






# Centroidal Angular Momentum-Aware Planning for Legged Locomotion Integrating 3D-DCM and Swing-Leg Trajectory Optimization

Robert Schuller , George Mesesan , Johannes Engelsberger , Jinoh Lee , and Alin Albu-Schäffer 

**Abstract**—In humanoid motion planning, centroidal angular momentum (CAM) is often neglected or simplified due to the complexity arising from its nonlinear and non-holonomic nature. As a result, unmodeled CAM effects can compromise center of pressure (CoP) tracking and, consequently, reduce robustness during motion execution. To address this, we propose an online iterative algorithm that computes the induced CAM and updates the corresponding center of mass (CoM) trajectory accordingly. Since leg motion is the primary source of angular momentum in legged locomotion, the proposed framework also optimizes the swing-leg trajectories to generate the desired CAM while respecting kinematic hardware limits. This results in CAM-consistent whole-body motions that enable stable yet faster locomotion with reduced demand for maximum joint velocities. We demonstrate the effectiveness of our approach through experiments and simulations with the humanoid robot TORO in several locomotion scenarios, including balancing, walking, and running.

**Index Terms**—Whole-body motion planning and control, humanoid and bipedal locomotion

## I. INTRODUCTION

**B**IPEDAL locomotion is challenging due to the inherently unstable, nonlinear dynamics and underactuation of the systems involved. Consequently, many motion generation approaches focus on the center of mass (CoM) dynamics and employ simplified models, such as the linear inverted pendulum (LIP) [1]. Based on these models, numerous gait planning strategies have been developed, among which the zero moment point (ZMP) method [2] is particularly notable.

Received 15 July 2025; accepted 3 January 2026. Date of publication 22 January 2026;. This article was recommended for publication by Associate Editor A. Hereid and Editor A. Kheddar upon evaluation of the reviewers' comments. This work was supported in part by ITECH R&D programs of MOTIE/KEIT under Grant 20026194. (Corresponding author: Robert Schuller.)

Robert Schuller, George Mesesan, and Alin Albu-Schäffer are with the Institute of Robotics and Mechatronics, German Aerospace Center (DLR), 82234 Wessling, Germany, and also with the School of Computation, Information and Technology, Technical University of Munich (TUM), 85748 Garching, Germany (e-mail: robert.schuller@dlr.de; george.mesesan@dlr.de; alin.albu-schaeffer@dlr.de).

Johannes Engelsberger is with the Institute of Robotics and Mechatronics, German Aerospace Center (DLR), 82234 Wessling, Germany. (e-mail: johannes.engelsberger@dlr.de).

Jinoh Lee is with the Institute of Robotics and Mechatronics, German Aerospace Center (DLR), 82234 Wessling, Germany, and also with the Department of Mechanical Engineering, KAIST, Daejeon 34141, South Korea (e-mail: jinoh.lee@dlr.de).

This article has supplementary downloadable material available at <https://doi.org/10.1109/LRA.2026.3656798>, provided by the authors.

Digital Object Identifier 10.1109/LRA.2026.3656798

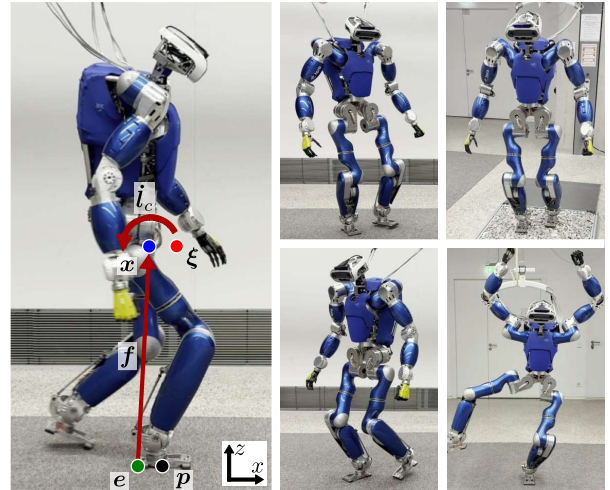


Fig. 1. Snapshots of the humanoid robot TORO performing dynamic locomotion tasks, including walking at 0.51 m/s, turning, walking on challenging terrain, curved walking, and dynamic balancing. The following quantities are shown: CoP ( $p$ ), eCMP ( $e$ ), DCM ( $\xi$ ), CoM ( $x$ ), rate of change of CAM ( $\dot{l}_c$ ), and external force vector ( $f$ ).

In the context of running, the approaches proposed in [3], [4] are well-established.

