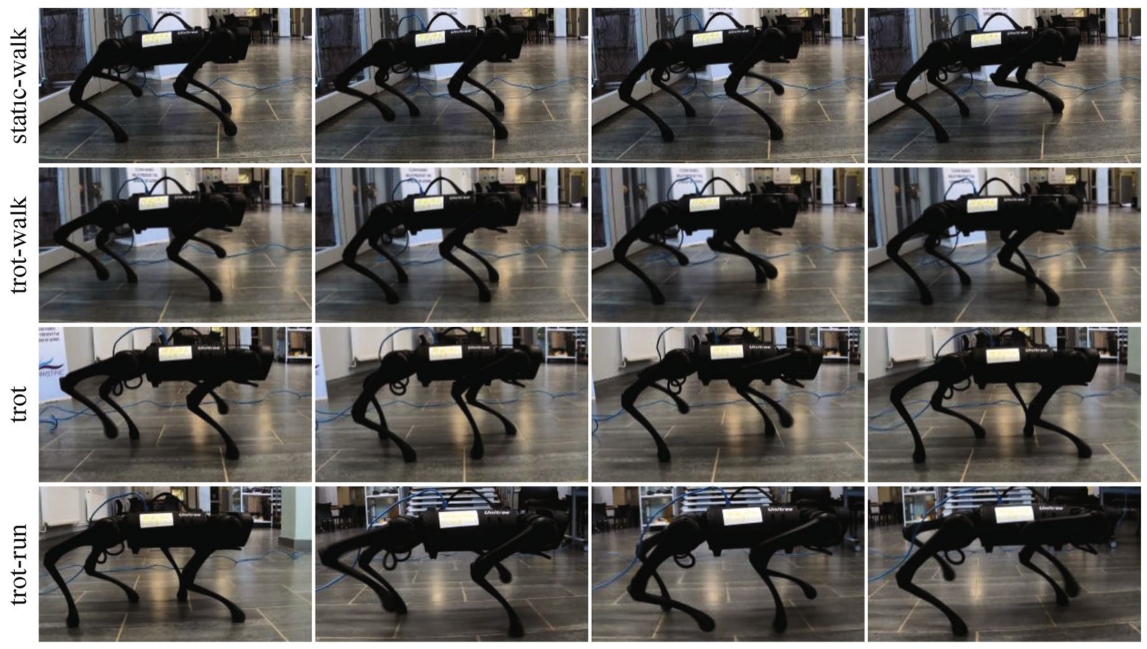


Fig. 11 Snapshots of the robot running with multiple gaits. From top to bottom, the speed of each gait increases respectively

experiment is presented. A set of gaits including static-walk, trot-walk, trot and trot-run, are tested in this experiment. Figure 11 shows the snapshots of all gaits during the experiment. The description of the gaits are shown below:

Static-Walk only one leg is in the swing, and the robot is statically stable at all times.
Trot-Walk the robot lifts diagonal legs at the same time at low speed, and exist time all four feet make contact with the ground.
Trot the robot lifts diagonal legs at the same time at a higher speed, and only two legs make contact with the ground at any time.
Trot-Run the robot lifts diagonal legs at the same time, and exist time when all four feet are in the air.

The difference between trot-walk, trot and trot-run gaits is the longer moment of suspension between each beat.

From the top and middle subplots in Fig. 12, different gaits are selected as the speed changes. This gait selection is consistent with the gait-changing behaviour of quadruped animals. From the bottom subplot in Fig. 12, the height trajectory shows a more dramatic height change as the speed of the gait increase because the longer aerial phase of all feet makes it harder to maintain the height of the body. For the same reason, in the trot-run gait, the aerial phase becomes the longest, where exists time none of the feet is making contact with the ground. This makes the robot unable to maintain body height and causes a lower mean value in the graph.

Figure 13 shows the trajectory of the front left foot in x–z plane. From the left subplot, the step length increases and then decreases as its speed increases and decreases, which is enforced by the rule illustrated in Eq. (10). The right subplot shows the process of lifting foot height illustrated in Eqs. (15) and (16).

This experiment shows that the proposed framework is independent of gait type, and can be applied to different gaits. It also shows the stability of the framework between gait changes.

4.4 Disturbance Response

To evaluate the disturbance rejection capability of the control framework, robot *A1* gets kicked from the side when it is trotting in place. Figure 14 presents the snapshots of this experiment. It can be seen that the robot can still remain balanced after strong kicks.

