Unified Motion Planner for Walking, Running, and Jumping Using the Three-Dimensional Divergent Component of Motion

George Mesesan, *Member, IEEE*, Robert Schuller, *Member, IEEE*, Johannes Englsberger, *Senior Member, IEEE*, Christian Ott, *Fellow, IEEE*, and Alin Albu-Schäffer, *Fellow, IEEE*

***Abstract*—Running and jumping are locomotion modes that allow legged robots to rapidly traverse great distances and overcome difficult terrain. In this article, we show that the 3-D divergent component of motion (3D-DCM) framework, which was successfully used for generating walking trajectories in previous works, retains its validity and coherence during flight phases, and, therefore, can be used for planning running and jumping motions. We propose a highly efficient motion planner that generates stable center-of-mass (CoM) trajectories for running and jumping with arbitrary contact sequences and time parametrizations. The proposed planner constructs the complete motion plan as a sequence of motion phases that can be of different types: stance, flight, transition phases, etc. We introduce a unified formulation of the CoM and DCM waypoints at the start and end of each motion phase, which makes the framework extensible and enables the efficient waypoint computation in matrix and algorithmic form. The feasibility of the generated reference trajectories is demonstrated by extensive whole-body simulations with the humanoid robot TORO.**

***Index Terms*—Bipedal locomotion, divergent component of motion (DCM), gait generation, gait transitions, jumping, running.**

I. Introduction

BIPEDALISM, as a form of locomotion, is commonly associated with four basic modes of motion:

1) *Standing:* Characterized by actively maintaining balance with one or both legs in contact with the ground.
2) *Walking:* Alternating single and double support phases, switching the stance leg at each step. At least one leg is always in contact with the ground.
3) *Running:* Alternating stance and flight phases, switching the stance leg at each step.
4) *Jumping:* Alternating stance and flight phases, using both legs simultaneously during the stance phases.

Humans are particularly well adapted to perform these modes of motion, and humanoid robots are expected to achieve similar performance or even outperform humans on common locomotion tasks such as walking or running. Early results of Takenaka et al. [1] showed the humanoid robot Asimo running at 10 km/h, Kojima et al. [2] demonstrated a 30 cm high jumping motion with 0.5 s flight time performed by the humanoid robot JAXON3-P, while more recent videos from Boston Dynamics highlight their robot's capabilities of traversing parkour-like environments and performing complex somersaults [3]. However, despite these successes, significant research effort is still needed in the field of bipedal locomotion, in particular regarding the online planning and execution of highly dynamic motions such as running and jumping, traversing uneven and rough terrain, or following a sequence of arbitrarily placed stepping stones.

A. Related Work

In addition to the increasingly popular reinforcement learning (RL) approaches [4], [5], a widely used method of implementing robotic bipedal locomotion is to focus on the center-of-mass (CoM) dynamics. As observed in biological systems [6], the CoM dynamics covers the most important aspects of locomotion. Especially during motion planning, the whole-body dynamics is often replaced by heuristics [7], simplified models such as the linear inverted pendulum (LIP) [8] or the spring-loaded inverted pendulum (SLIP) [9], and reduced models such as the 3-D divergent component of motion (3D-DCM) [10]. Of these, the SLIP model, grounded in biomechanical studies of human running [11], has been widely used in generating running motions for legged robots [12], [13] and bipedal hopping behavior [14], [15]. However, the SLIP model dynamics is nonlinear and cannot be analytically integrated to produce closed-form solutions, and, therefore, it is poorly suited for generating running trajectories for aperiodic locomotion with arbitrary footstep placement and

Manuscript received 4 August 2023; accepted 17 September 2023. Date of publication 19 October 2023; date of current version 6 December 2023. This paper was recommended for publication by Associate Editor Michael Posa and Editor Sven Behnke upon evaluation of the reviewers' comments. This work was supported by the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme under Grant 819358. *(Corresponding author: George Mesesan.)*

George Mesesan, Robert Schuller, Johannes Englsberger, and Alin Albu-Schäffer are with the Institute of Robotics and Mechatronics, German Aerospace Center (DLR), 82234 Wessling, Germany (e-mail: george.mesesan@dlr.de; robert.schuller@dlr.de; johannes.englsberger@dlr.de; alin.albu-schaeffer@dlr.de).

Christian Ott is with the Institute of Robotics and Mechatronics, German Aerospace Center (DLR), 82234 Wessling, Germany, and also with the Automation and Control Institute, Faculty of Electrical Engineering and Information Technology, TU Wien, A-1040 Vienna, Austria (e-mail: christian.ott@tuwien.ac.at).

This article has supplementary material provided by the authors and color versions of one or more figures available at https://doi.org/10.1109/TRO.2023.3321396.

Digital Object Identifier 10.1109/TRO.2023.3321396

IEEE Transactions on Robotics (T-RO) paper, presented at ICRA 2024, Yokohama, Japan. Cite as T-RO paper.

time parametrization. One method addressing these limitations was proposed by Englsberger et al. [16] as the biologically inspired deadbeat (BID) controller, with a recent work by Egle et al. [17] implementing the transitions between BID-based running and DCM-based walking.

An alternative approach is to build upon the insights and experience gained over the last decades in bipedal walking research. One immediate challenge is that some of the methods developed for bipedal walking, such as the widely used zero moment point (ZMP) [18] and the instantaneous capture point (iCP) [19], [20], are not defined or are difficult to use during the flight phases. The proposed solutions for running [1], [21], [22] and hopping [23] combine ZMP-based trajectories during stance phases with ballistic trajectories during flight phases. The horizontal and vertical motions are generated independently, and the CoM trajectory is obtained by solving the dynamics numerically. Further development driven by the requirement to generate a vertical CoM motion has led to the variable-height inverted pendulum (VHIP) model [24], an extension of LIP that removes the assumption of constant CoM height above ground. The VHIP model was used in a model predictive controller (MPC) formulation by Ahn and Cho [25] to generate jumping motions with the vertical trajectory computed offline, and by Smaldone et al. [26] in walking and running motions. An alternative approach to extend the ZMP to 3-D space and generate seamless transitions between walking and running was proposed by Sugihara et al. [27], and was used for running [28] and hopping as a push recovery strategy [29].

One drawback of the ZMP-based methods is that the CoM equations of motion are coupled, with the vertical CoM acceleration appearing in the equations of the horizontal motion, which is caused by utilizing the vertical offset between the CoM and the ZMP. The 3D-DCM framework [10] avoids this problem by introducing the virtual repellent point (VRP); the VRP offset above ground can be used as a free motion parameter instead of the CoM-ZMP offset, leading to a decoupled and identical dynamics on all three axes. The equations of motion can be solved in closed-form [30], and the 3D-DCM framework has been successfully used for dynamic walking on various terrains [31], walking on moving support surfaces [32], and multicontact locomotion [33]. A time-varying 3D-DCM method was introduced by Hopkins et al. [34], but the resulting equations of motion can no longer be solved in closed-form; a piecewise constant approach for the time-varying DCM with offline computation of the constant values was recently proposed by Hanasaki et al. [35].

B. Contributions

The main contributions of this work are: 1) we extend the 3D-DCM framework with flight phase equations of motion and introduce a unified formulation that can be used for walking, running, and jumping motions; 2) we propose a highly efficient waypoint computation algorithm for a general sequence of motion phases including stance, flight, and transition phases; 3) within the 3D-DCM framework, we implement the transitions between the four-mentioned modes of motion (standing, walking, running, and jumping). In this context, we introduce a time-varying DCM motion phase with online computation of the equations of motion, using a piecewise constant approach.

The rest of the article is organized as follows. Section II presents the main concepts regarding the 3D-DCM framework. Section III gives a short overview of the proposed motion planner, followed by a detailed presentation in Section IV. The implementation of gait transitions is presented in Section V. Section VI provides a brief overview of the whole-body motion generation and control methods used in conjunction with the CoM motion planner. Extensive whole-body simulations with the humanoid robot TORO [36], demonstrating the capabilities of the proposed motion planner, are presented in Section VII, followed in Section VIII by a discussion of the planner's advantages and limitations. Finally, Section IX concludes this article.

II. Fundamentals

In this section, we review the main results from our previous works [10], [30] regarding the 3D-DCM framework, setting the main focus on its usage for reference trajectory generation.

Formally, the motion planning problem for the CoM $\boldsymbol{x} \in \mathbb{R}^3$ of a legged robot consists of finding a trajectory that connects the initial state $(\boldsymbol{x}_s, \dot{\boldsymbol{x}}_s)$ to the goal state $(\boldsymbol{x}_f, \dot{\boldsymbol{x}}_f)$, subject to the second-order CoM dynamics

$$\ddot{\boldsymbol{x}} = \frac{1}{m}\boldsymbol{f}_{\text{ext}} + \boldsymbol{g} \tag{1}$$

where m denotes the total mass of the robot, $\boldsymbol{f}_{\text{ext}} \in \mathbb{R}^3$ is the sum of all external forces acting on the robot, and $\boldsymbol{g} = (0\ 0\ -g)^T$ is the gravitational acceleration vector.

A. 3D-DCM Framework

The 3D-DCM framework introduced in [10] is a reformulation of the CoM dynamics (1) without loss of generality. First, the 3D-DCM framework introduces two points separated by a constant vertical offset Δz: the *enhanced centroidal moment pivot* (eCMP) and the VRP. The eCMP $\boldsymbol{e} \in \mathbb{R}^3$ encodes the direction and magnitude of the external force $\boldsymbol{f}_{\text{ext}}$ via

$$\boldsymbol{f}_{\text{ext}} = \frac{m}{b^2}(\boldsymbol{x} - \boldsymbol{e}) \tag{2}$$

where b is a time constant defined as

$$b := \sqrt{\frac{\Delta z}{g}}. \tag{3}$$

Similarly, the VRP $\boldsymbol{v} \in \mathbb{R}^3$ encodes the direction and magnitude of the CoM acceleration $\ddot{\boldsymbol{x}}$

$$\ddot{\boldsymbol{x}} = \frac{1}{b^2}(\boldsymbol{x} - \boldsymbol{v}). \tag{4}$$

Combining (1), (2), and (4) into one equation confirms the stated relation between the eCMP and VRP to be

$$\boldsymbol{v} = \boldsymbol{e} + (0\ 0\ \Delta z)^T. \tag{5}$$

Remark 1: The eCMP point is closely related to the CMP [37]. While the eCMP is a three-dimensional point that is not necessarily bound to the ground surface, the CMP is located

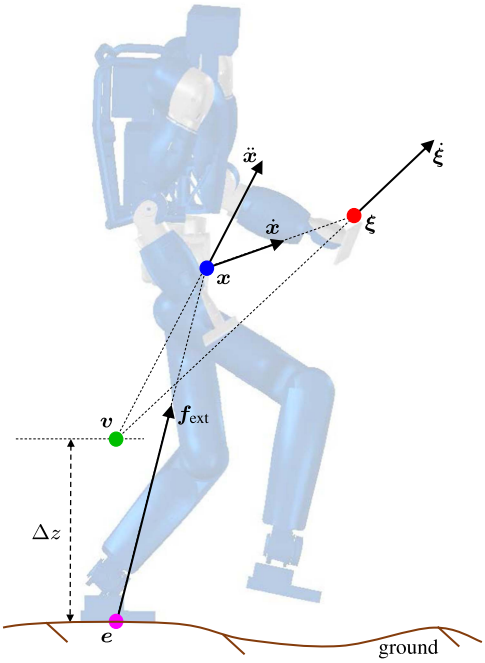


Fig. 1. Relations between the CoM ($\boldsymbol{x}$) and the points defined by the 3D-DCM framework: DCM ($\boldsymbol{\xi}$), VRP ($\boldsymbol{v}$), and eCMP ($\boldsymbol{e}$). The image depicts the robot pose and the instantaneous quantities during the stance phase of a running gait.

at the intersection of the ground surface with the line connecting the eCMP and the CoM.

Remark 2: The introduction of the VRP and the VRP offset Δz leads to a decoupled and identical dynamics on all three Cartesian axes. The VRP offset Δz is a motion parameter that can be chosen arbitrarily, as long as it remains strictly positive. For example, if Δz is chosen to be equal to the vertical distance between the CoM and eCMP, $\Delta z = x_z - e_z$, the vertical CoM acceleration $\ddot{x}_z$ becomes 0, which is typically used when generating walking trajectories on flat terrain. When walking on uneven terrain, Δz can be interpreted as the average CoM height above ground, assuming that the eCMP is placed on the ground. In this work, we propose an algorithm for computing Δz for running and jumping such that the generated CoM trajectory fulfills kinematic reachability constraints (see section IV-G).

A further point, the 3D-DCM $\boldsymbol{\xi}$ is defined in [10] as a linear combination of the CoM position and velocity using the time constant b

$$\boldsymbol{\xi} := \boldsymbol{x} + b\,\dot{\boldsymbol{x}}. \tag{6}$$

From (6) and (4), the relation between DCM and VRP is found to be

$$\dot{\boldsymbol{\xi}} = \frac{1}{b}(\boldsymbol{\xi} - \boldsymbol{v}). \tag{7}$$

The relations between the CoM and the points defined by the 3D-DCM framework are shown in Fig. 1.

Through the introduction of the DCM, the second-order CoM dynamics (4) is split into two first-order dynamics: the stable CoM dynamics relative to the DCM obtained from (6)

$$\dot{\boldsymbol{x}} = -\frac{1}{b}(\boldsymbol{x} - \boldsymbol{\xi}) \tag{8}$$

and the unstable DCM dynamics (7) with respect to the VRP.

B. Stable DCM Dynamics in Reverse Time

One of the main ideas presented in [10] is the generation of the DCM reference trajectory by starting at the end of the motion ($t = t_f$), choosing the goal DCM position[1] $\boldsymbol{\xi}(t_f) = \boldsymbol{\xi}_f$, and computing the DCM trajectory in reverse time. The key insight is that the unstable DCM dynamics (7) becomes stable when the time flow is reversed

$$\dot{\boldsymbol{\xi}}(-t) = -\frac{1}{b}\left(\boldsymbol{\xi}(-t) - \boldsymbol{v}(-t)\right). \tag{9}$$

Remark 3: This property of the DCM dynamics allows the motion planner to generate the CoM trajectory by combining two stable dynamics, the DCM dynamics in reverse time and the CoM dynamics in forward time. In this work, we refer to the generated reference trajectories obtained by integrating the two stable dynamics as *stable trajectories*. In the absence of perturbations, the reference trajectories can be followed by a tracking controller using a purely feedforward approach. However, although the reference DCM trajectory is stable when using the reverse time method, the actual DCM dynamics remains unstable; any deviation from the reference trajectory requires stabilization such as the one provided by the DCM Controller [10].

The reverse time approach brings two major benefits. First, for any reference VRP trajectory given as a spatial piecewise-linear interpolation over an arbitrary sequence of waypoints $(\boldsymbol{v}_i)_{i=1}^{n}$, the generated DCM trajectory is stable by construction. In effect, the set of all possible DCM trajectories is bounded by the convex hull constructed with the waypoints $\boldsymbol{v}_i$ and $\boldsymbol{\xi}_f$ as vertices, as shown in [30]. Second, the reference DCM trajectory is stable for any duration T of the generated motion, making the duration a parameter of the motion that can be chosen freely. Lower and upper bounds on the duration are imposed only by the limitations of the physical robotic system (kinematic reachability limits, friction cone constraints, etc.), and not by capturability concerns [38].

C. DCM and CoM Reference Trajectory Generation

As shown in our previous work [30], a complete, C^2 continuous CoM reference trajectory can be generated for walking or general multicontact locomotion by splitting the motion into a sequence of n_φ phases. The VRP offset Δz, the VRP trajectory, and the phase durations are the free parameters of the complete motion. For each phase φ with duration T_φ, the reference VRP trajectory $\boldsymbol{v}_\varphi(t)$ is designed as a spatial and temporal linear

[1] Typically, the goal DCM is chosen to be stationary ($\dot{\boldsymbol{\xi}}_f = \boldsymbol{0}$), i.e., the final DCM and VRP coincide, $\boldsymbol{\xi}_f = \boldsymbol{v}_f$, according to (7).

interpolation[2] between a start point $\boldsymbol{v}_{\varphi,0}$ and an end point $\boldsymbol{v}_{\varphi,T}$

$$\boldsymbol{v}_\varphi(t) = \left(1 - \frac{t}{T_\varphi}\right)\boldsymbol{v}_{\varphi,0} + \frac{t}{T_\varphi}\boldsymbol{v}_{\varphi,T} \tag{10}$$

where $t \in [0, T_\varphi]$ is the local time of the motion phase.