In humanoid motion planning, the influence of the centroidal angular momentum (CAM) is often neglected. This is mainly due to the complexity introduced by its nonlinear and non-holonomic nature. As a result, CAM is typically regarded as a byproduct of the whole-body trajectory rather than being explicitly planned. However, biomechanical studies have emphasized the significance of angular momentum in both walking [5] and running [6].

If not explicitly considered during planning, the unmodeled effects of the angular momentum can lead to several drawbacks. As pointed out in [7], the CAM resulting from the multi-body dynamics can lead to a substantial deflection of the center of pressure (CoP) during walking and running. The CoP refers to the point within the support area where the resultant ground reaction force vector acts [2]. If this point reaches the edge of the support area, the feet may tilt, potentially causing the robot to fall.

Despite the relevance of CAM, its integration into online motion planning remains an open research challenge. Some approaches try to extend the LIP model to more accurately approximate the CAM [8], [9]. More recently, in [10], the authors reformulated the LIP model in terms of angular momentum. In [11], the CAM is approximated using a simplified three-mass model, which is incorporated into the CoM planning.

However, these approximations do not fully capture the entire complexity of the humanoid’s CAM [7]. Other methods focus on the optimization of the leg trajectories to compensate the CAM during walking [12] or to ensure a reduced base motion during running [13].

A further branch of research employs trajectory optimization techniques to account for the CAM during motion generation. In [14], the authors incorporate the full robot dynamics to optimize trajectories that satisfy both contact and dynamic constraints. However, these approaches often entail high computational costs and are prone to local minima. Alternative solutions only use the centroidal dynamics combined with full kinematics to capture the essential dynamic behavior while reducing complexity, without modeling the entire robot’s dynamics [15], [16].

More recently, reinforcement learning-based methods have demonstrated the ability to generate robust walking and running motions [17], [18]. Although these methods do not explicitly model angular momentum as a physical quantity, extensive training enables them to learn complex behaviors; however, the resulting end-to-end learned policies lack interpretability.

The aforementioned approaches address different aspects of CAM in motion planning, depending on the specific scenario and locomotion mode, e.g., walking or running. In contrast, we propose a holistic framework that enables dynamic locomotion by generating CAM-consistent whole-body trajectories for generic locomotion scenarios, including balancing, walking, and running, while accounting for the full centroidal dynamics without simplifications. Since leg motion is the primary source of angular momentum in humanoid locomotion, we jointly optimize leg trajectories together with the resulting CAM and CoM trajectory. The proposed method enhances robustness through explicit CoP tracking and produces kinematically feasible swing leg trajectories, which are especially critical for highly dynamic locomotion operating close to hardware limits.

We build our method on the 3D divergent component of motion (3D-DCM) framework [19], which enables efficient closed-form computation of CoM trajectories [20]. The framework has been successfully applied to various scenarios, including stair climbing [21], running, and jumping [22]. The integration of CAM into the CoM trajectory generation can be naturally achieved through adjusting the enhanced centroidal moment pivot (eCMP), a point that encodes the direction and magnitude of external forces acting on the CoM (more details in Section II). The deviation of the eCMP from the CoP directly corresponds to the induced rate of change of the CAM. Notably, our proposed approach introduces a paradigm shift within the 3D-DCM framework: instead of specifying a desired eCMP trajectory as input, we can directly define a desired CoP trajectory for generating the CoM trajectory.

The main contributions of this work are: (1) an online iterative algorithm for computing eCMP offsets based on the induced rate of change of CAM, enabling CoP tracking across generic locomotion scenarios; and (2) a quadratic optimization method integrated into (1) that generates swing leg trajectories satisfying kinematic constraints while achieving a desired CAM objective.

## II. FUNDAMENTALS

This section derives the dynamic model of the robot and presents the fundamentals of the 3D-DCM framework [19].

### A. Modeling

For a humanoid robot with  $n$  torque-controlled joints and corresponding joint velocities  $\dot{\mathbf{q}} \in \mathbb{R}^n$ , we denote  $\dot{\mathbf{x}} \in \mathbb{R}^3$  and  $\boldsymbol{\omega}_b \in \mathbb{R}^3$  as the CoM velocity and angular base velocity. All velocities are stacked into the generalized velocity vector  $\dot{\mathbf{q}}_c^T = (\dot{\mathbf{x}}^T, \boldsymbol{\omega}_b^T, \dot{\mathbf{q}}^T)^T$ . The total number of degrees of freedom (DoF) of the system is  $\bar{n} = n + 6$ .