Figure 15 shows the changes in CoM velocity and position over time in the presence of external disturbance. These disturbance forces cause a sudden change in the CoM velocity, especially in the y direction, and the increase in y direction velocity results in a side-stepping motion. Since the *CoM position control task* penalizes deviations from the desired and actual CoM positions, the robot will walk back to its original standing position.

Figure 16 presents the y reaction forces and the y motion for a specific foot (front-left). Two recovery strategies are used when the robot reacts to the unpredictable external disturbance resulting from the first kicking. As

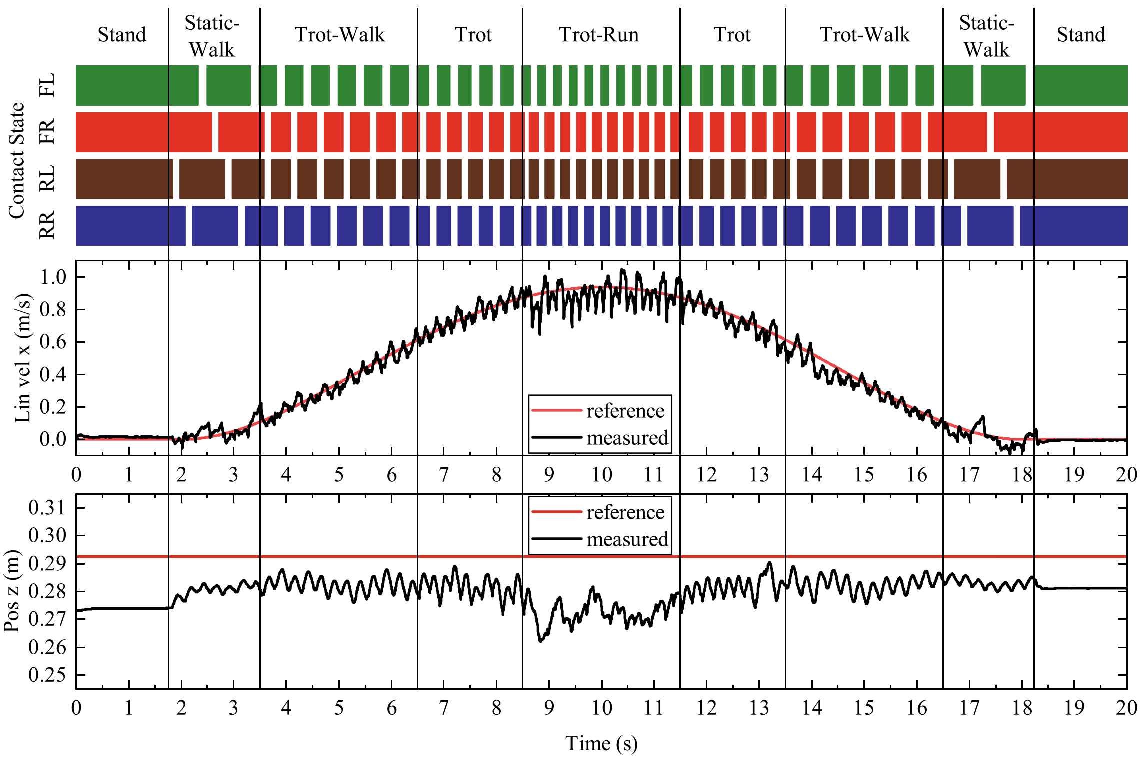


Fig. 12 Graph showing the gait type, contact state, velocity and height of the real robot during the multi-gaited locomotion experiment. In the top plot, a coloured region indicates when the foot is in stance phase, while a white space shows that it is in the swing phase. The middle and bottom plots compare the reference values generated by the motion planner and the actual values extracted from the state estimator