The DCM trajectory $\boldsymbol{\xi}_\varphi(t)$ is obtained by inserting (10) into (7), and solving the resulting differential equation using the DCM end point $\boldsymbol{\xi}_\varphi(T_\varphi) = \boldsymbol{\xi}_{\varphi,T}$ as a boundary condition, exploiting the reverse time stability of the DCM dynamics (9)

$$\begin{aligned}\boldsymbol{\xi}_\varphi(t) = &\left(1 - \frac{t+b}{T_\varphi} + \frac{b}{T_\varphi}e^{\frac{t-T_\varphi}{b}}\right)\boldsymbol{v}_{\varphi,0} \\ &+ \left(\frac{t+b}{T_\varphi} - \frac{b+T_\varphi}{T_\varphi}e^{\frac{t-T_\varphi}{b}}\right)\boldsymbol{v}_{\varphi,T} + e^{\frac{t-T_\varphi}{b}}\boldsymbol{\xi}_{\varphi,T}.\end{aligned} \tag{11}$$

The DCM start point $\boldsymbol{\xi}_{\varphi,0}$ of the current motion phase φ is obtained by evaluating (11) for $t = 0$

$$\begin{aligned}\boldsymbol{\xi}_{\varphi,0} = &\left(1 - \frac{b}{T_\varphi} + \frac{b}{T_\varphi}e^{-\frac{T_\varphi}{b}}\right)\boldsymbol{v}_{\varphi,0} \\ &+ \left(\frac{b}{T_\varphi} - \frac{b+T_\varphi}{T_\varphi}e^{-\frac{T_\varphi}{b}}\right)\boldsymbol{v}_{\varphi,T} + e^{-\frac{T_\varphi}{b}}\boldsymbol{\xi}_{\varphi,T}.\end{aligned} \tag{12}$$

Similarly, the CoM trajectory is derived by inserting (11) into (8) and solving the resulting linear differential equation using the CoM start point $\boldsymbol{x}_\varphi(0) = \boldsymbol{x}_{\varphi,0}$ as a boundary condition

$$\begin{aligned}\boldsymbol{x}_\varphi(t) = &\left(1 - \frac{t}{T_\varphi} + \frac{b}{T_\varphi}e^{-\frac{T_\varphi}{b}}\sinh\left(\frac{t}{b}\right) - e^{-\frac{t}{b}}\right)\boldsymbol{v}_{\varphi,0} \\ &+ \left(\frac{t}{T_\varphi} - \frac{b+T_\varphi}{T_\varphi}e^{-\frac{T_\varphi}{b}}\sinh\left(\frac{t}{b}\right)\right)\boldsymbol{v}_{\varphi,T} \\ &+ e^{-\frac{T_\varphi}{b}}\sinh\left(\frac{t}{b}\right)\boldsymbol{\xi}_{\varphi,T} + e^{-\frac{t}{b}}\boldsymbol{x}_{\varphi,0}.\end{aligned} \tag{13}$$

The CoM end point $\boldsymbol{x}_{\varphi,T}$ of the current motion phase is obtained by evaluating (13) for $t = T_\varphi$.

$$\begin{aligned}\boldsymbol{x}_{\varphi,T} = &\left(\frac{b}{T_\varphi}e^{-\frac{T_\varphi}{b}}\sinh\left(\frac{T_\varphi}{b}\right) - e^{-\frac{T_\varphi}{b}}\right)\boldsymbol{v}_{\varphi,0} \\ &+ \left(1 - \frac{b+T_\varphi}{T_\varphi}e^{-\frac{T_\varphi}{b}}\sinh\left(\frac{T_\varphi}{b}\right)\right)\boldsymbol{v}_{\varphi,T} \\ &+ e^{-\frac{T_\varphi}{b}}\sinh\left(\frac{T_\varphi}{b}\right)\boldsymbol{\xi}_{\varphi,T} + e^{-\frac{T_\varphi}{b}}\boldsymbol{x}_{\varphi,0}.\end{aligned} \tag{14}$$

The continuity of the complete trajectories is ensured by linking the end points of a motion phase with the start points of the subsequent phase, i.e., the following equalities hold for all phases: $\boldsymbol{v}_{\varphi,T} = \boldsymbol{v}_{\varphi+1,0}$, $\boldsymbol{\xi}_{\varphi,T} = \boldsymbol{\xi}_{\varphi+1,0}$, and $\boldsymbol{x}_{\varphi,T} = \boldsymbol{x}_{\varphi+1,0}$.

The complete reference trajectories for the VRP, DCM, and CoM can be described as piecewise interpolations over sequences of $n_w = n_\varphi + 1$ waypoints using (10), (11), and (13) as interpolation functions, respectively. Given the VRP waypoints, the DCM and CoM waypoints can be computed either sequentially or in matrix form using (12) and (14), as shown in [30]. Once the waypoints are known, the instantaneous VRP, DCM, and CoM positions are computed using the respective interpolation functions for the current phase φ and current time t. Finally, the CoM velocity and acceleration are obtained by evaluating (8) and (4), respectively.

[2]Higher order polynomials for temporal interpolation can be used to generate smoother CoM trajectories, as detailed in [30].

III. Overview

An overview of the proposed motion planner is shown in Fig. 2. The input consists of a sequence of footsteps given by a footstep planner, the motion phase durations, and the maximum distance between CoM and contact point at touchdown and takeoff, which is used to compute the VRP offset Δz. The motion planner constructs a sequence of motion phases with alternating stance and flight phases for running, single and double support phases for walking steps, and inserting gait transition phases into the phase sequence as described in Section V. The phase sequence construction and the VRP offset computation are performed asynchronously, i.e., only once for a given footstep sequence. Based on the motion phase sequence, the instantaneous CoM, the centroidal angular momentum (CAM), and the foot reference trajectories are computed synchronously at the execution rate of the whole-body controller. Finally, the waist orientation and the upper body reference trajectories are generated using a whole-body optimization approach [39] to produce the whole-body reference trajectory $\boldsymbol{x}_{\text{task}}^{\text{ref}}$. This is tracked by the passivity-based whole-body torque controller described in [31].

IV. Running and Jumping

Running and jumping are locomotion patterns characterized by alternating stance and flight phases; running switches the stance leg at each step, while jumping uses both legs simultaneously for takeoff and landing. In this section, we show how to describe the flight phase equations of motion such that they are compatible with the 3D-DCM framework. Additionally, we provide waypoint equations similar to (12) and (14) for the flight phase, and propose an algorithm for computing the DCM and CoM waypoints given an arbitrary sequence of motion phases, including flight phases.

A. Stance Phase

For bipedal running, the reference VRP trajectory during the stance phase can be designed either as a fixed point placed above the stance foot center $\boldsymbol{p}$ with the vertical offset Δz

$$\boldsymbol{v}_\varphi(t) = \boldsymbol{p} + (0\ 0\ \Delta z)^T \tag{15}$$

or as a heel-to-toe motion by positioning the VRP start and end points, $\boldsymbol{v}_{\varphi,0}$ and $\boldsymbol{v}_{\varphi,T}$, with appropriate offsets relative to the foot center. The DCM and CoM reference trajectories are generated using the method presented in Section II-C.

B. Flight Phase

During the flight phase, the CoM is affected only by gravity and necessarily follows a parabolic trajectory. The CoM equations of motion during a free-falling flight phase can be written

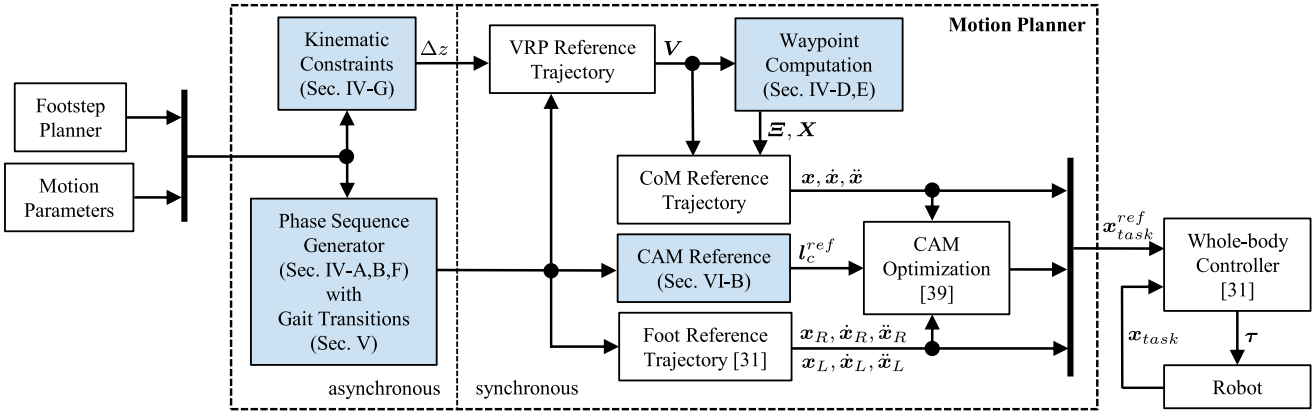


Fig. 2. Overview of the proposed motion planner. The main contributions of this work are highlighted in blue.

as follows:

$$\boldsymbol{x}_\varphi(t) = \boldsymbol{x}_{\varphi,0} + t\,\dot{\boldsymbol{x}}_{\varphi,0} + \frac{t^2}{2}\,\boldsymbol{g} \tag{16}$$

$$\dot{\boldsymbol{x}}_\varphi(t) = \dot{\boldsymbol{x}}_{\varphi,0} + t\,\boldsymbol{g}. \tag{17}$$

Note that during the flight phase, the VRP trajectory can no longer be chosen freely, as was the case for the stance phase. Nevertheless, the VRP trajectory is obtained from (4), using the free-falling condition $\ddot{\boldsymbol{x}} = \boldsymbol{g}$ and the definition of b as

$$\boldsymbol{v}_\varphi(t) = \boldsymbol{x}_\varphi(t) - b^2\boldsymbol{g} = \boldsymbol{x}_\varphi(t) + (0\ 0\ \Delta z)^T \tag{18}$$

which shows that the VRP trajectory follows the parabolic CoM trajectory with a constant vertical offset equal to Δz. This is equivalent to the observation that, during the flight phase, the eCMP trajectory $\boldsymbol{e}_\varphi(t)$ is identical to the CoM trajectory $\boldsymbol{x}_\varphi(t)$, which can also be verified by applying the free-falling condition $\boldsymbol{f}_{\text{ext}} = \boldsymbol{0}_{3\times1}$ to (2). This shows that the 3D-DCM framework is valid and coherent also during the flight phase (see Fig. 3).

The DCM trajectory is obtained by multiplying (17) by the time constant b, adding (16), and using the DCM definition (6) for $\boldsymbol{\xi}_\varphi(t)$ and $\boldsymbol{\xi}_{\varphi,0}$. The DCM trajectory with respect to the start CoM $\boldsymbol{x}_{\varphi,0}$ and DCM $\boldsymbol{\xi}_{\varphi,0}$ points is

$$\boldsymbol{\xi}_\varphi(t) = \boldsymbol{\xi}_{\varphi,0} + \frac{t}{b}\,(\boldsymbol{\xi}_{\varphi,0} - \boldsymbol{x}_{\varphi,0}) + \frac{t^2 + 2tb}{2}\,\boldsymbol{g}. \tag{19}$$

In order to use the reverse time approach described above, the DCM start point $\boldsymbol{\xi}_{\varphi,0}$ needs to be formulated with respect to the end point $\boldsymbol{\xi}_{\varphi,T}$. To this end, we evaluate (19) for $t = T_\varphi$ and rearrange the terms

$$\boldsymbol{\xi}_{\varphi,0} = \frac{b}{b+T_\varphi}\,\boldsymbol{\xi}_{\varphi,T} + \frac{T_\varphi}{b+T_\varphi}\,\boldsymbol{x}_{\varphi,0} - \frac{bT_\varphi^2 + 2b^2T_\varphi}{2(b+T_\varphi)}\,\boldsymbol{g}. \tag{20}$$

Remark 4: When the overall motion contains a flight phase, the DCM waypoints can no longer be computed independently of the CoM waypoints, as was the case in [30]. Here, in (20), the DCM start point $\boldsymbol{\xi}_{\varphi,0}$ of the flight phase depends on the CoM start point $\boldsymbol{x}_{\varphi,0}$. Nevertheless, the DCM and CoM waypoints can be computed together in matrix form or using an algorithmic approach, as shown in the subsequent section.

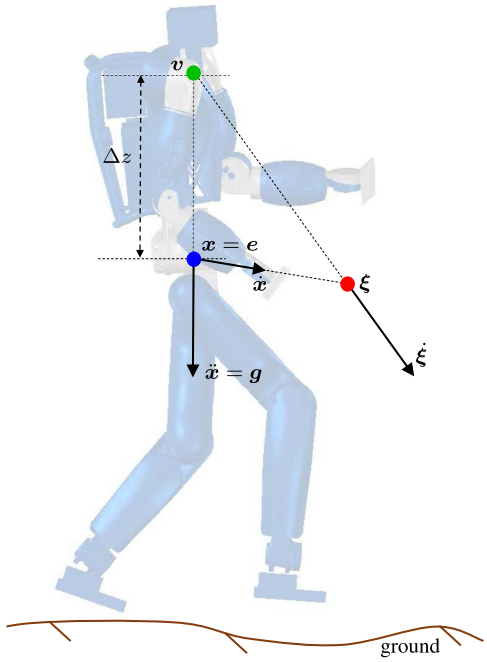


Fig. 3. Point relations of the 3D-DCM framework during a flight phase, showing that the framework maintains its coherence and validity under the free-falling condition $\boldsymbol{f}_{\text{ext}} = \boldsymbol{0}_{3\times1}$.

The CoM end point $\boldsymbol{x}_{\varphi,T}$ of the flight phase is obtained by evaluating (16) for $t = T_\varphi$, using the DCM definition (6) for $\boldsymbol{\xi}_{\varphi,0}$ to eliminate the velocity term $\dot{\boldsymbol{x}}_{\varphi,0}$, and replacing $\boldsymbol{\xi}_{\varphi,0}$ with the result from (20)

$$\boldsymbol{x}_{\varphi,T} = \frac{T_\varphi}{b+T_\varphi}\,\boldsymbol{\xi}_{\varphi,T} + \frac{b}{b+T_\varphi}\,\boldsymbol{x}_{\varphi,0} - \frac{bT_\varphi^2}{2(b+T_\varphi)}\,\boldsymbol{g}. \tag{21}$$

C. Unified Formulation for Waypoint Equations

Recall that the complete motion plan consists of a sequence of heterogeneous motion phases containing stance, flight, and other transition phases, which we introduce in subsequent sections of this article. In order to develop a general waypoint computation method, we propose a unified formulation of the DCM and

CoM waypoint equations. This approach makes the framework extensible with new motion phase types.

Combining (12) and (20) into one equation produces the general form for the DCM start point

$$\boldsymbol{\xi}_{\varphi,0} = \alpha_{\varphi,\xi}\,\boldsymbol{v}_{\varphi,0} + \beta_{\varphi,\xi}\,\boldsymbol{v}_{\varphi,T} + \gamma_{\varphi,\xi}\,\boldsymbol{\xi}_{\varphi,T} + \delta_{\varphi,\xi}\,\boldsymbol{x}_{\varphi,0} + \varepsilon_{\varphi,\xi}\,\boldsymbol{g}. \tag{22}$$

For the stance phase, the coefficients $\delta_{\varphi,\xi}$ and $\varepsilon_{\varphi,\xi}$ are 0, while $\alpha_{\varphi,\xi}$, $\beta_{\varphi,\xi}$ and $\gamma_{\varphi,\xi}$ correspond to the coefficients in (12). Accordingly, for the flight phase, the coefficients $\alpha_{\varphi,\xi}$ and $\beta_{\varphi,\xi}$ are 0, while $\gamma_{\varphi,\xi}$, $\delta_{\varphi,\xi}$, and $\varepsilon_{\varphi,\xi}$ correspond to the respective coefficients in (20).

Similarly, combining (14) and (21) into one equation produces the general form for computing the CoM end point:

$$\boldsymbol{x}_{\varphi,T} = \alpha_{\varphi,x}\,\boldsymbol{v}_{\varphi,0} + \beta_{\varphi,x}\,\boldsymbol{v}_{\varphi,T} + \gamma_{\varphi,x}\,\boldsymbol{\xi}_{\varphi,T} + \delta_{\varphi,x}\,\boldsymbol{x}_{\varphi,0} + \varepsilon_{\varphi,x}\,\boldsymbol{g} \tag{23}$$

with the same considerations regarding the coefficients as discussed above for (22).

D. Waypoint Computation in Matrix Form

The complete trajectories for VRP, DCM, and CoM can be described as n_φ piecewise interpolations over $n_w = n_\varphi + 1$ waypoints. We collect the VRP waypoints in a matrix $\boldsymbol{V} = \begin{bmatrix}\boldsymbol{v}_1 & \dots & \boldsymbol{v}_{n_w}\end{bmatrix}^T \in \mathbb{R}^{n_w\times 3}$, the DCM waypoints in $\boldsymbol{\Xi} = \begin{bmatrix}\boldsymbol{\xi}_1 & \dots & \boldsymbol{\xi}_{n_w}\end{bmatrix}^T \in \mathbb{R}^{n_w\times 3}$, and the CoM waypoints in $\boldsymbol{X} = \begin{bmatrix}\boldsymbol{x}_1 & \dots & \boldsymbol{x}_{n_w}\end{bmatrix}^T \in \mathbb{R}^{n_w\times 3}$. Our goal is to compute the unknown DCM and CoM waypoints, $\boldsymbol{\Xi}$ and $\boldsymbol{X}$, in terms of the VRP waypoints $\boldsymbol{V}$, the DCM final point $\boldsymbol{\xi}_f$, the CoM start point $\boldsymbol{x}_s$, and the gravity vector $\boldsymbol{g}$. To this end, we use the derived equations (22) and (23). As these equations are expressed in terms of start and end points ($\boldsymbol{v}_{\varphi,0}$, $\boldsymbol{v}_{\varphi,T}$, etc.), we introduce two selection matrices: $\boldsymbol{S}_0 = \begin{bmatrix}\boldsymbol{I}_{n_\varphi\times n_\varphi} & \boldsymbol{0}_{n_\varphi\times 1}\end{bmatrix} \in \mathbb{R}^{n_\varphi\times n_w}$ selects the motion phase start points, while $\boldsymbol{S}_T = \begin{bmatrix}\boldsymbol{0}_{n_\varphi\times 1} & \boldsymbol{I}_{n_\varphi\times n_\varphi}\end{bmatrix} \in \mathbb{R}^{n_\varphi\times n_w}$ selects the end points from any waypoint matrix $\boldsymbol{V}$, $\boldsymbol{\Xi}$, or $\boldsymbol{X}$.