For locomotion purposes, we introduce an operational task space including the 6-DoF task for tracking the CoM position and base orientation, and the end effector velocities of the stance leg  $\dot{\mathbf{x}}_{st} \in \mathbb{R}^6$  and the swing leg  $\dot{\mathbf{x}}_{sw} \in \mathbb{R}^6$  in Cartesian space. The remaining  $n_f$  DoF, mainly the joints in the upper body, are defined in joint space  $\dot{\mathbf{q}}_f \in \mathbb{R}^{n_f}$ , yielding the final task space velocity vector  $\dot{\mathbf{x}}_{task}^T = (\dot{\mathbf{x}}^T, \boldsymbol{\omega}_b^T, \dot{\mathbf{x}}_{st}^T, \dot{\mathbf{x}}_{sw}^T, \dot{\mathbf{q}}_f^T)^T$ . The generalized task space velocities are described by the Jacobian  $\mathbf{J} \in \mathbb{R}^{(18+n_f) \times \bar{n}}$  as

$$\dot{\mathbf{x}}_{task} = \mathbf{J}\dot{\mathbf{q}}_c. \quad (1)$$

In addition, we denote the CAM as  $\mathbf{l}_c \in \mathbb{R}^3$ , which is obtained by

$$\mathbf{l}_c = \mathbf{A}_l \dot{\mathbf{q}}_c = \mathbf{A}_l \mathbf{J}^\# \dot{\mathbf{x}}_{task} = \bar{\mathbf{A}}_l \dot{\mathbf{x}}_{task}, \quad (2)$$

where  $\mathbf{A}_l \in \mathbb{R}^{3 \times \bar{n}}$  denotes the rotational part of the centroidal momentum matrix (CMM) [23],  $\mathbf{J}^\# \in \mathbb{R}^{\bar{n} \times (18+n_f)}$  is a generalized damped pseudoinverse of  $\mathbf{J}$ , and  $\bar{\mathbf{A}}_l := \mathbf{A}_l \mathbf{J}^\#$  is defined as the task space-transformed CMM.

### B. 3D-Divergent Component of Motion (3D-DCM)

The core idea of the 3D-DCM is to reformulate the second-order CoM dynamics into a stable and an unstable component [19]. The state variable associated with the unstable part is the DCM,  $\boldsymbol{\xi} \in \mathbb{R}^3$ , defined as a linear combination of the CoM position and velocity, i.e.,  $\boldsymbol{\xi} := \mathbf{x} + b\dot{\mathbf{x}}$ . Here,  $b = \sqrt{\Delta z/g}$  is the DCM time constant,  $\Delta z > 0$  is a free motion parameter, and  $g$  is the gravitational constant. By introducing the virtual repellent point (VRP),  $\mathbf{v} \in \mathbb{R}^3$ , which encodes the direction and magnitude of the total force applied at the CoM, the unstable first-order dynamics of the DCM is formulated as

$$\dot{\boldsymbol{\xi}} = \frac{1}{b}(\boldsymbol{\xi} - \mathbf{v}). \quad (3)$$

Finally, the stable first-order CoM dynamics results from reordering the DCM definition as

$$\dot{\mathbf{x}} = -\frac{1}{b}(\mathbf{x} - \boldsymbol{\xi}). \quad (4)$$

The eCMP,  $\mathbf{e} \in \mathbb{R}^3$ , is a further point in the 3D-DCM framework, as shown in Fig. 1. It encodes the external forces acting on the CoM and therefore naturally serves as a planning entry point since it relates to the force  $\mathbf{f} \in \mathbb{R}^3$  that the robot generates in interaction with the environment, given as

$$\mathbf{f} = \frac{m}{b^2}(\mathbf{x} - \mathbf{e}). \quad (5)$$

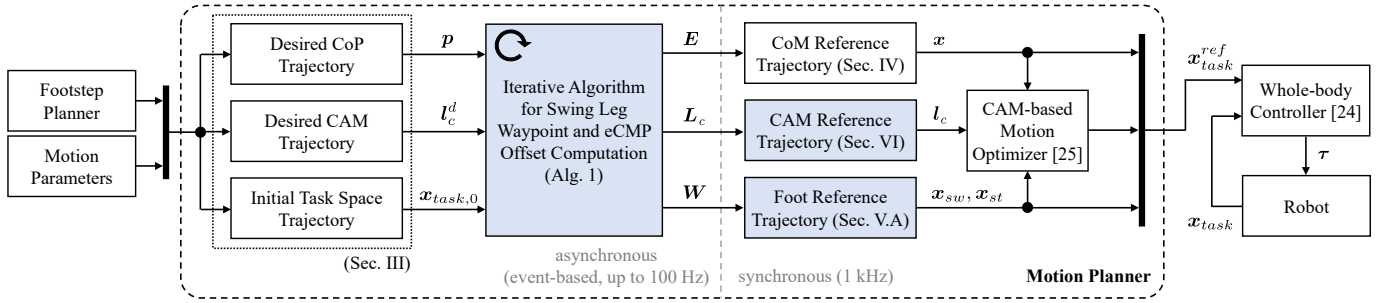


Fig. 2. Overview of the proposed planning and control architecture. For clarity and brevity, not all system states are displayed, while the contribution of this work is highlighted in blue.