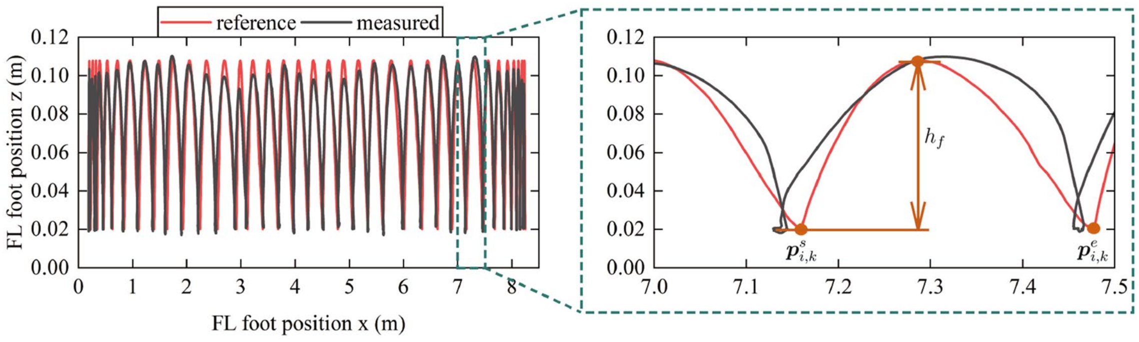


Fig. 13 Trajectory of the front left foot. The left subplot shows the whole trajectory during the experiment, and the right subplot is a zoomed-in view showing the trajectory of the foot within a gait cycle. h_f indicated the commanded height of the foot. The red line indicates the reference trajectory planned by the motion planner, and the black line indicates the actual trajectory estimated by a state estimator

shown in Fig. 16a, in the stance phase, the foot adjusts the ground reaction forces to resist the disturbance while satisfying in the constraint of the friction cone. This is realized by optimizing the desired contact forces in the motion planning module illustrated in Sect. 2.4, and tracking these forces via the *Contact force regularization task* in the whole-body control module. As shown in Fig. 16b, in the swing phase, it reacts to the external disturbance through a side-stepping motion which is realized by adjusting the foothold position according to the actual CoM velocity illustrated in Sect. 2.3. The difference between the reaction force generated by the motion planner and the whole body controller is due to the modelling of the robot being different, where the motion planner uses the 3D-SRBM and the whole body controller uses full-body dynamic model. For the same reason, the foot position planned by the motion

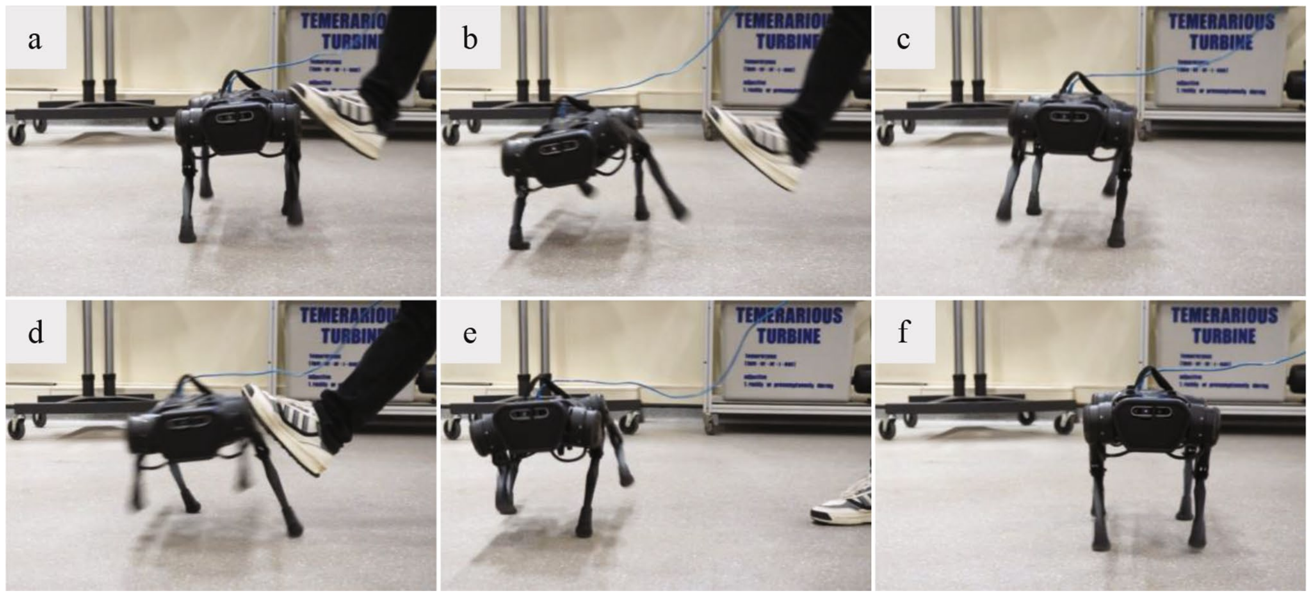


Fig. 14 Disturbance response experiment. Picture **a**–**c** shows first trial, and picture **d**–**f** shows second trial. Lateral disturbances are inserted by strongly kicking the robot twice from its left during trotting. In order to keep its balance, the robot step to the side. After the robot regain balance, it moves back to its initial position eventually