We write (22) in matrix form for all n_φ phases as

$$\boldsymbol{S}_0\boldsymbol{\Xi} = \boldsymbol{A}_\xi\boldsymbol{S}_0\boldsymbol{V} + \boldsymbol{B}_\xi\boldsymbol{S}_T\boldsymbol{V} + \boldsymbol{\Gamma}_\xi\boldsymbol{S}_T\boldsymbol{\Xi} + \boldsymbol{\Delta}_\xi\boldsymbol{S}_0\boldsymbol{X} + \boldsymbol{\varepsilon}_\xi\,\boldsymbol{g}^T \tag{24}$$

where $\boldsymbol{A}_\xi$, $\boldsymbol{B}_\xi$, $\boldsymbol{\Gamma}_\xi$, and $\boldsymbol{\Delta}_\xi$ are square $(n_\varphi \times n_\varphi)$, diagonal matrices containing the coefficients $\alpha_{\varphi,\xi}$, $\beta_{\varphi,\xi}$, $\gamma_{\varphi,\xi}$, and $\delta_{\varphi,\xi}$, respectively, while $\boldsymbol{\varepsilon}_\xi$ is a vector containing the coefficients $\varepsilon_{\varphi,\xi}$. Note that (24) consists of n_φ equations, whereas $\boldsymbol{\Xi}$ contains n_w waypoints. In order to bring (24) into square form, we add the *DCM terminal constraint*, $\boldsymbol{\xi}_{n_w} = \boldsymbol{\xi}_f$, which can be expressed in terms of $\boldsymbol{\Xi}$ as

$$\underbrace{\begin{bmatrix}\boldsymbol{0}_{n_\varphi\times n_\varphi} & \boldsymbol{0}_{n_\varphi\times 1}\\ \boldsymbol{0}_{1\times n_\varphi} & 1\end{bmatrix}}_{\boldsymbol{S}_n\,\in\,\mathbb{R}^{n_w\times n_w}}\boldsymbol{\Xi} = \underbrace{\begin{bmatrix}\boldsymbol{0}_{n_\varphi\times 1}\\ 1\end{bmatrix}}_{\boldsymbol{s}_f\,\in\,\mathbb{R}^{n_w}}\boldsymbol{\xi}_f^T. \tag{25}$$

Multiplying (24) on the left with $\boldsymbol{S}_0^T$, adding (25), and grouping the $\boldsymbol{\Xi}$ and $\boldsymbol{X}$ terms on the left side yields

$$(\boldsymbol{I} - \boldsymbol{S}_0^T\boldsymbol{\Gamma}_\xi\boldsymbol{S}_T)\boldsymbol{\Xi} - \boldsymbol{S}_0^T\boldsymbol{\Delta}_\xi\boldsymbol{S}_0\boldsymbol{X} = \boldsymbol{S}_0^T(\boldsymbol{A}_\xi\boldsymbol{S}_0 + \boldsymbol{B}_\xi\boldsymbol{S}_T)\boldsymbol{V} + \boldsymbol{S}_0^T\boldsymbol{\varepsilon}_\xi\,\boldsymbol{g}^T + \boldsymbol{s}_f\,\boldsymbol{\xi}_f^T \tag{26}$$

where we used the equality $\boldsymbol{S}_0^T\boldsymbol{S}_0 + \boldsymbol{S}_n = \boldsymbol{I}_{n_w\times n_w} = \boldsymbol{I}$.

For the CoM waypoints, we write (23) in matrix form for all n_φ phases as

$$\boldsymbol{S}_T\boldsymbol{X} = \boldsymbol{A}_x\boldsymbol{S}_0\boldsymbol{V} + \boldsymbol{B}_x\boldsymbol{S}_T\boldsymbol{V} + \boldsymbol{\Gamma}_x\boldsymbol{S}_T\boldsymbol{\Xi} + \boldsymbol{\Delta}_x\boldsymbol{S}_0\boldsymbol{X} + \boldsymbol{\varepsilon}_x\,\boldsymbol{g}^T \tag{27}$$

gathering the coefficients $\alpha_{\varphi,x}$, $\beta_{\varphi,x}$, $\gamma_{\varphi,x}$, $\delta_{\varphi,x}$, $\varepsilon_{\varphi,x}$ into the square, diagonal matrices $\boldsymbol{A}_x$, $\boldsymbol{B}_x$, $\boldsymbol{\Gamma}_x$, $\boldsymbol{\Delta}_x$, and the vector $\boldsymbol{\varepsilon}_x$, respectively. Multiplying (27) on the left with $\boldsymbol{S}_T^T$, and adding the *CoM initial constraint* at the start of the motion, $\boldsymbol{x}_1 = \boldsymbol{x}_s$, expressed in terms of $\boldsymbol{X}$ as

$$\underbrace{\begin{bmatrix}1 & \boldsymbol{0}_{1\times n_\varphi}\\ \boldsymbol{0}_{n_\varphi\times 1} & \boldsymbol{0}_{n_\varphi\times n_\varphi}\end{bmatrix}}_{\boldsymbol{S}_1\,\in\,\mathbb{R}^{n_w\times n_w}}\boldsymbol{X} = \underbrace{\begin{bmatrix}1\\ \boldsymbol{0}_{n_\varphi\times 1}\end{bmatrix}}_{\boldsymbol{s}_s\,\in\,\mathbb{R}^{n_w}}\boldsymbol{x}_s^T \tag{28}$$

yields

$$(\boldsymbol{I} - \boldsymbol{S}_T^T\boldsymbol{\Delta}_x\boldsymbol{S}_0)\boldsymbol{X} - \boldsymbol{S}_T^T\boldsymbol{\Gamma}_x\boldsymbol{S}_T\boldsymbol{\Xi} = \boldsymbol{S}_T^T(\boldsymbol{A}_x\boldsymbol{S}_0 + \boldsymbol{B}_x\boldsymbol{S}_T)\boldsymbol{V}\,\boldsymbol{S}_T^T\boldsymbol{\varepsilon}_x\,\boldsymbol{g}^T + \boldsymbol{s}_s\,\boldsymbol{x}_s^T \tag{29}$$

where we used the equality $\boldsymbol{S}_T^T\boldsymbol{S}_T + \boldsymbol{S}_1 = \boldsymbol{I}$.

Finally, stacking (26) and (29) into one equation leads to the linear system

$$\begin{bmatrix}\boldsymbol{I} - \boldsymbol{S}_0^T\boldsymbol{\Gamma}_\xi\boldsymbol{S}_T & -\boldsymbol{S}_0^T\boldsymbol{\Delta}_\xi\boldsymbol{S}_0\\ -\boldsymbol{S}_T^T\boldsymbol{\Gamma}_x\boldsymbol{S}_T & \boldsymbol{I} - \boldsymbol{S}_T^T\boldsymbol{\Delta}_x\boldsymbol{S}_0\end{bmatrix}\begin{bmatrix}\boldsymbol{\Xi}\\ \boldsymbol{X}\end{bmatrix} = \begin{bmatrix}\boldsymbol{S}_0^T\boldsymbol{A}_\xi\boldsymbol{S}_0 + \boldsymbol{S}_0^T\boldsymbol{B}_\xi\boldsymbol{S}_T\\ \boldsymbol{S}_T^T\boldsymbol{A}_x\boldsymbol{S}_0 + \boldsymbol{S}_T^T\boldsymbol{B}_x\boldsymbol{S}_T\end{bmatrix}\boldsymbol{V} + \begin{bmatrix}\boldsymbol{S}_0^T\boldsymbol{\varepsilon}_\xi\\ \boldsymbol{S}_T^T\boldsymbol{\varepsilon}_x\end{bmatrix}\boldsymbol{g}^T + \begin{bmatrix}\boldsymbol{s}_f\\ \boldsymbol{0}_{n_w\times 1}\end{bmatrix}\boldsymbol{\xi}_f^T + \begin{bmatrix}\boldsymbol{0}_{n_w\times 1}\\ \boldsymbol{s}_s\end{bmatrix}\boldsymbol{x}_s^T \tag{30}$$

which can be solved to find the unknown $\boldsymbol{\Xi}$ and $\boldsymbol{X}$.

E. Highly Efficient Waypoint Computation Using an Algorithmic Approach

The waypoint computation in matrix form, presented in (30), is a compact and mathematically appealing way of computing the DCM and CoM waypoints. However, the computational complexity of the algorithm is $\mathcal{O}(n_\varphi^3)$ due to the matrix multiplications involved and the requirement to solve a linear system with $2n_\varphi$ variables. From a practical point of view, when implementing the algorithm on a realtime robotic hardware, its high computational complexity is detrimental, potentially imposing an upper limit on the number of phases n_φ. Motivated by this shortcoming of the matrix form algorithm, we propose an algorithm for waypoint computation with a computational complexity of $\mathcal{O}(n_\varphi^2)$ as Algorithm 1.

The algorithm requires three iterations over all motion phases. The main idea is to compute partial solutions for the DCM and CoM waypoints from the known quantities during the first two iterations and to keep track of the coefficients corresponding to the unknown CoM waypoints. During the third iteration, the gathered coefficients are used to compute the complete solution. The input to the algorithm consists of the VRP waypoint matrix $\boldsymbol{V}$, the DCM final point $\boldsymbol{\xi}_f$, the CoM start point $\boldsymbol{x}_s$, the phase durations given as a list $\boldsymbol{T} = (T_i)_{i=1}^{n_\varphi}$, and the DCM time constant b. The term $\boldsymbol{v}_i$ denotes the ith waypoint in the matrix $\boldsymbol{V}$, with analogous notation for the DCM and CoM waypoint matrices, $\boldsymbol{\Xi}$ and $\boldsymbol{X}$. The algorithm employs two square $(n_w \times n_w)$ matrices, $\boldsymbol{C}_\xi$ and $\boldsymbol{C}_x$, to store the coefficients corresponding to the unknown CoM waypoints. The value at the ith row and jth column in the matrix $\boldsymbol{C}_\xi$, written as $\boldsymbol{C}_\xi[i,j]$, denotes the coefficient with which the CoM waypoint $\boldsymbol{x}_j$ needs to be multiplied when computing the DCM waypoint $\boldsymbol{\xi}_i$ (see Algorithm 1, line 31). The matrix $\boldsymbol{C}_x$ is defined in a similar manner, with $\boldsymbol{C}_x[i,j]$ relating the CoM waypoint $\boldsymbol{x}_j$ with $\boldsymbol{x}_i$ (see line 27). The partial DCM and CoM waypoints are stored in the matrices $\boldsymbol{\Xi}^p = \begin{bmatrix}\boldsymbol{\xi}_1^p \ \dots \ \boldsymbol{\xi}_{n_w}^p\end{bmatrix}^T \in \mathbb{R}^{n_w \times 3}$ and $\boldsymbol{X}^p = \begin{bmatrix}\boldsymbol{x}_1^p \ \dots \ \boldsymbol{x}_{n_w}^p\end{bmatrix}^T \in \mathbb{R}^{n_w \times 3}$, respectively.

In the following, we describe in detail how the formulas for computing the partial DCM and CoM waypoints and the coefficients $\boldsymbol{C}_\xi$ and $\boldsymbol{C}_x$ are obtained from the previously derived equations. We start by rewriting (22) using the waypoint notation as

$$\boldsymbol{\xi}_i = \alpha_\xi \, \boldsymbol{v}_i + \beta_\xi \, \boldsymbol{v}_{i+1} + \gamma_\xi \, \boldsymbol{\xi}_{i+1} + \delta_\xi \, \boldsymbol{x}_i + \varepsilon_\xi \, \boldsymbol{g}. \tag{31}$$

It can be proven by induction that the DCM waypoint $\boldsymbol{\xi}_i$ can be written as

$$\boldsymbol{\xi}_i = \boldsymbol{\xi}_i^p + \sum_{j=i}^{n_w} C_\xi[i,j] \, \boldsymbol{x}_j. \tag{32}$$

For $i = n_w$, (32) holds due to the initialization of $\boldsymbol{C}_\xi$ and $\boldsymbol{\xi}_{n_w}^p$ in lines 2 and 3. Assuming that (32) holds for $i+1$, we replace $\boldsymbol{\xi}_{i+1}$ in (31) with the corresponding form from (32) and rearrange the terms to obtain

$$\boldsymbol{\xi}_i = \overbrace{\alpha_\xi \, \boldsymbol{v}_i + \beta_\xi \, \boldsymbol{v}_{i+1} + \gamma_\xi \, \boldsymbol{\xi}_{i+1}^p + \varepsilon_\xi \, \boldsymbol{g}}^{\boldsymbol{\xi}_i^p} + \delta_\xi \, \boldsymbol{x}_i + \sum_{j=i+1}^{n_w} \gamma_\xi \, C_\xi[i+1,j] \, \boldsymbol{x}_j. \tag{33}$$

We take advantage of the fact that the first four terms are independent of the unknown CoM waypoints $\boldsymbol{X}$ and can be summed to produce the partial DCM waypoint $\boldsymbol{\xi}_i^p$ (see line 6). The last two terms in (33) give the corresponding equations for computing the DCM coefficients $\boldsymbol{C}_\xi$ (see lines 7 to 10).

We proceed analogously for the CoM waypoints, by rewriting (23) using the waypoint notation as

$$\boldsymbol{x}_i = \alpha_x \, \boldsymbol{v}_{i-1} + \beta_x \, \boldsymbol{v}_i + \gamma_x \, \boldsymbol{\xi}_i + \delta_x \, \boldsymbol{x}_{i-1} + \varepsilon_x \, \boldsymbol{g}. \tag{34}$$

Algorithm 1: Efficient Algorithm for Waypoint Computation.

Input: $\boldsymbol{V}, \boldsymbol{\xi}_f, \boldsymbol{x}_s, \boldsymbol{T}, b$
Output: $\boldsymbol{\Xi}, \boldsymbol{X}$

1: // backward iteration: partial solution for DCM waypoints
2: $\boldsymbol{C}_\xi \leftarrow \boldsymbol{0}_{n_w \times n_w}$
3: $\boldsymbol{\xi}_{n_w}^p \leftarrow \boldsymbol{\xi}_f$
4: **for** $i \leftarrow n_\varphi$ to 1 **do**
5: $\quad \alpha_\xi, \beta_\xi, \gamma_\xi, \delta_\xi, \varepsilon_\xi \leftarrow$ DCMCoefficients (T_i, b)
6: $\quad \boldsymbol{\xi}_i^p \leftarrow \alpha_\xi \, \boldsymbol{v}_i + \beta_\xi \, \boldsymbol{v}_{i+1} + \gamma_\xi \, \boldsymbol{\xi}_{i+1}^p + \varepsilon_\xi \, \boldsymbol{g}$
7: $\quad \boldsymbol{C}_\xi[i,i] \leftarrow \delta_\xi$
8: $\quad$ **for** $j \leftarrow i+1$ to n_w **do**
9: $\quad\quad \boldsymbol{C}_\xi[i,j] \leftarrow \gamma_\xi \, \boldsymbol{C}_\xi[i+1,j]$
10: $\quad$ **end for**
11: **end for**
12: // forward iteration: partial solution for CoM waypoints
13: $\boldsymbol{C}_x \leftarrow \boldsymbol{0}_{n_w \times n_w}$
14: $\boldsymbol{x}_1^p \leftarrow \boldsymbol{x}_s$
15: **for** $i \leftarrow 2$ to n_w **do**
16: $\quad \alpha_x, \beta_x, \gamma_x, \delta_x, \varepsilon_x \leftarrow$ CoMCoefficients (T_{i-1}, b)
17: $\quad \eta \leftarrow 1 - \gamma_x \, \boldsymbol{C}_\xi[i,i] - \delta_x \, \boldsymbol{C}_x[i-1,i]$
18: $\quad \boldsymbol{x}_i^p \leftarrow$
$\quad (\alpha_x \, \boldsymbol{v}_{i-1} + \beta_x \, \boldsymbol{v}_i + \gamma_x \, \boldsymbol{\xi}_i^p + \delta_x \, \boldsymbol{x}_{i-1}^p + \varepsilon_x \, \boldsymbol{g}) \, / \, \eta$
19: $\quad$ **for** $j \leftarrow i+1$ to n_w **do**
20: $\quad\quad \boldsymbol{C}_x[i,j] \leftarrow (\gamma_x \, \boldsymbol{C}_\xi[i,j] + \delta_x \, \boldsymbol{C}_x[i-1,j]) \, / \, \eta$
21: $\quad$ **end for**
22: **end for**
23: // second backward iteration: complete solution
24: **for** $i \leftarrow n_w$ to 1 **do**
25: $\quad \boldsymbol{x}_i \leftarrow \boldsymbol{x}_i^p$
26: $\quad$ **for** $j \leftarrow i+1$ to n_w **do**
27: $\quad\quad \boldsymbol{x}_i \leftarrow \boldsymbol{x}_i + \boldsymbol{C}_x[i,j] \, \boldsymbol{x}_j$
28: $\quad$ **end for**
29: $\quad \boldsymbol{\xi}_i \leftarrow \boldsymbol{\xi}_i^p$
30: $\quad$ **for** $j \leftarrow i$ to n_w **do**
31: $\quad\quad \boldsymbol{\xi}_i \leftarrow \boldsymbol{\xi}_i + \boldsymbol{C}_\xi[i,j] \, \boldsymbol{x}_j$
32: $\quad$ **end for**
33: **end for**

and showing that it can also be written in the following form:

$$\boldsymbol{x}_i = \boldsymbol{x}_i^p + \sum_{j=i+1}^{n_w} C_x[i,j] \, \boldsymbol{x}_j. \tag{35}$$

The proof is again by induction. For $i = 1$, (35) holds, i.e., $\boldsymbol{x}_1 = \boldsymbol{x}_s$, due to the initialization of $\boldsymbol{C}_x$ and $\boldsymbol{x}_1^p$ in lines 13 and 14. Assuming that (35) holds for $i-1$, we replace in (34) the DCM waypoint $\boldsymbol{\xi}_i$ with the result from (32) and $\boldsymbol{x}_{i-1}$ with the corresponding equation from (35)

$$\boldsymbol{x}_i = \alpha_x \, \boldsymbol{v}_{i-1} + \beta_x \, \boldsymbol{v}_i + \gamma_x \, \boldsymbol{\xi}_i^p + \sum_{j=i}^{n_w} \gamma_x \, C_\xi[i,j] \, \boldsymbol{x}_j + \delta_x \, \boldsymbol{x}_{i-1}^p + \sum_{j=i}^{n_w} \delta_x \, C_x[i-1,j] \, \boldsymbol{x}_j + \varepsilon_x \, \boldsymbol{g}. \tag{36}$$

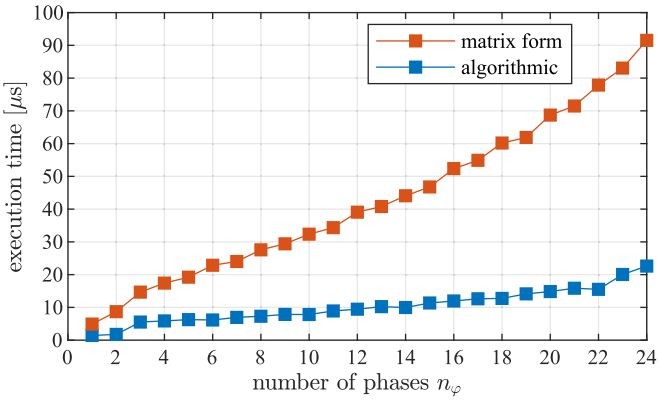


Fig. 4. Execution time comparison of the two waypoint computation methods. Algorithm 1 outperforms the matrix form on the realtime robotic hardware due to its lower computational complexity, $\mathcal{O}(n_\varphi^2)$ versus $\mathcal{O}(n_\varphi^3)$.