In contrast to the eCMP, the VRP also accounts for gravitational forces. As a result, the two points are separated by a vertical offset, given by  $\mathbf{v} = \mathbf{e} + (0, 0, \Delta z)^T$ .

The CoP, denoted by  $\mathbf{p} \in \mathbb{R}^3$ , is constrained to lie within the contact area as follows [2]:

$$\mathbf{p}_{xy} = \frac{1}{f_z}(-\tau_y, \tau_x)^T, \quad (6)$$

where  $\boldsymbol{\tau} \in \mathbb{R}^3$  is the contact torque, resulting from the CoM force and rate of change of CAM, i.e.,  $\boldsymbol{\tau} = \mathbf{x} \times \mathbf{f} + \dot{\mathbf{l}}_c$ . Note that all quantities are expressed in the contact frame, whose origin is located within the contact area. Combined with (5) and (6), the CoP can be related to the eCMP through the following expression [7]:

$$\mathbf{e}_{xy} = \mathbf{p}_{xy} + \underbrace{\frac{\mathbf{e}_z}{\mathbf{X}_z}(\mathbf{x}_{xy} - \mathbf{p}_{xy}) + \frac{b^2}{m\mathbf{X}_z}(\dot{\mathbf{l}}_{c,y}, -\dot{\mathbf{l}}_{c,x})^T}_{\Delta \mathbf{e}_{xy}}. \quad (7)$$

### III. OVERVIEW

An overview of the proposed planning and control architecture is presented in Fig. 2. A footstep planner generates a sequence of desired footsteps and motion parameters, such as step timings. Based on these inputs, a desired CoP trajectory  $\mathbf{p}(t) \in \mathbb{R}^3$  is specified. Typically, the CoP transitions from one footstep to the next. In addition, we design a desired CAM  $\mathbf{l}_c^d(t) \in \mathbb{R}^3$  trajectory. For walking and running, as also observed in biomechanical studies [5], the CAM is typically regulated to zero, while for highly-dynamic tasks such as a backflip, a non-constant CAM is required.

Finally, an initial task-space trajectory,  $\mathbf{x}_{\text{task},0}(t) \in \mathbb{R}^{\bar{n}}$ , is generated, including the initial swing-foot trajectories. These are constructed using piecewise fifth-order polynomials for each Cartesian coordinate, except for the vertical motion, which is modeled by two fifth-order polynomials. Similarly, the base reference trajectory is generated using piecewise fifth-order polynomials. Additionally, the free joints are commanded to maintain a constant posture [24].

An iterative algorithm takes these quantities as inputs and is executed asynchronously to optimize the swing leg waypoints and compute eCMP offsets. The resulting eCMP waypoint sequence is then used to compute the reference CoM trajectory in real-time, and the swing and stance leg trajectories are derived from the optimized waypoints. Subsequently, the base and the upper body reference trajectories are adjusted by the motion optimizer to realize the reference CAM [25].

The resulting whole-body reference trajectory is tracked by a passivity-based whole-body controller [24], which generates corresponding joint torque commands.

### IV. eCMP WAYPOINT PLACEMENT

As presented in [22], the DCM and CoM trajectories can be obtained in matrix form by piecewise interpolation over a sequence of waypoints, using the analytical solutions of (3) and (4) as interpolation functions. All  $n_w$  eCMP waypoints are collected in the matrix  $\mathbf{E} = [\mathbf{e}_1, \dots, \mathbf{e}_{n_w}]^T \in \mathbb{R}^{n_w \times 3}$ , the corresponding DCM waypoints in  $\boldsymbol{\Xi} \in \mathbb{R}^{n_w \times 3}$ , and the CoM waypoints in  $\mathbf{X} \in \mathbb{R}^{n_w \times 3}$ . The timings associated with the waypoints are collected in  $\mathbf{t} = (0, t_2, \dots, t_{n_w})^T \in \mathbb{R}^{n_w}$  with the properties  $0 < t_2 < \dots < t_{n_w}$ , where  $t_{n_w}$  represents the total time of the motion sequence.