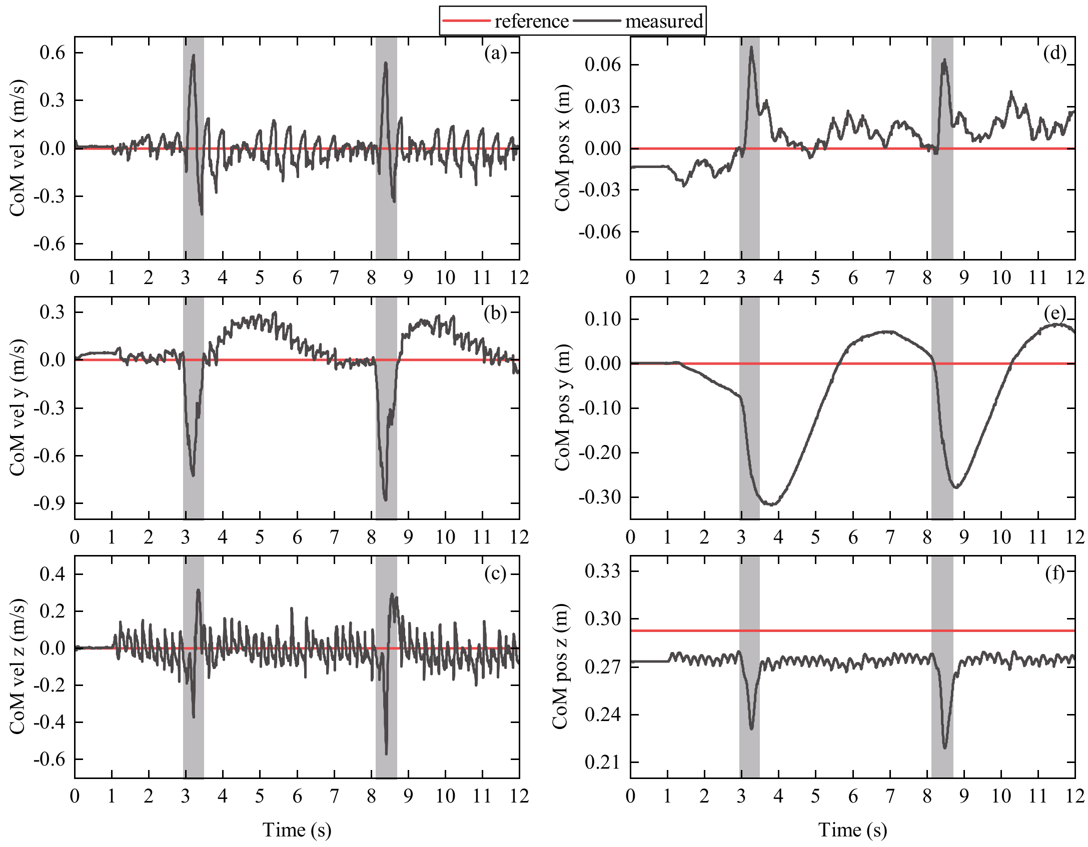


Fig. 15 CoM velocity and position. The gray area shows the trajectories at the time around a disturbance, where the disturbance force results in a sudden change in the CoM velocity. The red line shows the reference velocity and position, while the black line shows the actual velocity and position