Note that $\boldsymbol{x}_i$ appears on both sides of the equation (on the right side, as part of the sum terms containing $\boldsymbol{x}_j$, for $j = i$). Grouping all terms containing $\boldsymbol{x}_i$ on the left side and rearranging the terms allows us to bring (36) in conformance to (35)

$$\begin{aligned}&\underbrace{\left(1 - \gamma_x\, C_\xi[i,i] - \delta_x\, C_x[i-1,i]\right)}_{\eta} \boldsymbol{x}_i \\ &= \alpha_x\, \boldsymbol{v}_{i-1} + \beta_x\, \boldsymbol{v}_i + \gamma_x\, \boldsymbol{\xi}_i^p + \delta_x\, \boldsymbol{x}_{i-1}^p + \varepsilon_x\, \boldsymbol{g} \\ &\quad + \sum_{j=i+1}^{n_w} \left(\gamma_x\, C_\xi[i,j] + \delta_x\, C_x[i-1,j]\right) \boldsymbol{x}_j. \end{aligned} \tag{37}$$

The sum of the first five terms on the right-hand side of the equation divided by η represents the partial CoM waypoint $\boldsymbol{x}_i^p$ (see line 18), while the last term provides the formula for computing $\boldsymbol{C}_x[i,j]$ (see lines 19–21).

To obtain the complete solution, the algorithm exploits the structure of (32) and (35) by iterating backwards over all motion phases. During each iteration cycle, the CoM waypoint $\boldsymbol{x}_i$ is computed from the known future waypoints $\boldsymbol{x}_{i+1}$ to $\boldsymbol{x}_{n_w}$ (see lines 25–28). Once $\boldsymbol{x}_i$ is known, the DCM waypoint $\boldsymbol{\xi}_i$ can be computed immediately as it depends only on the CoM waypoints $\boldsymbol{x}_i$ to $\boldsymbol{x}_{n_w}$ (see lines 29–32).

To verify the execution performance of the two waypoint computation methods, Algorithm 1 and the matrix form (30), we implemented both in C++ and tested them on the realtime computer of our humanoid robot TORO. The median execution time for various number of phases is shown in Fig. 4: the algorithmic approach clearly outperforms the matrix form in all cases due to its lower computational complexity.

F. Flight-to-Stance and Stance-to-Flight Motion Phases

In our framework, we aim to avoid discontinuities in the generated reference trajectories, which are undesirable for two reasons. First, a real robot has limited control bandwidth, which means that the controller is unable to follow the reference trajectory in case of a discontinuity. Second, discontinuities tend to excite unmodeled joint and link elasticities on the real robot, thereby further degrading the tracking performance of the controller.

The continuity of the DCM and CoM reference trajectories is ensured by linking the start and end points of adjacent phases. In contrast, the VRP and eCMP trajectories presented so far are discontinuous at the start and at the end of the flight phase, leading to discontinuities in the generated forces. For example, the eCMP trajectory during the flight phase is identical to the CoM trajectory, while during the stance phase it is designed to be a fixed point on the ground, coinciding with the foot center. In order to eliminate the discontinuities, we introduce two new motion phases: a flight-to-stance and stance-to-flight phase. For these phases, we aim to express the DCM start point $\boldsymbol{\xi}_{\varphi,0}$ and the CoM end point $\boldsymbol{x}_{\varphi,T}$ in the general form given by (22) and (23), respectively, which allows us to use the waypoint computation algorithm without any modifications. During both motion phases, the VRP is interpolated from a start point $\boldsymbol{v}_{\varphi,0}$ to an end point $\boldsymbol{v}_{\varphi,T}$ using the interpolation function (10). However, unlike the stance phase, either the start or the end point is constrained by the free-falling condition $\ddot{\boldsymbol{x}} = \boldsymbol{g}$, the effects of which are discussed in detail as follows.

Flight-to-Stance Phase: For the flight-to-stance phase, the VRP start point $\boldsymbol{v}_{\varphi,0}$ can be written as

$$\boldsymbol{v}_{\varphi,0} = \boldsymbol{x}_{\varphi,0} - b^2 \boldsymbol{g} \tag{38}$$

matching the preceding flight phase trajectory. This means that the VRP start point $\boldsymbol{v}_{\varphi,0}$ is an unknown quantity that depends on the CoM start point $\boldsymbol{x}_{\varphi,0}$, which is yet to be computed using the waypoint computation algorithm. Inserting (38) into (12) yields

$$\boldsymbol{\xi}_{\varphi,0} = \beta_{\varphi,\xi}\, \boldsymbol{v}_{\varphi,T} + \gamma_{\varphi,\xi}\, \boldsymbol{\xi}_{\varphi,T} + \alpha_{\varphi,\xi}\, \boldsymbol{x}_{\varphi,0} - \alpha_{\varphi,\xi} b^2 \boldsymbol{g} \tag{39}$$

which matches the general form (22) with $\alpha_{\varphi,\xi} = 0$ and $\varepsilon_{\varphi,\xi} = -\alpha_{\varphi,\xi} b^2$. Similarly, inserting (38) into (14) yields

$$\begin{aligned}\boldsymbol{x}_{\varphi,T} &= \beta_{\varphi,x}\, \boldsymbol{v}_{\varphi,T} + \gamma_{\varphi,x}\, \boldsymbol{\xi}_{\varphi,T} \\ &\quad + (\alpha_{\varphi,x} + \delta_{\varphi,x})\, \boldsymbol{x}_{\varphi,0} - \alpha_{\varphi,x} b^2 \boldsymbol{g} \end{aligned} \tag{40}$$

which matches the CoM end point general form (23).

Stance-to-Flight Phase: For the stance-to-flight phase, the VRP end point $\boldsymbol{v}_{\varphi,T}$ constraint can be written as

$$\boldsymbol{v}_{\varphi,T} = \boldsymbol{x}_{\varphi,T} - b^2 \boldsymbol{g}. \tag{41}$$

Replacing (41) in (14) and solving for $\boldsymbol{x}_{\varphi,T}$ leads to a CoM end point equation matching the general form:

$$\begin{aligned}\boldsymbol{x}_{\varphi,T} &= \frac{\alpha_{\varphi,x}}{1-\beta_{\varphi,x}}\, \boldsymbol{v}_{\varphi,0} + \frac{\gamma_{\varphi,x}}{1-\beta_{\varphi,x}}\, \boldsymbol{\xi}_{\varphi,T} \\ &\quad + \frac{\delta_{\varphi,x}}{1-\beta_{\varphi,x}}\, \boldsymbol{x}_{\varphi,0} - \frac{\beta_{\varphi,x} b^2}{1-\beta_{\varphi,x}}\, \boldsymbol{g}. \end{aligned} \tag{42}$$

Note: It can be easily verified that $\beta_{\varphi,x}$, the coefficient corresponding to $\boldsymbol{v}_{\varphi,T}$ in (14), satisfies the inequality $0 < \beta_{\varphi,x} < 1$ for all possible durations $T_\varphi > 0$ and time constants $b > 0$. This guarantees that the denominator $1 - \beta_{\varphi,x}$ can never be equal to 0. Replacing (41) in (12) and substituting $\boldsymbol{x}_{\varphi,T}$ with the result obtained in (42) leads to a DCM start point equation matching

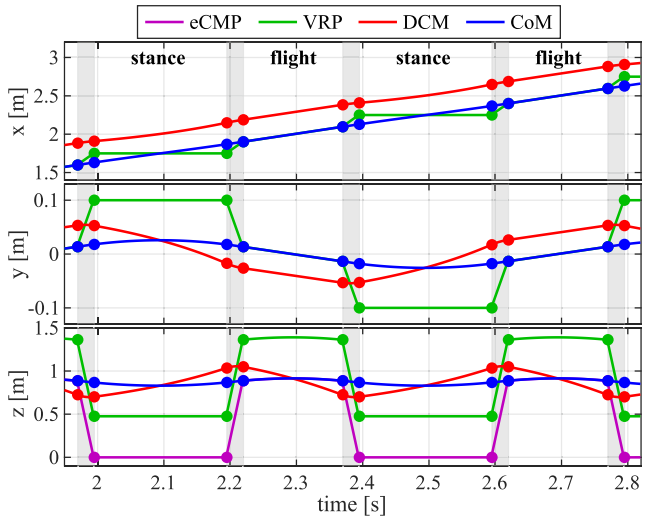


Fig. 5. Reference trajectory generation example for two running steps. The gray areas denote the stance-to-flight and flight-to-stance phases. The dots indicate the start and end waypoints of each phase.

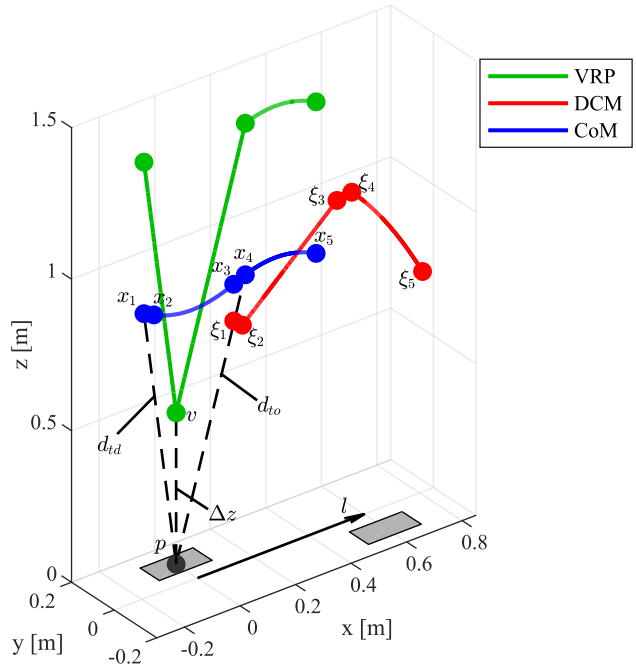


Fig. 6. Modeling kinematic reachability constraints during running. The continuous lines illustrate the trajectories during one running step from the footstep $\boldsymbol{p}$ with step vector $\boldsymbol{l}$, using a constant VRP $\boldsymbol{v}$ during the stance phase. The distances between CoM and contact point at touchdown and takeoff are denoted by d_{td} and d_{to}, respectively, and can be computed as closed-form functions of the VRP offset above ground Δz.

the general form

$$\boldsymbol{\xi}_{\varphi,0} = \left(\alpha_{\varphi,\xi} + \frac{\beta_{\varphi,\xi}\,\alpha_{\varphi,x}}{1-\beta_{\varphi,x}}\right)\boldsymbol{v}_{\varphi,0} + \left(\gamma_{\varphi,\xi} + \frac{\beta_{\varphi,\xi}\,\gamma_{\varphi,x}}{1-\beta_{\varphi,x}}\right)\boldsymbol{\xi}_{\varphi,T} + \frac{\beta_{\varphi,\xi}\,\delta_{\varphi,x}}{1-\beta_{\varphi,x}}\,\boldsymbol{x}_{\varphi,0} - \frac{\beta_{\varphi,\xi} b^2}{1-\beta_{\varphi,x}}\,\boldsymbol{g}. \tag{43}$$

The generated trajectories for two running steps are presented in Fig. 5, showing all the characteristics of the 3D-DCM framework discussed so far. All trajectories are continuous after the introduction of the flight-to-stance and stance-to-flight phases. The eCMP and the VRP are identical on the x- and y-axes, and separated by a fixed vertical offset on the z-axis throughout the motion.[3] During the flight phase, the CoM and the eCMP are identical, while on the x- and y-axes the offset between the CoM and the DCM is constant, corresponding to a constant CoM velocity.

G. Kinematic Reachability Constraint

In general, humanoid robotic walking avoids kinematic reachability limits by employing a constant CoM height above ground and limiting the step length. When walking with longer steps, switching to a toe contact at the end of the stance phase and using a heel contact at touchdown are common solutions for avoiding knee singularities and kinematic reachability limits. In contrast, humanoid running typically employs longer step lengths, while the existence of the flight phases induces a significant vertical motion of the DCM and CoM. In particular, the motion parameter Δz, denoting the vertical offset between VRP and eCMP, can no longer be chosen and interpreted as the average height of the CoM above ground, as in the case of bipedal walking. As can be observed in the z-axis plot of Fig. 5, for running, the VRP is notably lower than the CoM and DCM trajectories during the stance phases; in this example, a VRP offset of 0.476 m creates a CoM trajectory with an average height above ground of 0.87 m. Therefore, for bipedal running and jumping, the value of Δz needs to be chosen such that the kinematic reachability constraints are not violated during the stance phases. In this section, we propose a method of computing Δz by specifying the maximum distance between CoM and contact point at touchdown and takeoff (see Fig. 6). The choice of this metric is motivated by the fact that the motion planner generates trajectories for a reduced CoM model with no kinematic body.

Given a footstep $\boldsymbol{p}$ and a step vector $\boldsymbol{l}$, we construct a running cycle consisting of four phases: flight-to-stance, stance, stance-to-flight, and flight. During the stance phase, the VRP $\boldsymbol{v}$ is chosen to have a fixed position, with the vertical offset Δz above the footstep $\boldsymbol{p}$. The coordinate system is defined such that p_x and p_z are both 0, p_y corresponds to the footstep offset from the robot sagittal plane, and the x and y components of the step vector $\boldsymbol{l}$ are positive. The DCM waypoints $\boldsymbol{\Xi} = \begin{bmatrix}\boldsymbol{\xi}_1 & \dots & \boldsymbol{\xi}_5\end{bmatrix}^T$, and the CoM waypoints $\boldsymbol{X} = \begin{bmatrix}\boldsymbol{x}_1 & \dots & \boldsymbol{x}_5\end{bmatrix}^T$ can be obtained as closed-form functions of Δz by solving (30) with the following boundary conditions:

$$\boldsymbol{\xi}_f = \boldsymbol{\xi}_5 = \boldsymbol{T}_r\,\boldsymbol{\xi}_1 + \boldsymbol{l} \tag{44}$$

$$\boldsymbol{x}_s = \boldsymbol{x}_1 = \boldsymbol{T}_r\,(\boldsymbol{x}_5 - \boldsymbol{l}) \tag{45}$$

[3]This explains why the eCMP is not visible in Fig. 5 in the x- and y-plots.

where $\boldsymbol{T}_r = \begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$. Note that both the DCM and the CoM waypoints change sign on the y-axis; this results from the alternate succession of left and right footsteps during running. For jumping, the procedure is similar, with two differences regarding the y-axis: the VRP is placed in the center of the support polygon, i.e. $v_y = 0$, and the transformation matrix $\boldsymbol{T}_r$ in (44) and (45) is replaced by the identity matrix, as there are no alternating footsteps while jumping.

The distance between the CoM waypoint at touchdown $\boldsymbol{x}_1$ and the contact point $\boldsymbol{p}$, can be written as

$$d_{td}(\Delta z) = \|\boldsymbol{x}_1(\Delta z) - \boldsymbol{p}\| \tag{46}$$

i.e., as a closed-form function of Δz. Therefore, we can also compute the derivative of d_{td} with respect to Δz in closed-form, and solve $d_{td} = d^{\max}$ iteratively using Newton's method, where $d^{\max}$ is a configuration parameter denoting the kinematic reachability limit. We proceed similarly for the distance between the CoM waypoint and the contact point at takeoff

$$d_{to}(\Delta z) = \|\boldsymbol{x}_4(\Delta z) - \boldsymbol{p}\|. \tag{47}$$

Finally, the vertical offset Δz is chosen such that kinematic reachability limit is enforced for both waypoints.

Remark 5: If the durations of the flight-to-stance and stance-to-flight phases are equal, then the CoM motion with respect to the contact point $\boldsymbol{p}$ is symmetric, and the two distances discussed above are identical $d_{td} = d_{to}$.

V. Gait Transitions

Motivated by the goal of generating continuous reference trajectories, we propose explicit transition phases for connecting the four mentioned modes: standing, walking, running, and jumping. In this section, the transitions between standing, walking, and running are discussed in detail. They can be applied with small modifications to jumping, as the differences between running and jumping manifest themselves only on the y-axis.

A. Standing-to-Running

In our framework, the standing state is characterized by a stationary DCM ($\dot{\boldsymbol{\xi}} = \boldsymbol{0}$), which corresponds to the DCM being equal to the VRP, according to (7). In contrast to the commonly used definition for standing, the CoM is not required to be stationary, as its stable dynamics means that the CoM converges asymptotically to the DCM position at the end of motion sequence. The standing-to-running transition is implemented using two motion phases: the first phase, called a *stand-to-move* phase, ensures the continuity of the DCM trajectory during the transition; the second phase, called a *height-change* motion phase, lowers the VRP from the initial CoM height to the relative height above ground computed for the running gait using the kinematic reachability constraint. For both phases, we aim to write the DCM start point $\boldsymbol{\xi}_{\varphi,0}$ and the CoM end point $\boldsymbol{x}_{\varphi,T}$ in their respective general forms given as (22) and (23), which, as shown in Section IV-D, is a precondition for employing the waypoint computation algorithm.

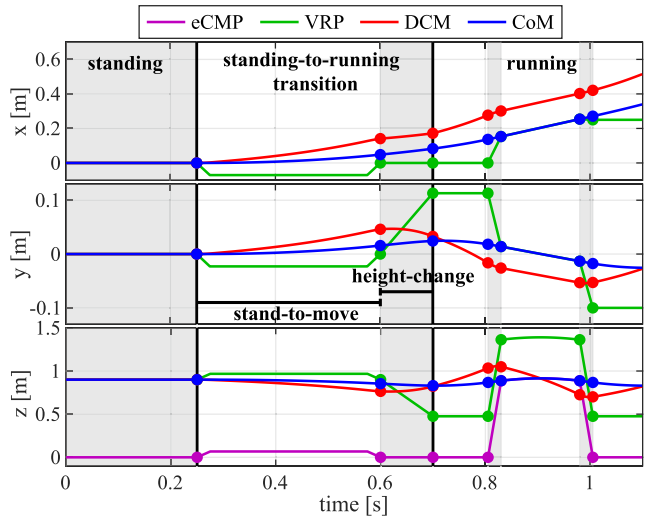


Fig. 7. Standing-to-running transition example. The individual motion phases are depicted using alternating gray and white backgrounds. The dots indicate the start and end waypoints of each phase.

Stand-to-Move Phase: During the stand-to-move phase, the VRP is shifted by a computed amount $\Delta\boldsymbol{v}$ which ensures that, for a given DCM final point $\boldsymbol{\xi}_{\varphi,T}$, the initial DCM position $\boldsymbol{\xi}_{\varphi,0}$ coincides with the initial VRP $\boldsymbol{v}_{\varphi,0}$ (see Fig. 7). At the end of the stand-to-move phase, the VRP returns to its initial position, i.e., $\boldsymbol{v}_{\varphi,T} = \boldsymbol{v}_{\varphi,0}$. This phase has two parameters: the duration T_{shift} of the VRP shift, occurring at the start and at the end of the phase, and the amount of time T_{hold} that the VRP remains stationary; the total phase duration is $T_\varphi = T_{\text{hold}} + 2\,T_{\text{shift}}$. The described VRP motion can be also interpreted as a sequence of three subphases: a shift from $\boldsymbol{v}_{\varphi,0}$ to a yet undetermined VRP position $\boldsymbol{v}_{\varphi,1} = \boldsymbol{v}_{\varphi,0} + \Delta\boldsymbol{v}$, a holding subphase at $\boldsymbol{v}_{\varphi,1}$ with duration T_{hold}, and a final shift back to the initial position. Writing the DCM waypoint equations for the three subphases using (12), imposing the standing state condition at the start of phase ($\boldsymbol{\xi}_{\varphi,0} = \boldsymbol{v}_{\varphi,0}$), and solving for $\Delta\boldsymbol{v}$ yields the closed-form solution

$$\Delta\boldsymbol{v} = \frac{T_{\text{shift}}}{b} \frac{e^{-T_\varphi/b}}{\left(1 - e^{-T_{\text{shift}}/b}\right)\left(1 - e^{-(T_{\text{shift}}+T_{\text{hold}})/b}\right)} (\boldsymbol{v}_{\varphi,0} - \boldsymbol{\xi}_{\varphi,T}). \tag{48}$$

The CoM end point equation for the stand-to-move phase is obtained by writing the CoM waypoint equations for the three subphases using (14), combining them with the DCM waypoint equations, and replacing $\Delta\boldsymbol{v}$ with the result from (48). Remarkably, the resulting equation simplifies to

$$\boldsymbol{x}_{\varphi,T} = \frac{1 - e^{-\frac{T_\varphi}{b}}}{2}\boldsymbol{v}_{\varphi,0} + \frac{1 - e^{-\frac{T_\varphi}{b}}}{2}\boldsymbol{\xi}_{\varphi,T} + e^{-\frac{T_\varphi}{b}}\boldsymbol{x}_{\varphi,0}. \tag{49}$$

Compared to the standing-to-walking transition introduced in [40], where an additional VRP waypoint was inserted into the VRP matrix $\boldsymbol{V}$ and computed by inverting a square $(n_w \times n_w)$ matrix, the proposed stand-to-move phase in this work has two advantages. First, at the end of the phase, the VRP returns to its

start position, keeping $\boldsymbol{V}$ fully determined, which enables the usage of the efficient waypoint computation algorithm presented above. Moreover, the computation of $\Delta\boldsymbol{v}$ requires only local information such as the start VRP position $\boldsymbol{v}_{\varphi,0}$, the DCM final point $\boldsymbol{\xi}_{\varphi,T}$, and the durations T_{shift} and T_{hold}. Second, given the same phase duration T_φ and DCM final point $\boldsymbol{\xi}_{\varphi,T}$, the proposed stand-to-move phase requires a significantly smaller shift in the VRP position than the method from [40]. For the example given in Fig. 7, the VRP shift is reduced by a factor of approximately 2.2. The reduction in the VRP movement increases the feasibility and the robustness of the overall motion, as the VRP remains closer to the center of the support area in the xy-plane during this phase.

Height-Change Phase: The height-change phase is characterized by the vertical motion of the VRP, which can be combined with a stationary VRP or a linearly interpolated motion on the x- and y-axes.[4] In the standing-to-running transition, the VRP remains stationary on the x-axis, while on the y-axis it moves from the middle of the support polygon corresponding to the double support stance toward the leg that acts as the first stance leg during the running motion (see Fig. 7). The running-to-standing and the walking-to-running transitions also employ a height-change phase, but the VRP motion on the y-axis is different than the one used here, as explained in the corresponding sections.

During the height change phase, the VRP needs to be lowered on the z-axis from the starting height corresponding to the standing stance to the relative height above ground computed for the running gait cycle. At the same time, the eCMP should maintain its position on the z-axis (see Fig. 7), such that the ground reaction forces are focused at the contact point. This relative motion leads to a change in the offset Δz between the VRP and eCMP, which corresponds to a changing DCM time factor b. However, in all equations so far, b was chosen to be constant, as the DCM dynamics has no closed-form solution for a time-varying b. Therefore, we propose to implement the height change phase using a piecewise constant approach for Δz and b, where the discretization time interval Δt can be chosen arbitrarily small, for example, equal to the controller execution time interval.

First, we investigate the effect of an instantaneous change in the DCM time constant b on the DCM and VRP positions. For the same CoM position $\boldsymbol{x}$ and velocity $\dot{\boldsymbol{x}}$, two DCM positions with different time constants b and b' can be written as the DCM definition (6) and

$$\boldsymbol{\xi}' = \boldsymbol{x} + b'\,\dot{\boldsymbol{x}}. \tag{50}$$

Replacing $\dot{\boldsymbol{x}}$ in (50) with the corresponding value from (6) leads to the DCM conversion formula

$$\boldsymbol{\xi}' = \left(1 - \frac{b'}{b}\right)\boldsymbol{x} + \frac{b'}{b}\,\boldsymbol{\xi}. \tag{51}$$

[4]The VRP trajectory can be designed independently on the three Cartesian axes.

Similarly, starting from the VRP definition (4) and equating $\ddot{\boldsymbol{x}}$ in both equations, the VRP conversion formula is found to be

$$\boldsymbol{v}' = \left(1 - \frac{b'^2}{b^2}\right)\boldsymbol{x} + \frac{b'^2}{b^2}\,\boldsymbol{v} \tag{52}$$

where $\boldsymbol{v}'$ denotes the VRP corresponding to the DCM time constant b'. Intuitively, the conversion formulas can be understood as a split of the DCM and VRP into their component parts, followed by a reassembly using a different time constant. In the process, the DCM and VRP computation becomes dependent on the instantaneous CoM position, while the DCM and VRP trajectories become discontinuous at the moment of the time constant change. Note that, while the DCM and VRP trajectories are discontinuous, the generated CoM trajectory remains smooth, i.e., the CoM position, velocity and acceleration are continuous throughout the motion.

In our framework, consisting of multiple motion phases, the DCM time constant change is implemented at the phase boundaries. For a phase φ, we denote by $\boldsymbol{v}'_{\varphi,0}$ and $\boldsymbol{\xi}'_{\varphi,0}$ the VRP and DCM waypoints $\boldsymbol{v}_{\varphi,0}$ and $\boldsymbol{\xi}_{\varphi,0}$ after the transformation from the local DCM time constant b_φ to the one of the preceding phase $b_{\varphi-1}$. We can then link adjacent phases with $\boldsymbol{v}_{\varphi-1,T} = \boldsymbol{v}'_{\varphi,0}$ and $\boldsymbol{\xi}_{\varphi-1,T} = \boldsymbol{\xi}'_{\varphi,0}$, and employ the waypoint computation algorithm. Applying (51) and (52) with $b = b_\varphi$ and $b' = b_{\varphi-1}$ to the DCM waypoint (22) produces the following result:

$$\begin{aligned}\boldsymbol{\xi}'_{\varphi,0} = {}& \frac{\alpha_{\varphi,\xi}}{r_\varphi}\,\boldsymbol{v}'_{\varphi,0} + r_\varphi\,\beta_{\varphi,\xi}\,\boldsymbol{v}_{\varphi,T} + r_\varphi\,\gamma_{\varphi,\xi}\,\boldsymbol{\xi}_{\varphi,T} \\ & + \left(1 - r_\varphi\,(1 - \alpha_{\varphi,\xi} - \delta_{\varphi,\xi}) - \frac{\alpha_{\varphi,\xi}}{r_\varphi}\right)\boldsymbol{x}_{\varphi,0} \\ & + r_\varphi\,\varepsilon_{\varphi,\xi}\,\boldsymbol{g}\end{aligned} \tag{53}$$

where $r_\varphi = b_{\varphi-1}/b_\varphi$ denotes the ratio between the DCM time constants. The CoM end point equation is obtained similarly by applying (52) to (23), which yields

$$\begin{aligned}\boldsymbol{x}_{\varphi,T} = {}& \frac{\alpha_{\varphi,x}}{r_\varphi^2}\,\boldsymbol{v}'_{\varphi,0} + \beta_{\varphi,x}\,\boldsymbol{v}_{\varphi,T} + \gamma_{\varphi,x}\,\boldsymbol{\xi}_{\varphi,T} \\ & + \left(\alpha_{\varphi,x} + \delta_{\varphi,x} - \frac{\alpha_{\varphi,x}}{r_\varphi^2}\right)\boldsymbol{x}_{\varphi,0} + \varepsilon_{\varphi,x}\,\boldsymbol{g}.\end{aligned} \tag{54}$$

Remark 6: The waypoint equations (53) and (54) can also be used for a sequence of running steps with different time parametrization for each step. Following the approach proposed in Section IV-G, for each step, a distinct offset Δz is computed based on the stance and flight duration parameters. The instantaneous change of the DCM time constant can be implemented most conveniently at the start of the flight phase, as the waypoint equations (53) and (54) simplify significantly due to the fact that the coefficients $\alpha_{\varphi,\xi}$, $\beta_{\varphi,\xi}$, $\alpha_{\varphi,x}$, and $\beta_{\varphi,x}$ are equal to 0 for the flight phase.

Using the equations derived above, the height-change phase is implemented as a sequence of $n_h = T_\varphi/\Delta t$ subphases, where T_φ is the total phase duration, and Δt is the chosen discretization interval. During each subphase $i \in \{1, \ldots, n_h\}$, the vertical

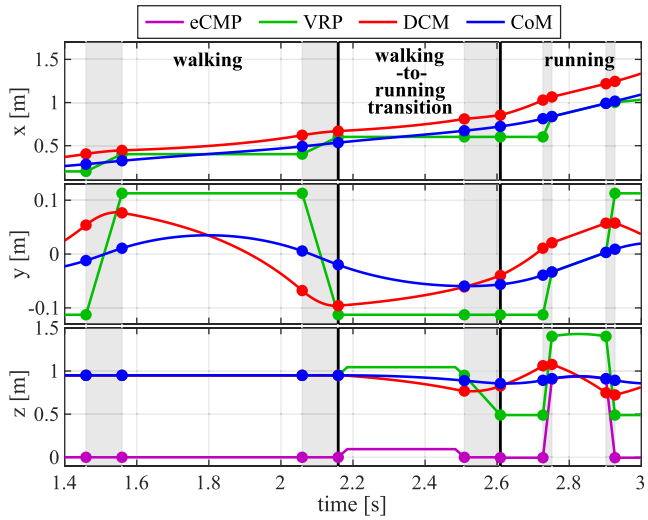


Fig. 8. Walking-to-running transition example. The individual motion phases are depicted using alternating gray and white backgrounds. The dots indicate the start and end waypoints of each phase.

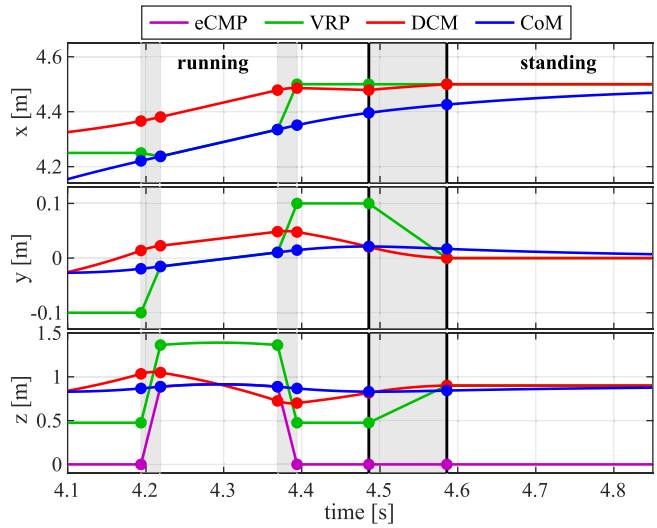


Fig. 9. Running-to-standing transition example. The individual motion phases are depicted using alternating gray and white backgrounds, the dots indicate the start and end waypoints of each phase. The start and end of the running-to-standing transition are highlighted with black vertical lines.

offset $\Delta z_{\varphi,i}$ is constant and equal to

$$\Delta z_{\varphi,i} = \Delta z_{\varphi,0} + \frac{i}{n_h}(\Delta z_{\varphi,T} - \Delta z_{\varphi,0}) \tag{55}$$

where $\Delta z_{\varphi,0}$ and $\Delta z_{\varphi,T}$ denote the offset at the start and the end of the height-change phase, respectively. The VRP motion can be described as a linear interpolation over a sequence of VRP waypoints $\boldsymbol{v}_{\varphi,i}$ that can be written with respect to the start and end points, $\boldsymbol{v}_{\varphi,0}$ and $\boldsymbol{v}_{\varphi,T}$, as

$$\boldsymbol{v}_{\varphi,i} = \frac{n_h - i}{n_h}\boldsymbol{v}_{\varphi,0} + \frac{i}{n_h}\boldsymbol{v}_{\varphi,T}. \tag{56}$$

Applying (53) and (54) to each subphase, the DCM start point coefficients for the height-change phase, $\alpha_{\varphi,\xi}$ to $\varepsilon_{\varphi,\xi}$, and the CoM end point coefficients, $\alpha_{\varphi,x}$ to $\varepsilon_{\varphi,x}$, can be computed using an algorithm similar to Algorithm 1. The only difference is that instead of DCM and CoM waypoints, the algorithm keeps track of the waypoint coefficients during the backward and forward iterations over the n_h subphases. The computational complexity of the algorithm is $\mathcal{O}(n_h^2)$.

B. Walking-to-Running

On the z-axis, the walking-to-running transition (see Fig. 8) is implemented in a similar manner to the standing-to-running transition, with the VRP trajectory consisting of a stand-to-move and a height-change phase. On the x- and y-axes, the VRP is stationary, as the walking-to-running transition is designed to begin after a double support phase, such that the subsequent single support phase of the walking gait seamlessly transforms into the first stance phase of the running gait. Note, however, that despite being stationary, the VRP trajectory on the x- and y-axes consists of three distinct phases: the walking single support, the height-change, and the running stance phase. Each phase computes the DCM and CoM waypoints using different equations or different DCM time constants b.

C. Running-to-Standing

The running-to-standing transition consists of a single height-change phase during which the VRP is raised from the running gait height to the level corresponding to the standing stance (see Fig. 9). On the x-axis, the VRP remains stationary, while on the y-axis it moves from the last footstep position to the center of support polygon, where it remains stationary during the standing state. Note that, at the end of the running-to-standing transition, the DCM is stationary, while the CoM converges asymptotically to the final VRP and DCM common waypoint.

D. Running-to-Walking

The running-to-walking transition is implemented similarly to both the running-to-standing and the walking-to-running transitions. On the z-axis, the VRP trajectory is the same as during the running-to-standing transition, while on the x- and y-axes, the last stance phase of the running gait seamlessly transforms into the first single support phase of the walking gait (see Fig. 10).

VI. Whole-Body Motion Generation and Control

The proposed CoM reference trajectory planner is integrated with the whole-body motion generation and control methods presented in our previous works. The swing leg reference trajectory including utilization of edge contacts (toe-off) is described in [31], while the angular momentum generation method from [39] is used to compensate the angular momentum induced by the motion of the legs during running and jumping. The generated reference trajectories are tracked by the passivity-based whole-body controller presented in detail in [31] and [41]. In this section, we give a short overview of the angular momentum generation method, describing the extension introduced in this work for handling the flight phase.

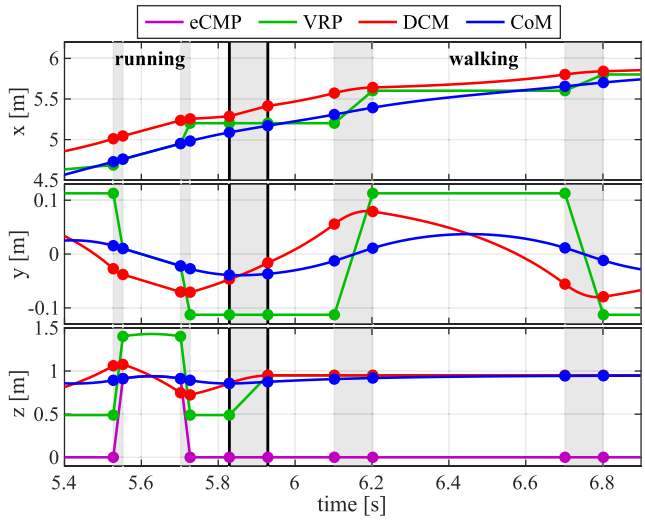


Fig. 10. Running-to-walking transition example. The individual motion phases are depicted using alternating gray and white backgrounds, the dots indicate the start and end waypoints of each phase. The start and end of the running-to-walking transition are highlighted with black vertical lines.