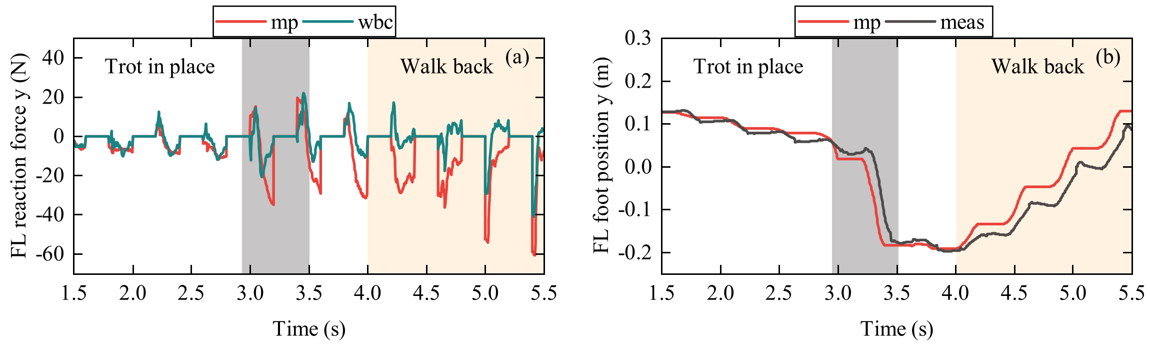


Fig. 16 The front-left foot's contact force and position in the *y* direction, during the first kicking and its recovery period. The gray area indicates the process of kicking, and the orange area shows the process of walking back to its initial position. The red line shows the reference contact force and foot position from the motion planner, while the cyan line shows the reference force from the whole-body controller, and the black line shows actual position from the state estimator

planner is different from the one planned by the whole-body controller.

5 Conclusion

A real-time planning and control framework is proposed and deployed in a real robot, which enables the robot to perform various dynamic locomotion and be robust to external disturbance. The existing methods using the same model in the planning and control phases, where simplified models can not ensure physical consistency and full-order models make computationally expensive. To take advantage of both methods, this research employs both the 3D-SRBM and the full-body dynamic model to build a more comprehensive framework suitable for different tasks. The proposed control framework does not have a limitation in gait types and gait transitions. The experiment result proved the practicability of quadruped robots with the proposed framework in open environments and under disturbances. The above advantages are very beneficial to deploy quadruped robots in complex real environment to complete tasks. The immediate next step will be focused on extending the framework to generate base and foot trajectories that guide the motion of the robot over obstacles with a sequence of suitable footholds. To fully explore coupling effects between the base and limbs and provide sufficient flexibility to generate more complex whole-body motions, future work will replace the 3D-SRBM used in the motion planner with a mixed model combining the centroidal dynamics model and the whole-body kinematics model.

Supplementary Information The online version contains supplementary material available at https://doi.org/10.1007/s42235-023-00347-9.

Author Contributions All authors contributed to the study's conception and design. Material preparation, data collection and analysis were performed by JL, YW and CZ. The first draft of the manuscript was written by JL and YW, and all authors commented on previous versions of the manuscript. The study was performed under the instruction of HG, HY and CZ. All authors read and approved the final manuscript.

Funding This work is supported by the China Scholarship Council under Grant [2020]06120186, the Engineering and Physical Sciences Research Council [Grant number EP/V026801/2], the Advanced Machinery and Productivity Institute [Innovate UK project number 84646], and the National Natural Science Foundation of China under Grant 52175011.

Data accessibility The datasets generated during and/or analysed during the current study are available from the corresponding author on reasonable request.

Declarations

Conflict of interest The authors have no relevant financial or non-financial interests to disclose.

Ethical approval No approval of research ethics committees was required to accomplish the goals of this study because experimental work was conducted with an unregulated invertebrate species.

References

1. Zhou, C. X., Wang, X., Li, Z. B., & Tsagarakis, N. (2017). Overview of gait synthesis for the humanoid COMAN. *Journal of Bionic Engineering, 14*(1), 15–25.