A. Dynamic Model

Following the approach taken in [42], the humanoid robot dynamics is described using a floating base model with the CoM position $\boldsymbol{x} \in \mathbb{R}^3$ and the waist orientation $\boldsymbol{R}_b \in \mathcal{SO}(3)$ as base coordinates. The corresponding linear and angular velocities, $\dot{\boldsymbol{x}} \in \mathbb{R}^3$ and $\boldsymbol{\omega}_b \in \mathbb{R}^3$, are stacked into the velocity vector $\boldsymbol{\nu} = (\dot{\boldsymbol{x}}^T \ \boldsymbol{\omega}_b^T)^T$. For a humanoid robot with n actuated joints, the equations of motion can be written as

$$\boldsymbol{M}\begin{pmatrix}\dot{\boldsymbol{\nu}}\\ \ddot{\boldsymbol{q}}\end{pmatrix} + \boldsymbol{C}\begin{pmatrix}\boldsymbol{\nu}\\ \dot{\boldsymbol{q}}\end{pmatrix} + \begin{pmatrix}-\boldsymbol{w}_g\\ \boldsymbol{0}_{n\times 1}\end{pmatrix} = \begin{pmatrix}\boldsymbol{0}_{6\times 1}\\ \boldsymbol{\tau}\end{pmatrix} + \bar{\boldsymbol{\tau}}_{\text{ext}} \tag{57}$$

where $\boldsymbol{M}$ and $\boldsymbol{C}$ represent the inertia and Coriolis matrices, respectively, $\boldsymbol{w}_g = (m\boldsymbol{g}^T \ \boldsymbol{0}_{3\times 1}^T)^T \in \mathbb{R}^6$ denotes the gravitational wrench, $\boldsymbol{q} \in \mathbb{R}^n$ the joint positions, $\boldsymbol{\tau} \in \mathbb{R}^n$ is the vector of actuator torques, while $\bar{\boldsymbol{\tau}}_{\text{ext}} \in \mathbb{R}^{6+n}$ denotes the generalized external forces.

The task space consists of the CoM position, the waist orientation, the Cartesian position and orientation of the feet, and the joint positions of the upper body [31]. The task velocity vector can be written as

$$\underbrace{\begin{pmatrix}\boldsymbol{\nu}\\ \boldsymbol{\nu}_R\\ \boldsymbol{\nu}_L\\ \dot{\boldsymbol{q}}_u\end{pmatrix}}_{\dot{\boldsymbol{x}}_{task}} = \underbrace{\begin{bmatrix}\boldsymbol{I}_{6\times 6} & \boldsymbol{0}_{6\times n}\\ \boldsymbol{Ad}_R & \boldsymbol{J}_R\\ \boldsymbol{Ad}_L & \boldsymbol{J}_L\\ \boldsymbol{0}_{n_u\times 6} & \boldsymbol{S}_u\end{bmatrix}}_{\boldsymbol{J}} \begin{pmatrix}\boldsymbol{\nu}\\ \dot{\boldsymbol{q}}\end{pmatrix} \tag{58}$$

where $\boldsymbol{\nu}_R \in \mathbb{R}^6$ denotes the velocity vector, combining translational and rotational velocities, $\boldsymbol{Ad}_R \in \mathbb{R}^{6\times 6}$ is the adjoint matrix, and $\boldsymbol{J}_R \in \mathbb{R}^{6\times n}$ the Jacobian matrix for the right foot. Analogous quantities for the left foot are denoted by the subscript "L". The velocities of the n_u upper body joints are indicated by $\dot{\boldsymbol{q}}_u \in \mathbb{R}^{n_u}$, with $\boldsymbol{S}_u \in \mathbb{R}^{n_u\times n}$ selecting the corresponding joints from the complete vector $\dot{\boldsymbol{q}}$.

B. Angular Momentum Compensation

The CAM $\boldsymbol{l}_c \in \mathbb{R}^3$, representing the angular momentum expressed around the CoM, depends linearly on the velocity vector

$$\boldsymbol{l}_c = \boldsymbol{M}_\omega \begin{pmatrix}\boldsymbol{\nu}\\ \dot{\boldsymbol{q}}\end{pmatrix} = \underbrace{\boldsymbol{M}_\omega \boldsymbol{J}^{\#}}_{\bar{\boldsymbol{A}}} \dot{\boldsymbol{x}}_{\text{task}} \tag{59}$$

where $\boldsymbol{M}_\omega$ is the rotational part (i.e., the rows 4 to 6) of the inertia matrix $\boldsymbol{M}$ [43]. Here, $\boldsymbol{J}^{\#}$ is a generalized damped pseudoinverse of the task space Jacobian that is robust to singular configurations

$$\boldsymbol{J}^{\#} = \boldsymbol{J}^T(\boldsymbol{J}\boldsymbol{J}^T + \rho^2\boldsymbol{I})^{-1} \tag{60}$$

with $\rho \in \mathbb{R}$ as a damping factor.

The whole-body motion optimization [39] uses a subset of the task space degrees of freedom (DoF) to actively generate angular momentum with the goal of inducing a reference CAM $\boldsymbol{l}_c^{\text{ref}}$ to the system. In this work, we choose these DoF to be the waist orientation $\boldsymbol{R}_b$ and the upper body joints $\boldsymbol{q}_u$ excluding the wrist joints, as their contribution to the total angular momentum is negligible. Let $\dot{\boldsymbol{x}}_a$ denote the velocity vector, and $\bar{\boldsymbol{A}}_a$ the matrix obtained by selecting the corresponding rows from $\bar{\boldsymbol{A}}$ for these DoF. Given a task space reference velocity $\dot{\boldsymbol{x}}_{\text{task}}^{\text{ref}}$ and a reference angular momentum $\boldsymbol{l}_c^{\text{ref}}$, the motion optimization finds optimal velocities $\dot{\boldsymbol{x}}_a^{\text{opt}}$ that fulfill the angular momentum task while minimizing the deviation from the reference task quantities (for details, see [39]).

For running and jumping, we generate the reference CAM individually for the various motion phases presented in this work. During the phases in which the robot is in contact with ground, such as stance phases or gait transition phases, $\boldsymbol{l}_c^{\text{ref}}$ is chosen to be

$$\boldsymbol{l}_c^{ref} = \bar{\boldsymbol{A}}_a \dot{\boldsymbol{x}}_a^{\text{ref}}. \tag{61}$$

Intuitively, (61) states that the angular momentum induced by the selected DoF is required for the planned motion and should not be compensated by the motion optimizer. Specifically, the reference CAM of the complete system is given only by the selected DoF, while the angular momentum induced by the remaining DoF (i.e., the swing leg motion) is undesirable and needs to be compensated. In many scenarios, the reference velocities in $\dot{\boldsymbol{x}}_a^{\text{ref}}$ are 0, such that the reference angular momentum is also 0 ($\boldsymbol{l}_c^{\text{ref}} = \boldsymbol{0}_{3\times 1}$). However, this is not always the case, such as when running on a curved trajectory (see Fig. 13).

During the flight phase, the total angular momentum is constant. In order to generate consistent trajectories for the whole-body motion, we choose the reference angular momentum to be equal to the measured angular momentum

$$\boldsymbol{l}_c^{\text{ref}} = \boldsymbol{l}_c. \tag{62}$$

During the stance-to-flight and the flight-to-stance phases, the reference angular momentum is linearly interpolated between

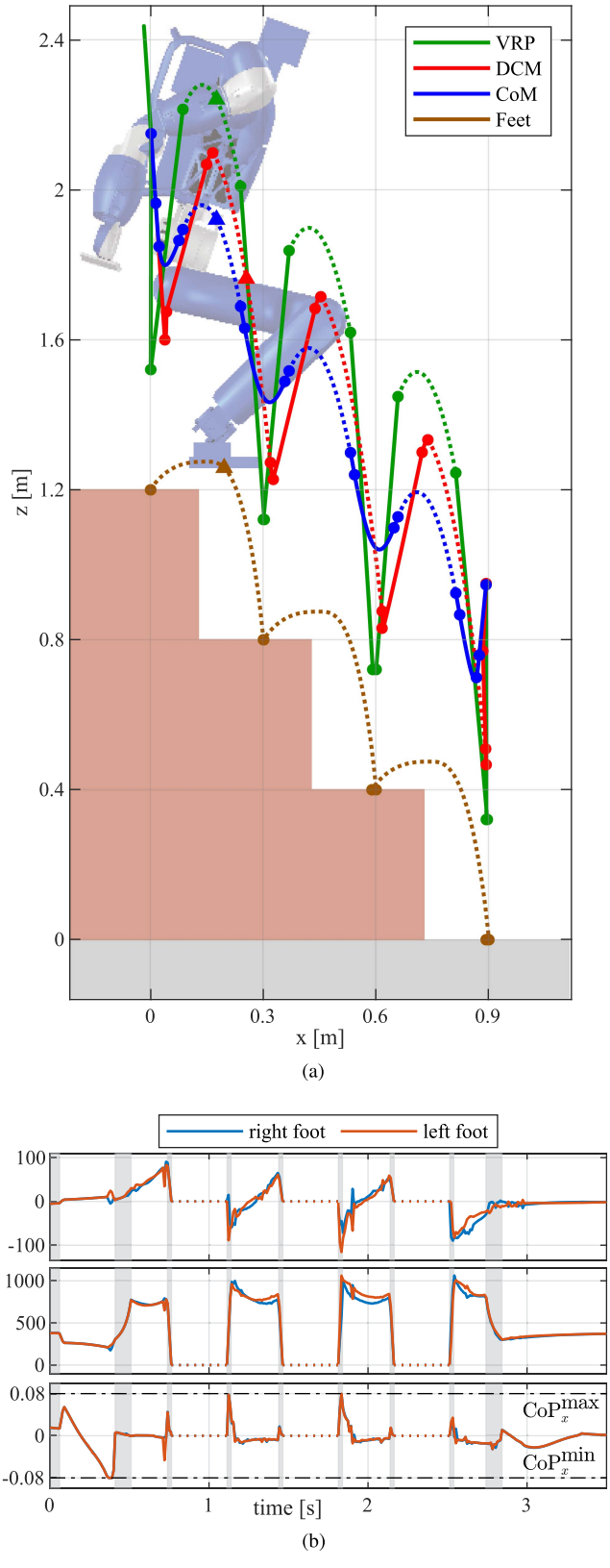


Fig. 11. Descending a stair with large steps by jumping. Each stair step is 30 cm long and 40 cm high. (a) Generated reference CoM trajectories (spatial side view). Continuous lines show the trajectories during stance phases; dotted lines are used during the flight phases. The dots mark the location of the start and end points of each individual motion phase, while the triangles mark the instantaneous quantities for the depicted robot. (b) Commanded forces on the x- and z-axes, and commanded center of pressure on the x-axis for both feet. Dotted lines denote flight phases.

the two reference generation schemes. For example, during the stance-to-flight phase, $\boldsymbol{l}_c^{\text{ref}}$ is generated by

$$\boldsymbol{l}_c^{ref} = \left(1 - \frac{t}{T_\varphi}\right) \bar{\boldsymbol{A}}_a \dot{\boldsymbol{x}}_a^{\text{ref}} + \frac{t}{T_\varphi} \boldsymbol{l}_c \tag{63}$$

where T_φ is the duration of the stance-to-flight phase, and $t \in [0, T_\varphi]$ is the local phase time.

VII. Simulations

We performed extensive simulations of the proposed CoM trajectory generator with the passivity-based whole-body controller in OpenHRP [44] using the torque controlled humanoid robot TORO [36], a 27 DoF robot with a height of 1.74 m and a total weight of 77.5 kg. Videos of the performed simulations can be found in the multimedia attachment. The trajectory generator and the whole-body controller are implemented in MATLAB/Simulink and are executed at a rate of 1 kHz. The realtime capability of the algorithms was verified on TORO's computation hardware, an Intel Core i7 computer; the total computation time varied between 380 and 540 μs, with the whole-body reference trajectory generator including the angular momentum optimization accounting for 75–90 μs.

In all simulation scenarios, the robot starts and ends in a standing stance, showcasing the ability of the proposed method to initiate and complete highly dynamic motions such as running and jumping, while maintaining balance throughout the motion. Unless stated otherwise, the default duration of the stance-to-flight and flight-to-stance phases is 25 ms, the duration of the height-change phase is 100 ms, while the maximum distance between CoM and contact point, used as a kinematic reachability constraint for computing the VRP offset Δz in Section IV-G, is $d^{\max} = 0.95$ m. The whole-body controller constrains the commanded ground reaction forces using a friction coefficient of 0.4 and center-of-pressure (CoP) bounds of ± 0.08 m on the x-axis and ± 0.035 m on the y-axis relative to the center of the foot, expressed in local foot coordinates. For comparison, TORO's foot is 19 cm long and 9.5 cm wide.

The first scenario (see Fig. 11) shows the robot performing a sequence of three jumps, each with a length of 30 cm and a height of 40 cm. The stance phase duration is set to 300 ms, the flight phase duration is 350 ms, while the kinematic reachability constraint used for computing the VRP offset Δz is chosen to be $d^{\max} = 0.9$ m. The high stair steps in this scenario are purposefully chosen to emphasize the need for using both legs simultaneously (i.e., employing a jumping gait), as the required force acting on the CoM exceeds 2000 N during landing [see Fig. 11(b)]. Fig. 11(b) also shows the importance of the stance-to-flight and flight-to-stance phases in preserving the continuity of the commanded contact forces. For example, at $t = 1.1$ s, during the flight-to-stance phase, the trajectory of the normal force f_z is similar to a linear interpolation from 0 to 1000 N, thereby increasing the feasibility of the motion and reducing the effort of tracking the commanded force trajectory.

The second scenario (see Fig. 12) shows a walking and running gait progression, with increasing forward velocity (i.e., on the x-axis). The motion starts with a stand-to-move phase

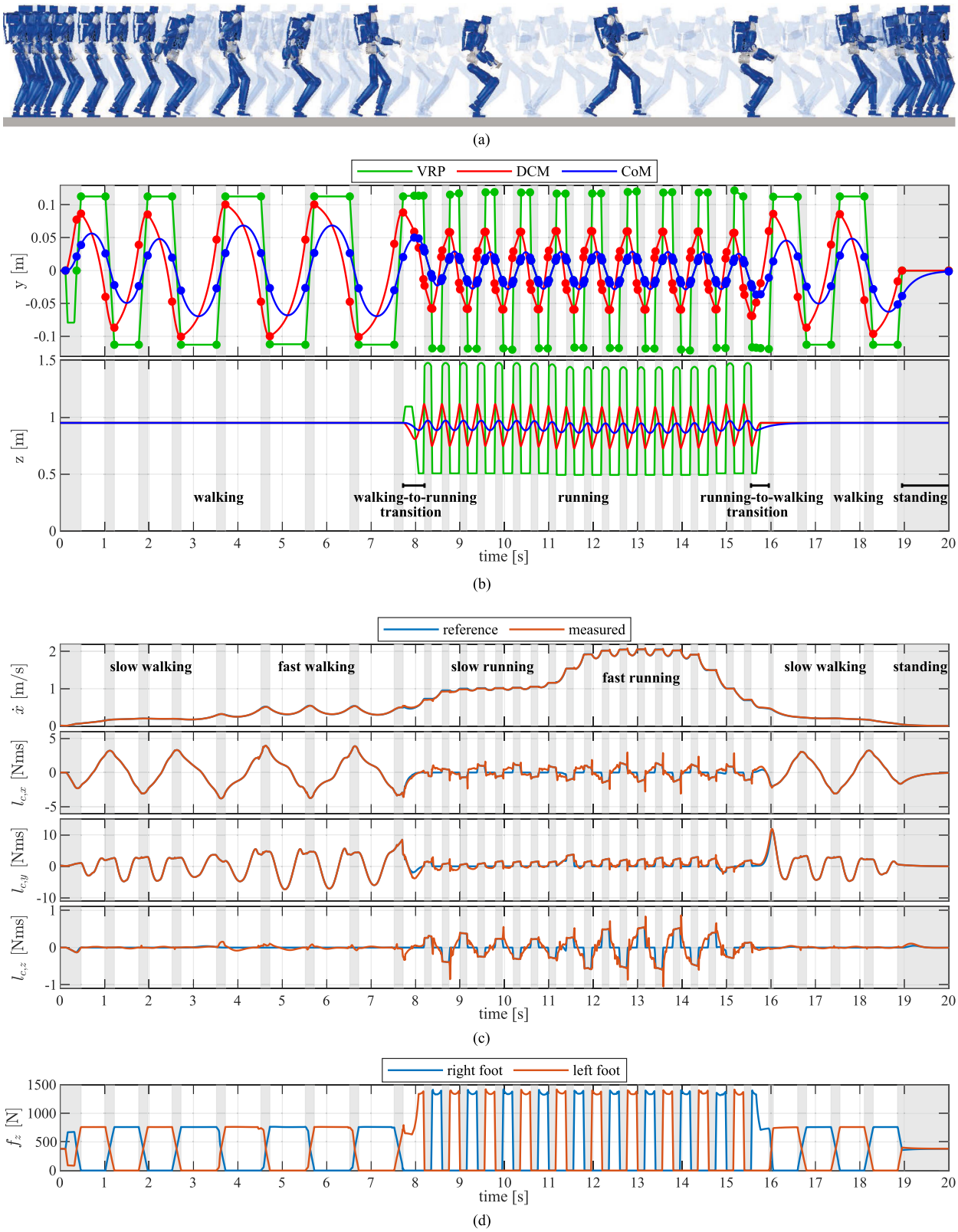


Fig. 12. Walking and running gait progression, starting with slow walking (0.2 m/s), followed by fast walking (0.4 m/s), slow running (1 m/s), and fast running (2 m/s). (a) Simulation images (solid images taken at 1 s intervals, translucent images at 0.2 s intervals during running and 0.5 s during walking). (b) Generated reference CoM trajectory (y- and z-axes), showing the start and end points of each motion phase on the y-axis. Gray background depicts double support phases during walking, and flight phases during running. (c) Reference and measured data: CoM velocity on the x-axis, CAM on all three axes. (d) Commanded forces (z-axis).