2. Kuindersma, S. (2020). Recent progress on Atlas, the world's most dynamic humanoid robot. https://youtu.be/EGABAx52GKI.
3. Chignoli, M., Kim, D., Stanger-Jones, E., & Kim, S. (2021). The MIT humanoid robot: Design, motion planning, and control for acrobatic behaviors. In Proceedings of the IEEE-RAS International Conference on Humanoid Robots, Munich, Germany, 2021, pp. 1–8.
4. Duan, H., Dao, J., Green, K., Apgar, T., Fern, A., & Hurst, J. (2021). Learning task space actions for bipedal locomotion. In Proceedings of the IEEE International Conference on Robotics and Automation, Xi'an, China, 2021, pp. 1276–1282.
5. Dafarra, S., Romualdi, G., & Pucci, D. (2022). Dynamic complementarity conditions and whole-body trajectory optimization for humanoid robot locomotion. *IEEE Transactions on Robotics, 38*(6), 3414–3433.
6. Carlo, J. D., Wensing, P. M., Katz, B., Bledt, G., & Kim, S. (2018). Dynamic locomotion in the MIT Cheetah 3 through convex model predictive control. In Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems, Madrid, Spain, 2018, pp. 1–9.
7. Neunert, M., Stauble, M., Giftthaler, M., Bellicoso, C. D., Carius, J., Gehring, C., Hutter, M., & Buchli, J. (2018). Whole-body nonlinear model predictive control through contacts for quadrupeds. *IEEE Robotics and Automation Letters, 3*(3), 1458–1465.
8. Xin, G., Xin, S., Cebe, O., Pollayil, M. J., Angelini, F., Garabini, M., Vijayakumar, S., & Mistry, M. (2021). Robust footstep planning and lqr control for dynamic quadrupedal locomotion. *IEEE Robotics and Automation Letters, 6*(3), 4488–4495.
9. Bledt, G., Wensing, P. M., & Kim, S. (2017). Policy regularized model predictive control to stabilize diverse quadrupedal gaits for the MIT Cheetah. In Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems, Vancouver, Canada, 2017, pp. 4102—4109.
10. Bledt, G., & Kim, S. (2020). Extracting legged locomotion heuristics with regularized predictive control. In Proceedings of the IEEE International Conference on Robotics and Automation, Paris, France, 2020, pp. 406–412.
11. Ding, Y., Pandala, A., Li, C., Shin, Y.-H., & Park, H.-W. (2021). Representation-free model predictive control for dynamic motions in quadrupeds. *IEEE Transactions on Robotics, 37*(4), 1154–1171.
12. Hwangbo, J., Lee, J., Dosovitskiy, A., Bellicoso, D., Tsounis, V., Koltun, V., & Hutter, M. (2019). Learning agile and dynamic motor skills for legged robots. *Science Robotics, 4*(26), eaau5872.
13. Lee, J., Hwangbo, J., & Hutter, M. (2020). Learning quadrupedal locomotion over challenging terrain. *Science Robotics, 5*(47), eabc5986.
14. Zhou, L. L., Li, T. F., Liu, Z. Y., & Li, Y. B. (2021). An efficient gait-generating method for electrical quadruped robot based on humanoid power planning approach. *Journal of Bionic Engineering, 18*, 1463–1474.
15. Kim, Y. K., Seol, W., & Park, J. (2021). Biomimetic quadruped robot with a spinal joint and optimal spinal motion via reinforcement learning. *Journal of Bionic Engineering, 18*, 1280–1290.
16. Posa, M., Cantu, C., & Tedrake, R. (2014). A direct method for trajectory optimization of rigid bodies through contact. *International Journal of Robotics Research, 33*(1), 69–81.
17. Herzog, A., Rotella, N., Schaal, S., & Righetti, L. (2015). Trajectory generation for multi-contact momentum control. In Proceedings of the IEEE-RAS International Conference on Humanoid Robots, Seoul, Korea (South), 2015, pp. 874–880.
18. Kuindersma, S., Deits, R., Fallon, M., Valenzuela, A., Dai, H., Permenter, F., Koolen, T., Marion, P., & Tedrake, R. (2016). Optimization based locomotion planning, estimation, and control design for the atlas humanoid robot. *Autonomous Robots, 40*, 429–455.
19. Budhiraja, R., Carpentier, J., & Mansard, N. (2019). Dynamics consensus between centroidal and whole-body models for locomotion of legged robots. In Proceedings of the IEEE International Conference on Robotics and Automation, Montreal, Canada, 2019, pp. 6727–6733.
20. 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. Preprint at arXiv:1909.06586.
21. Shuster, M. D. (1993). A survey of attitude representations. *Journal of the Aeronautical Sciences, 41*, 439–517.
22. Bhat, S. P., & Bernstein, D. S. (1998). A topological obstruction to global asymptotic stabilization of rotational motion and the unwinding phenomenon. In Proceedings of the American Control Conference, Philadelphia, USA, 1998, pp. 2785–2789.
23. Farshidian, F., Jelavic, E., Satapathy, A., Giftthaler, M., & Buchli, J. Real-time motion planning of legged robots: A model predictive control approach. In Proceedings of the IEEE-RAS International Conference on Humanoid Robotics, Birmingham, UK, 2017, pp. 577–584.
24. Katayama, S., & Ohtsuka, T. (2022). Whole-body model predictive control with rigid contacts via online switching time optimization. In Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems, Kyoto, Japan, 2022, pp. 8858–8865.
25. Yu, H. T., Gao, H. B., & Deng, Z. Q. (2021). Toward a unified approximate analytical representation for spatially running spring-loaded inverted pendulum model. *IEEE Transactions on Robotics, 37*(2), 691–698.
26. Tassa, Y., Erez, T., & Todorov, E. (2012). Synthesis and stabilization of complex behaviors through online trajectory optimization. In Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems, Vilamoura-Algarve, Portugal, 2012, pp. 4906–4913.
27. Herzog, A., Rotella, N., Mason, S., Grimminger, F., Schaal, S., & Righetti, L. (2016). Momentum control with hierarchical inverse dynamics on a torque-controlled humanoid. *Autonomous Robots, 40*, 473–491.
28. Zhou, C. X., Fang, C., Wang, X., Li, Z. B., & Tsagarakis, N. (2016). A generic optimization-based framework for reactive collision avoidance in bipedal locomotion. In Proceedings of the IEEE International Conference on Automation Science and Engineering, Fort Worth, USA, 2016, pp. 1026–1033.
29. Xing, H. J., Ding, L., Gao, H. B., Li, W. H., & Tavakoli, M. (2022). Dual-user haptic teleoperation of complementary motions of a redundant wheeled mobile manipulator considering task priority. *IEEE Transactions on Systems, Man, and Cybernetics: Systems, 52*(10), 6283–6295.
30. Li, J., Gao, H. B., Wan, Y. H., Humphreys, J., Peers, C., Yu, H. T., & Zhou, C. X. (2022). Whole-body control for a torque-controlled legged mobile manipulator. *Actuators, 11*(11), 304.
31. Yang, Y. X., Zhang, T. N., Coumans, E., Tan, J., & Boots, B. (2021). Fast and efficient locomotion via learned gait transitions. In Proceedings of the Annual Conference on Robot Learning, London, UK, 2021, pp. 773–783.
32. Humphreys, J., Li, J., Gao, H. B., & Zhou, C. X. (2022). Bio-inspired gait transitions for quadruped locomotion. (**Under review**).
33. Bloesch, M., Hutter, M., Hoepflinger, M., Leutenegger, S., Gehring, C., Remy, C.D., & Siegwart, R. (2012). State estimation for legged robots—consistent fusion of leg kinematics and IMU. In Proceedings of the Robotics: Science and Systems, Sydney, Australia, 2012.
34. Xinjilefu, X., Feng, S., & Atkeson, C. G. (2014). Dynamic state estimation using quadratic programming. In Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems, Chicago, USA, 2014, pp. 989–994.