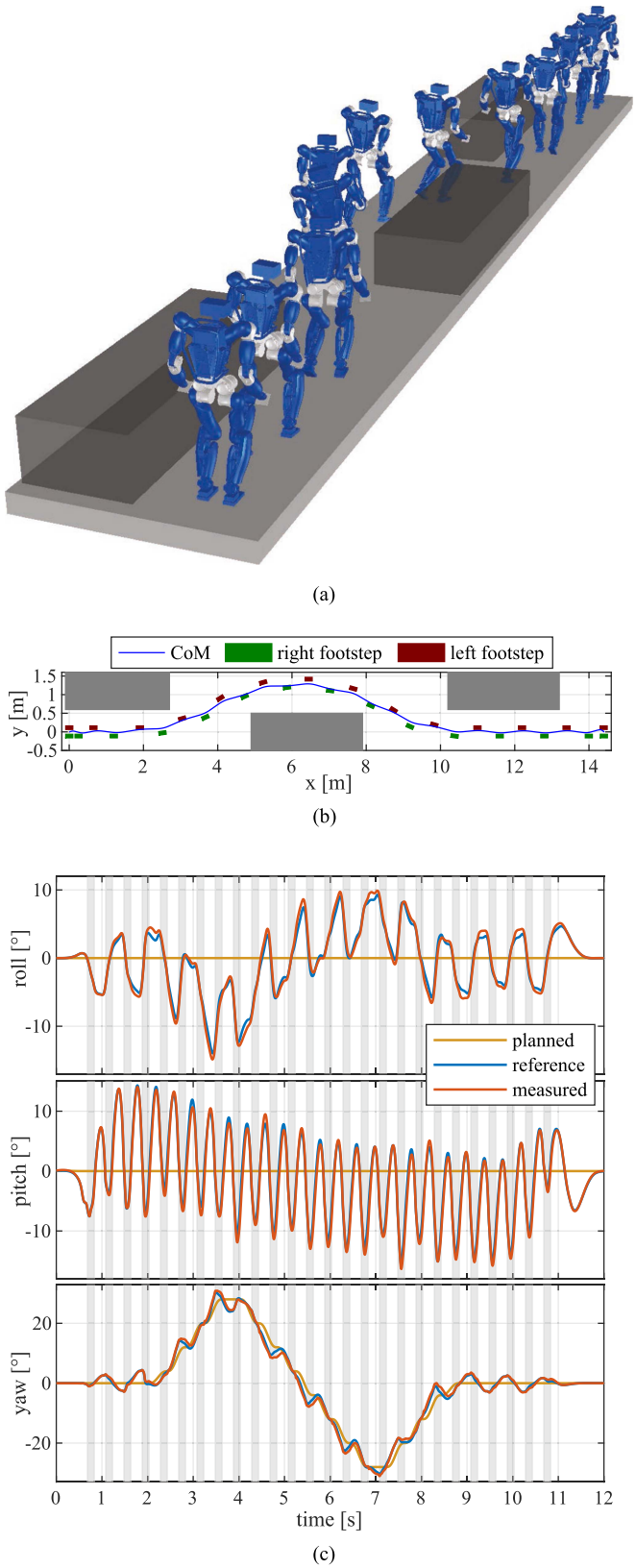
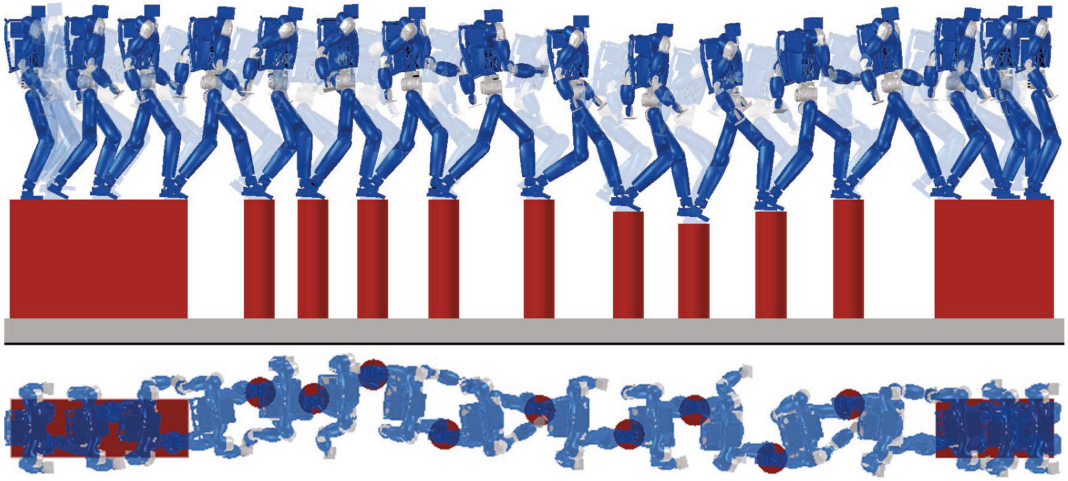


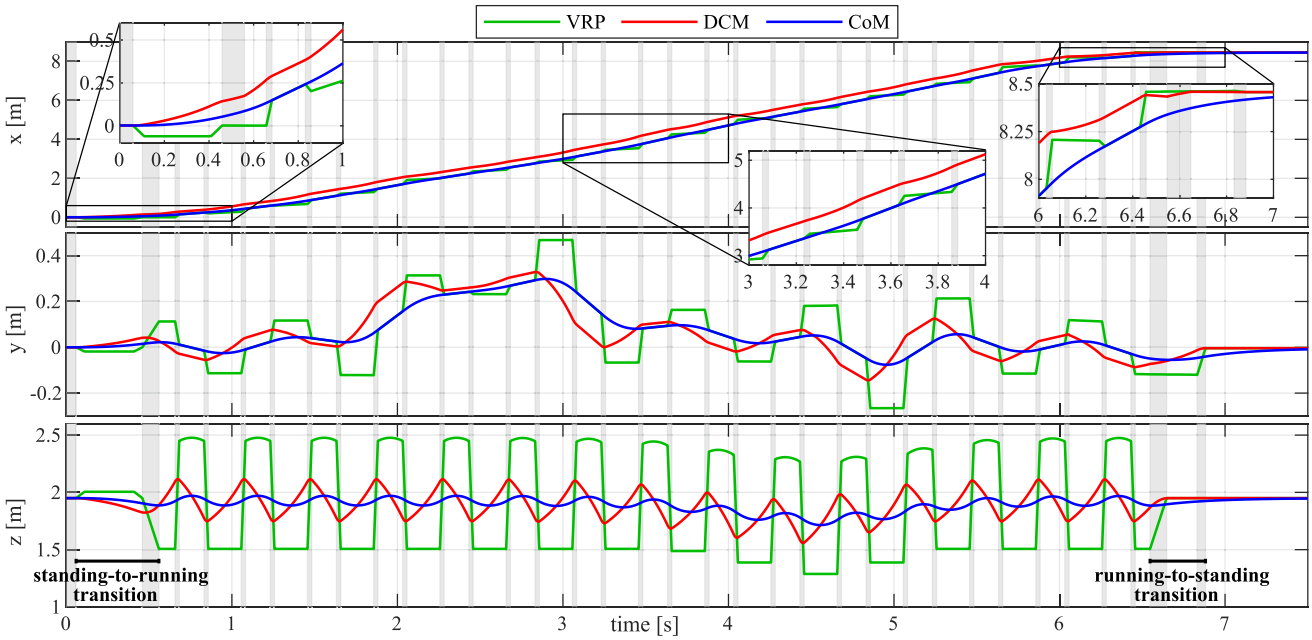
Fig. 13. Running on a curved trajectory, avoiding obstacles. (a) Simulation images, taken at 1 s intervals. (b) Planned footsteps and reference CoM trajectory (spatial top-down view). (c) Planned, reference, and measured waist orientation. The reference trajectory is generated by the angular momentum optimizer. Gray background depicts flight phases.

of 250 ms total duration ($T_{\text{shift}} = 50$ ms, $T_{\text{hold}} = 150$ ms, see Section V-A), which ensures the continuity of the trajectory while transitioning from the standing stance to the walking gait. The first three walking steps are 15 cm long and are executed with single- and double-support phase durations of $T_{SS} = 550$ ms and $T_{DS} = 200$ ms, corresponding to a walking speed of 0.2 m/s. For the next four steps, the step length is increased to 40 cm by employing the toe-off edge contacts [31], with a simultaneous increase of the single-support phase duration to $T_{SS} = 800$ ms, resulting in a walking speed of 0.4 m/s. During walking, the angular momentum compensation is disabled for the x- and y-axes, being active only on the z-axis [see Fig. 12(c)]. These settings significantly reduce the waist motion produced by the motion optimizer [39], as the required angular momentum on the x- and y-axes can be easily generated by the contact torques. At $t = 8$ s, the robot transitions from walking to a running gait using the walking-to-running transition method described in Section V-B. The running gait uses toe-off edge contacts, has a stance phase duration of 200 ms, and a flight phase duration of 150 ms, for a total step time of 400 ms, including the stance-to-flight and flight-to-stance phases. During the first 4 s of running, the step length is chosen to be 40 cm, corresponding to a running speed of 1 m/s, while at $t = 11.5$ s the step length is increased to 80 cm leading to the maximum running speed of 2 m/s. During running, the angular momentum is compensated on all three axes [see Fig. 12(c)]. Note the spikes in angular momentum caused by foot impacts with the ground at the end of the flight phases. At $t = 16$ s, the robot transitions back to walking using the running-to-walking transition method, before finally stopping at $t = 19$ s.

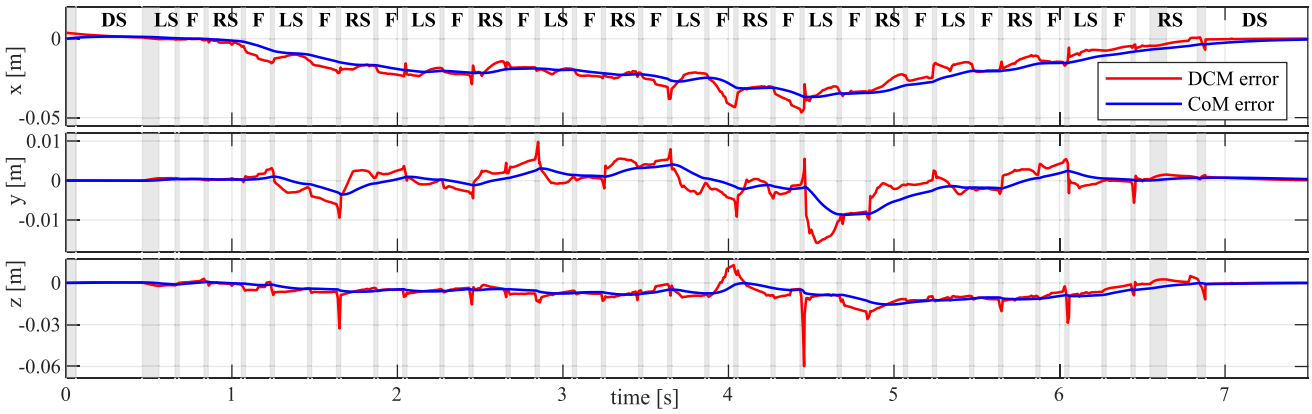
In the third scenario (see Fig. 13), the robot runs along a curved trajectory, avoiding large obstacles located in its path. The running gait uses the same stance and flight phase durations as the second scenario, while the standing-to-running transition employs a stand-to-move phase of 350 ms duration ($T_{\text{shift}} = 25$ ms, $T_{\text{hold}} = 300$ ms). The longer duration of this phase compared to the second scenario ensures that the VRP remains within the support area (see Section V-A) for the more dynamic type of motion (running vs. walking). During the first four running steps, the step length is gradually increased to 60 cm, which is subsequently used as the nominal running gait, corresponding to a running speed of 1.5 m/s. Starting with the fifth step, the footsteps are rotated in the xy-plane by 8° increments, turning first to the left (counterclockwise) for four steps, then to the right over the course of eight steps, and finally to the left for additional four steps, such that the final orientation is the same as the initial one [see Fig. 13(b)]. The yaw angle of the planned waist orientation follows the footstep orientations using fifth-order polynomial interpolations, while the reference trajectory is generated online by the angular momentum optimizer to compensate the effects of the leg motion. The maximum tracking error of the waist orientation over the whole motion is 1.7° for the roll angle, 2.4° for the pitch, and 2.2° for the yaw [see Fig. 13(c)], which demonstrates the feasibility of the reference trajectories and the tracking performance of the passivity-based whole-body controller.



(a)



(b)



(c)

Fig. 14. Running over a sequence of given stepping stones, placed with a large degree of variability on all three Cartesian axes. (a) Simulation images. Side view: Solid images show the robot during the flight phases, translucent images during the stance phases. Top-down view: Only flight phases are shown. (b) Generated reference CoM trajectory. The individual motion phases are depicted with alternating gray and white backgrounds. (c) DCM and CoM tracking errors. The motion phases are labeled with the corresponding contact configuration (DS: double support, LS: left leg support, RS: Right leg support, F: Flight phase).

In the fourth scenario (see Fig. 14), the robot traverses a sequence of nine stepping stones modeled as cylindrical columns 1 m in height and with a diameter of 26 cm. The small contact area provided by each column requires precise tracking of the CoM and foot trajectories in order to successfully traverse the given terrain. The robot starts on a platform 1.5 m long and 0.5 m wide, which is used to accelerate to the nominal running speed of 1.5 m/s, corresponding to a step length of 60 cm, stance phase duration of 200 ms, and flight phase duration of 150 ms. The columns are placed with a large degree of variability on all three Cartesian axes: on the x-axis, the distance between the columns varies between 45 and 85 cm, on the y-axis between 5 and 50 cm, and on the z-axis, the sixth and the eighth columns are lowered by 10 cm, while the seventh column is lowered by 20 cm. The final platform is chosen to be shorter than the start platform (1 m long vs. 1.5 m), such that the robot is required to decelerate rapidly after stepping onto the platform in order to come to a stop. The maximum tracking error of the CoM position over the whole motion is 3.7 cm on the x-axis, 0.9 cm on the y-axis, and 1.6 cm on the z-axis [see Fig. 14(c)]. The spikes in the DCM tracking error signal are caused by delayed contact acquisition after the flight phases; for example, at $t = 4.4$ s when the robot steps onto the seventh column, the spike is caused by a left foot tracking error on the z-axis of 2.5 cm. Nevertheless, after the contact is established, the DCM tracking error is quickly corrected by the whole-body controller.

TABLE I
INDUCED DISCONTINUITIES IN THE INSTANTANEOUS DCM COMPUTATION BY A DCM FINAL POINT CHANGE OF 0.6 M, FOR DIFFERENT LENGTHS OF THE MOTION PLAN'S PREVIEW WINDOW

Steps	Motion phases	Total duration	$\Delta\boldsymbol{\xi}_x$
1	10	1.0 s	0.1438 m
2	14	1.4 s	0.0389 m
3	18	1.8 s	0.0105 m
4	22	2.2 s	0.0028 m
5	26	2.6 s	0.0008 m

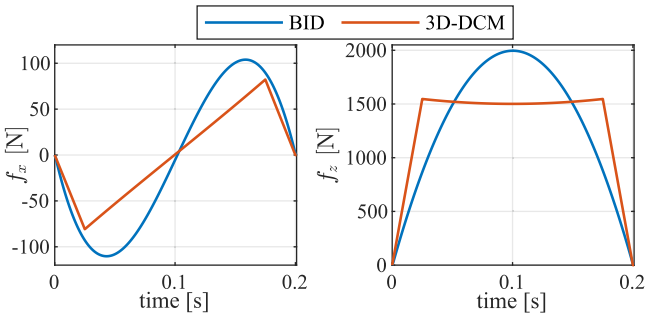


Fig. 15. BID versus 3D-DCM comparison of generated contact forces.

VIII. DISCUSSION

When discussing the advantages and limitations of the proposed method, we need to differentiate between the 3D-DCM framework and the specific trajectory generation method employed for running, jumping, and the various gait transitions.

A. 3D-DCM Framework: Advantages and Limitations

The 3D-DCM framework, as a reformulation of the CoM dynamics, retains its generality and can be used with any legged robot; unlike *simplified* models such as SLIP or LIPM, 3D-DCM is a *reduced* model of the whole-body dynamics. Taking advantage of the reverse time stability of the DCM dynamics, the framework can be used to generate stable reference trajectories for piecewise linearly interpolated VRP trajectories with arbitrary durations. Notably, the framework generates stable trajectories for arbitrarily slow motions, which are particularly challenging for methods that only consider the forward time dynamics of the DCM [26]. Furthermore, the 3D-DCM framework utilizes closed-form continuous-time trajectories, requiring no discretization of the dynamics, which enables trajectory generation at arbitrary sampling rates. For example, in our implementation, we compute the instantaneous CoM, DCM, and VRP every millisecond, i.e., at the same rate as the whole-body controller execution rate of 1 kHz.

A further advantage of the computation efficiency is that the framework can generate CoM and DCM trajectories for motions with long durations. For example, for running, we use a preview window of 5 steps, which corresponds to a plan duration of 2.6 s, including the running-to-standing transition at the end of the motion. The rationale for this choice is provided by the data given in Table I. Due to the reverse time computation approach, any change in the position of the DCM final point $\boldsymbol{\xi}_f$ leads to a discontinuity of the computed instantaneous DCM $\boldsymbol{\xi}$. The magnitude of the discontinuity is reduced by longer preview windows; for five steps, a DCM final point change on the x-axis of 0.6 m creates a DCM discontinuity, which is less than 1 mm. This choice is made possible by employing a highly efficient algorithm for waypoint computation: the DCM and CoM waypoints of the corresponding 26 motion phases can be computed in approx. 25 μs with Algorithm 1 by TORO's computing hardware.

Arguably, one limitation of the 3D-DCM framework is that it cannot produce a particular contact force profile such as the biologically inspired forces proposed by BID [16]. By its very nature, the 3D-DCM framework requires an eCMP trajectory (or associated VRP trajectory), for which it computes the corresponding CoM trajectory, while the contact force profile is subsequently given by (2). A comparison of the generated contact forces on the x- and z-axes for a running stance phase is shown in Fig. 15. Note that the required 3D-DCM forces are smaller in magnitude than the BID forces for the same motion parameters. On the other hand, although continuous, the 3D-DCM forces are not continuously differentiable, making them potentially more difficult to generate on robots employing elastic elements in the legs. Nevertheless, if needed, smoother forces profiles can be produced by the 3D-DCM framework using higher order polynomial interpolations in the eCMP reference trajectories. A further advantage of 3D-DCM is that using a common framework for walking and running greatly simplifies the gait transition implementation, as shown in Section V, whereas

combining BID for running with 3D-DCM for walking requires significantly more computation effort [17].