35. Raghavan, V. S., Kanoulas, D., Zhou, C. X., Caldwell, D. G. & Tsagarakis, N. G. (2018). A study on low-drift state estimation for humanoid locomotion, using lidar and kinematic-inertial data fusion. In Proceedings of the IEEE-RAS International Conference on Humanoid Robots, Beijing, China, 2018, pp. 1–8.
36. Hartley, R., Ghaffari, M., Eustice, R. M., & Grizzle, J. W. (2020). Contact-aided invariant extended Kalman filtering for robot state estimation. *International Journal of Robotics Research, 39*, 402–430.
37. Raibert, M. H. (1986). *Legged robots that balance*. MIT Press.
38. Pratt, J., Carff, J., Drakunov, S., & Goswami, A. (2006). Capture point: A step toward humanoid push recovery. In Proceedings of the IEEE-RAS International Conference on Humanoid Robots, Genova, Italy, 2006, pp. 200–207.
39. Carpentier, J., Saurel, G., Buondonno, G., Mirabel, J., Lamiraux, F., Stasse, O., & Mansard, N. (2019). The Pinocchio C++ library: A fast and flexible implementation of rigid body dynamics algorithms and their analytical derivatives. In Proceedings of the IEEE/SICE International Symposium on System Integration, Paris, France, 2019, pp. 614–619.
40. Featherstone, R. (2008). *Rigid body dynamics algorithms*. Springer.
41. Stellato, B., Banjac, G., Goulart, P., Bemporad, A., & Boyd, S. (2020). OSQP: An operator splitting solver for quadratic programs. *Mathematical Programming Computation, 12*, 637–672.