These characteristics of the 3D-DCM framework can be exploited by more computationally intensive methods like MPC or RL in order to reduce the total computational effort. For example, the reverse time DCM computation can be combined with MPC as shown in [34], and RL can be used to learn VRP offsets during the stance phase in order to generate a contact force profile that is more similar to BID. We intend to pursue these ideas in our future work.

B. Whole-Body Reference Trajectories

One important factor affecting the robustness of dynamic bipedal locomotion is the controller's ability of keeping the CoP close to the middle of the foot contact area during the stance phases. This can be achieved by suitably designed reference trajectories, or by using the upper body to generate angular momentum, thereby reducing the required contact torques. In this work, the eCMP reference trajectory is generated using a constant angular momentum assumption; the eCMP waypoints are either placed in the middle of the foot contact area, or, alternatively, the eCMP follows a heel-to-toe trajectory during each stance phase, which has the additional advantage of reducing the magnitude of the necessary contact forces on the local x-axis. The employed motion optimization [39] significantly reduces the CAM during running, as shown in the performed simulations [see Fig. 12(c)]. As part of our future work, we plan to include the online learning algorithm proposed in [45] to generate eCMP trajectories that lead to improved CoP tracking during the contact phases.

The leg reference trajectories are generated using simple fifth-order polynomials for each Cartesian translational and rotational axis. Despite their simplicity, the controller is able to track the generated motions without difficulty, as demonstrated by the performed simulations. Nevertheless, one challenge encountered during highly dynamic motions is avoiding knee singularities. At takeoff and touchdown, this problem is addressed by the proposed VRP offset computation in Section IV-G, which ensures that the distance between CoM and contact point is bounded by a configurable maximum value. However, during the flight phase, the independent computation of the leg and CoM trajectories can lead to the kinematic constraint being violated. We plan to address this issue in our future work.

One limitation of the angular momentum optimization is given by the instantaneous computation of the upper body motion without an explicitly planned angular momentum trajectory or a preview over a future time-horizon. One noticeable side-effect is the gradual upward movement of the arms over the span of multiple running steps, which can be seen during the fast running segment of the second simulation [see Fig. 12(a)].

Finally, the whole-body simulations demonstrate the feasibility of the generated trajectories with respect to maintaining balance throughout the motion, controlling the angular momentum, and conforming to contact constraints (friction cone, unilateral normal force, and limited contact area). However, the required joint velocities and torques exceed the capabilities of the real humanoid robot TORO, preventing us from demonstrating the proposed method in an experimental setup.

IX. Conclusion

In this work, we proposed a motion planner based on the 3D-DCM framework, capable of generating a CoM reference trajectory for motions that include flight phases, such as running and jumping. Taking advantage of the reverse time stability of the DCM dynamics, the motion planner creates stable reference trajectories for arbitrary contact sequences and time parametrization. Furthermore, using the 3D-DCM framework for running and jumping greatly simplifies the implementation of gait transitions, such as walking-to-running or running-to-walking transitions. The proposed motion planner was validated in various simulations with the humanoid robot TORO, demonstrating the feasibility of the generated trajectories.

Two key aspects of the proposed motion planner, the unified approach of describing different motion phases and the high efficiency in computing the reference trajectories, enhance the planner's versatility, making it well suited for usage on a wide variety of legged robots: bipeds, quadrupeds, or fully humanoid robots.

Acknowledgment

The authors would like to thank Jinoh Lee and Grazia Zambella for their valuable comments.

References

[1] T. Takenaka, T. Matsumoto, T. Yoshiike, and S. Shirokura, "Real time motion generation and control for biped robot -2nd report: Running gait pattern generation," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2009, pp. 1092–1099.

[2] K. Kojima et al., "A robot design method for weight saving aimed at dynamic motions: Design of humanoid JAXON3-P and realization of jump motions," in *Proc. IEEE-RAS Int. Conf. Humanoid Robots*, 2019, pp. 586–593.

[3] Boston Dynamics, "Atlas," 2023. Accessed: Oct. 10, 2023. [Online]. Available: https://www.bostondynamics.com/atlas

[4] J. Siekmann, Y. Godse, A. Fern, and J. Hurst, "Sim-to-real learning of all common bipedal gaits via periodic reward composition," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2021, pp. 7309–7315.

[5] D. Crowley, J. Dao, H. Duan, K. Green, J. Hurst, and A. Fern, "Optimizing bipedal locomotion for the 100m dash with comparison to human running," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2023, pp. 12205–12211.

[6] R. J. Full and D. E. Koditschek, "Templates and anchors: Neuromechanical hypotheses of legged locomotion on land," *J. Exp. Biol.*, vol. 202, no. 23, pp. 3325–3332, 1999.

[7] M. H. Raibert, H. B. Brown Jr., and M. Chepponis, "Experiments in balance with a 3D one-legged hopping machine," *Int. J. Robot. Res.*, vol. 3, no. 2, pp. 75–92, 1984.

[8] S. Kajita, F. Kanehiro, K. Kaneko, K. Yokoi, and H. Hirukawa, "The 3D linear inverted pendulum mode: A simple modeling for a biped walking pattern generation," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, vol. 1, 2001, pp. 239–246.

[9] H. Geyer, A. Seyfarth, and R. Blickhan, "Compliant leg behaviour explains basic dynamics of walking and running," *Proc. Roy. Soc. B: Biol. Sci.*, vol. 273, no. 1603, pp. 2861–2867, 2006.

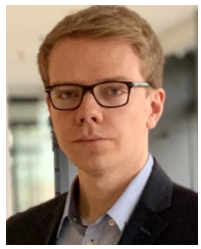
[10] J. Englsberger, C. Ott, and A. Albu-Schäffer, "Three-dimensional bipedal walking control based on divergent component of motion," *IEEE Trans. Robot.*, vol. 31, no. 2, pp. 355–368, Apr. 2015.

[11] R. Blickhan, "The spring-mass model for running and hopping," *J. Biomech.*, vol. 22, no. 11/12, pp. 1217–1227, 1989.

[12] P. M. Wensing and D. E. Orin, "High-speed humanoid running through control with a 3D-SLIP model," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2013, pp. 5134–5140.
[13] G. Secer and U. Saranli, "Control of planar spring–mass running through virtual tuning of radial leg damping," *IEEE Trans. Robot.*, vol. 34, no. 5, pp. 1370–1383, Oct. 2018.
[14] X. Xiong and A. D. Ames, "Bipedal hopping: Reduced-order model embedding via optimization-based control," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2018, pp. 3821–3828.
[15] S. Li, H. Chen, W. Zhang, and P. M. Wensing, "Quadruped robot hopping on two legs," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2021, pp. 7448–7455.
[16] J. Englsberger, P. Kozłowski, C. Ott, and A. Albu-Schäffer, "Biologically inspired deadbeat control for running: From human analysis to humanoid control and back," *IEEE Trans. Robot.*, vol. 32, no. 4, pp. 854–867, Aug. 2016.
[17] T. Egle, J. Englsberger, and C. Ott, "Analytical center of mass trajectory generation for humanoid walking and running with continuous gait transitions," in *Proc. IEEE-RAS Int. Conf. Humanoid Robots*, 2022, pp. 630–637.
[18] M. Vukobratović and J. Stepanenko, "On the stability of anthropomorphic systems," *Math. Biosciences*, vol. 15, no. 1/2, pp. 1–37, 1972.
[19] J. Pratt et al., "Capturability-based analysis and control of legged locomotion, part 2: Application to M2V2, a lower-body humanoid," *Int. J. Robot. Res.*, vol. 31, no. 10, pp. 1117–1133, 2012.
[20] T. Takenaka, T. Matsumoto, and T. Yoshiike, "Real time motion generation and control for biped robot -1st report: Walking gait pattern generation," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2009, pp. 1084–1091.
[21] S. Kajita, T. Nagasaki, K. Kaneko, and H. Hirukawa, "ZMP-based biped running control," *IEEE Robot. Automat. Mag.*, vol. 14, no. 2, pp. 63–72, Jun. 2007.
[22] R. Tajima, D. Honda, and K. Suga, "Fast running experiments involving a humanoid robot," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2009, pp. 1571–1576.
[23] B. Ugurlu, E. Sariyildiz, T. Kawasaki, and T. Narikiyo, "Agile and stable running locomotion control for an untethered and one-legged hopping robot," *Auton. Robots*, vol. 45, pp. 805–819, 2021.
[24] T. Koolen, M. Posa, and R. Tedrake, "Balance control using center of mass height variation: Limitations imposed by unilateral contact," in *Proc. IEEE-RAS Int. Conf. Humanoid Robots*, 2016, pp. 8–15.
[25] D. Ahn and B.-K. Cho, "Online jumping motion generation via model predictive control," *IEEE Trans. Ind. Electron.*, vol. 69, no. 5, pp. 4957–4965, May 2022.
[26] F. M. Smaldone, N. Scianca, L. Lanari, and G. Oriolo, "From walking to running: 3D humanoid gait generation via MPC," *Front. Robot. AI*, vol. 9, 2022, Art. no. 876613.
[27] T. Sugihara, K. Imanishi, T. Yamamoto, and S. Caron, "3D biped locomotion control including seamless transition between walking and running via 3D ZMP manipulation," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2021, pp. 6258–6263.
[28] Z. He and K. Yamamoto, "Humanoid running based on 3D COG-ZMP model and resolved centroidal viscoelasticity control," in *Proc. IEEE-RAS Int. Conf. Humanoid Robots*, 2022, pp. 645–650.
[29] K. Yamamoto, N. Kobayashi, T. Ishigaki, and Y. Sakemi, "Integration of variable-height and hopping strategies for humanoid push recovery," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2022, pp. 12979–12985.
[30] G. Mesesan, J. Englsberger, C. Ott, and A. Albu-Schäffer, "Convex properties of center-of-mass trajectories for locomotion based on divergent component of motion," *IEEE Robot. Automat. Lett.*, vol. 3, no. 4, pp. 3449–3456, Oct. 2018.
[31] G. Mesesan, J. Englsberger, G. Garofalo, C. Ott, and A. Albu-Schäffer, "Dynamic walking on compliant and uneven terrain using DCM and passivity-based whole-body control," in *Proc. IEEE-RAS Int. Conf. Humanoid Robots*, 2019, pp. 25–32.
[32] J. Englsberger, G. Mesesan, C. Ott, and A. Albu-Schäffer, "DCM-based gait generation for walking on moving support surfaces," in *Proc. IEEE-RAS Int. Conf. Humanoid Robots*, 2018, pp. 1–8.
[33] G. Mesesan, J. Englsberger, B. Henze, and C. Ott, "Dynamic multi-contact transitions for humanoid robots using divergent component of motion," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2017, pp. 4108–4115.
[34] M. A. Hopkins, D. W. Hong, and A. Leonessa, "Humanoid locomotion on uneven terrain using the time-varying divergent component of motion," in *Proc. IEEE-RAS Int. Conf. Humanoid Robots*, 2014, pp. 266–272.
[35] S. Hanasaki, Y. Tazaki, H. Nagano, and Y. Yokokohji, "Running trajectory generation including gait transition between walking based on the time-varying linear inverted pendulum mode," in *Proc. IEEE-RAS Int. Conf. Humanoid Robots*, 2022, pp. 851–857.
[36] J. Englsberger et al., "Overview of the torque-controlled humanoid robot TORO," in *Proc. IEEE-RAS Int. Conf. Humanoid Robots*, 2014, pp. 916–923.
[37] M. B. Popovic, A. Goswami, and H. Herr, "Ground reference points in legged locomotion: Definitions, biological trajectories and control implications," *Int. J. Robot. Res.*, vol. 24, no. 12, pp. 1013–1032, 2005.
[38] T. Koolen, T. DeJ. BoerA. R. Goswami, and J. Pratt, "Capturability-based analysis and control of legged locomotion, Part 1: Theory and application to three simple gait models," *Int. J. Robot. Res.*, vol. 31, no. 9, pp. 1094–1113, 2012.
[39] R. Schuller, G. Mesesan, J. Englsberger, J. Lee, and C. Ott, "Online centroidal angular momentum reference generation and motion optimization for humanoid push recovery," *IEEE Robot. Automat. Lett.*, vol. 6, no. 3, pp. 5689–5696, Jul. 2021.
[40] J. Englsberger, G. Mesesan, and C. Ott, "Smooth trajectory generation and push-recovery based on divergent component of motion," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2017, pp. 4560–4567.
[41] B. Henze, M. A. Roa, and C. Ott, "Passivity-based whole-body balancing for torque-controlled humanoid robots in multi-contact scenarios," *Int. J. Robot. Res.*, vol. 35, no. 12, pp. 1522–1543, 2016.
[42] S.-H. Hyon, J. G. Hale, and G. Cheng, "Full-body compliant human–humanoid interaction: Balancing in the presence of unknown external forces," *IEEE Trans. Robot.*, vol. 23, no. 5, pp. 884–898, Oct. 2007.
[43] R. Hinata and D. N. Nenchev, "Balance stabilization with angular momentum damping derived from the reaction null-space," in *Proc. IEEE-RAS Int. Conf. Humanoid Robots*, 2018, pp. 188–195.
[44] F. Kanehiro, H. Hirukawa, and S. Kajita, "OpenHRP: Open architecture humanoid robotics platform," *Int. J. Robot. Res.*, vol. 23, no. 2, pp. 155–165, 2004.
[45] R. Schuller, G. Mesesan, J. Englsberger, J. Lee, and C. Ott, "Online learning of centroidal angular momentum towards enhancing DCM-based locomotion," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2022, pp. 10442–10448.

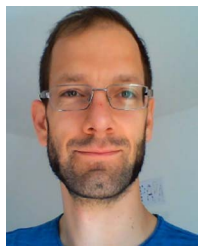
George Mesesan (Member, IEEE) received the Dipl.-Ing. degree in computer science from the Politehnica University of Timişoara, Timişoara, Romania, in 2000, and the M.Sc. degree in robotics from the Technical University of Munich, Munich, Germany, in 2016.

In 2017, he joined the Institute of Robotics and Mechatronics, German Aerospace Center, Cologne, Germany, as a Researcher. His research interests include legged locomotion, path and motion planning, humanoid robots, and whole-body control.



Robert Schuller (Member, IEEE) received the M.Sc. degree in mechanical engineering from the Technical University of Munich, Munich, Germany, in 2020.

In 2020, he joined the Institute of Robotics and Mechatronics of the German Aerospace Center, Cologne, Germany as a Research Scientist. His research interests include humanoid robots, legged locomotion, and whole-body control.



Johannes Englsberger (Senior Member, IEEE) received the diploma in mechanical engineering and the Ph.D. (Dr.-Ing.) degree in robotics from the Technical University of Munich (TUM), Munich, Germany, in 2009 and 2016, respectively.

He is currently holding the position of the Team Leader of Legged Locomotion at the Institute of Robotics and Mechatronics, German Aerospace Center, Cologne, Germany. His research interests lie in the fields of bipedal walking and running, torque-based whole-body control (WBC), robust passivity-based controls, sample-efficient learning for advanced whole-body controls, and teleoperation.

Dr. Englsberger was a recipient of the European Georges Giralt Ph.D. Award for the best Ph.D. thesis in the field of robotics.

Christian Ott (Fellow, IEEE) received the Dipl.-Ing. degree in mechatronics from the University of Linz, Linz, Austria, in 2001, and the Dr.-Ing. degree in control engineering from Saarland University, Saarbrucken, Germany, in 2005.

He is currently a Full Professor with TU Wien, Vienna, Austria. From 2001 to 2007, he worked as a Researcher with the German Aerospace Center (DLR), Weßling, Germany. From 2007 to 2009, he was a Project Assistant Professor with the Department of Mechano-Informatics, University of Tokyo, Tokyo, Japan. After 2009, returned to DLR as Team Leader on Bipedal Walking and, from 2011 to 2016, he led a Helmholtz Young Investigators Group for Dynamic Control of Legged Humanoid Robots. From 2014 to 2022, he was the Head of the Department for Analysis and Control of Advanced Robotic Systems, Institute of Robotics and Mechatronics, DLR. His current research interests include nonlinear robot control, elastic robots, whole-body control, impedance control, and control of humanoid robots.

Dr. Ott was a recipient of an ERC consolidator grant on energy efficient locomotion for elastic robots in 2018. He was the General Chair of Humanoids 2020 in Munich, Germany and he served as an Associate Editor for IEEE TRANSACTIONS ON ROBOTICS and as the Co-Editor-in-Chief for *IFAC Mechatronics*.

Alin Albu-Schäffer (Fellow, IEEE) received the Ph.D. degree in automatic control from the Technical University of Munich (TUM), Munich, Germany, in 2002.

Since 2012, he has been the Head of the Institute of Robotics and Mechatronics, German Aerospace Center (DLR), Weßling, Germany, which he joined in 1995. He is also a Professor with TUM and holds the Chair for "Sensor-Based Robotic Systems and Intelligent Assistance Systems" with the Computer Science Department. He was strongly involved in the development of the DLR lightweight robot and its commercialization through technology transfer to KUKA. His research interests include robot design, modeling and control, nonlinear control, flexible joint and variable compliance robots for manipulation and locomotion, physical human—robot interaction, and bioinspired robot design.

Dr. Albu-Schäffer was a recipient of several awards, including the IEEE King-Sun Fu Best Paper Award of IEEE TRANSACTIONS ON ROBOTICS in 2012 and 2014, the EU-Robotics Technology Transfer Award in 2011, several Best Paper Awards at the major IEEE Robotics Conferences as well as the DLR Science Award. In 2019, he was also a recipient of an ERC Advanced Grant for the Project M-Runner, about energy efficient locomotion based on nonlinear mechanical resonance principles.

IEEE Transactions on Robotics (T-RO) paper, presented at ICRA 2024, Yokohama, Japan. Cite as T-RO paper.