First, we sample CoP waypoints  $\mathbf{P} \in \mathbb{R}^{n_w \times 3}$  from the desired CoP trajectory given as

$$\mathbf{P}(i) = \mathbf{p}(t)^T, \quad \text{for } i = 1 \dots n_w \text{ and } t = \mathbf{t}(i), \quad (8)$$

where  $\mathbf{P}(i)$  refers the  $i$ -th waypoint in the waypoint matrix  $\mathbf{P}$  and  $\mathbf{t}(i)$  selects the  $i$ -th time instance in the waypoint timings vector  $\mathbf{t}$  (see Algorithm 1, line 3).

Next, we derive the eCMP waypoints. As shown in (7), the horizontal eCMP and CoP positions are separated by an offset that depends on the induced rate of change of CAM. We reformulate (7) using waypoint notation as follows:

$$\mathbf{E}_{xy} = \mathbf{P}_{xy} + \Delta \mathbf{E}_{xy}, \quad \mathbf{E}_z = \mathbf{P}_z, \quad (9)$$

where the horizontal eCMP offsets  $\Delta \mathbf{E}_{xy} \in \mathbb{R}^{n_w \times 2}$  are computed element-wise for  $i = 1 \dots n_w$  as

$$\begin{aligned} \Delta \mathbf{E}_{xy}(i)^T &= \frac{\mathbf{E}_z(i)}{\mathbf{X}_z(i)} (\mathbf{X}_{xy}(i)^T - \mathbf{P}_{xy}(i)^T) \\ &\quad + \frac{b^2}{m\mathbf{X}_z(i)} \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix} \dot{\mathbf{l}}_{c,xy}(i)^T. \end{aligned} \quad (10)$$

Here,  $\mathbf{X}_z$  and  $\mathbf{E}_z$  represent the vertical components of the CoM and eCMP waypoints, respectively. Since they depend only on the vertical elements, they can be computed directly. The horizontal CoM position  $\mathbf{X}_{xy}$  and the induced horizontal rate of change of angular momentum  $\dot{\mathbf{l}}_{c,xy} \in \mathbb{R}^{n_w \times 2}$  are computed iteratively at the waypoint timings. In the initial iteration, we assume  $\Delta \mathbf{E}_{xy} = \mathbf{0}_{n_w \times 2}$  (see line 7). Note that the first term in (10) is zero when the eCMP lies in the plane of the contact, i.e.,  $\mathbf{E}_z(i) = 0$ .

The DCM and CoM waypoint matrices are computed from the eCMP waypoints using the following expressions, respectively:

$$\Xi = C_{\Xi}E + c_{\Xi}, \quad X = C_X E + c_X. \quad (11)$$

The matrices  $C_{\Xi}, C_X, c_{\Xi}, c_X$  can be computed in closed form, and depend on the waypoint timings  $t$ , the DCM constant  $b$ , the final DCM position, the initial CoM position, and, in the case of running, the gravity vector. Detailed formulations of these matrices can be found in [20], [22].

Based on (3), (4), and their time derivatives, we derive expressions for the CoM velocity and acceleration at the waypoints as follows:

$$\dot{X} = \frac{1}{b}(\Xi - X), \quad \ddot{X} = \frac{1}{b^2}(X - E - \Delta Z), \quad (12)$$

with  $\Delta Z = \mathbf{1}_{n_w}(0, 0, \Delta z) \in \mathbb{R}^{n_w \times 3}$  (see line 10).

## V. SWING LEG TRAJECTORY OPTIMIZATION

As the next step, we optimize the swing leg trajectory with two objectives: to ensure that it satisfies the robot's kinematic constraints and to realize a desired CAM trajectory.

### A. Swing Leg Trajectory Formulation

We define the swing leg trajectory in Cartesian space as a function of time and optimized waypoints. We start by establishing  $n_{sw} > 2$  waypoints, which are connected by 5th-order polynomials to ensure  $C^2$  continuity. The time intervals between the waypoints  $T_i, \forall i \in \{1 \dots (n_{sw} - 1)\}$ , with total swing time  $T_{sw} = \sum_{i=1}^{n_{sw}-1} T_i$ , are fixed to maintain a quadratic optimization problem. Each waypoint contains position, velocity, and acceleration values for each spatial dimension (translational and rotational), i.e.,  $W_i \in \mathbb{R}^{3 \times 6}, \forall i \in \{1 \dots n_{sw}\}$ . Here,

$$W = [W_1^T, W_2^T, \dots, W_{n_{sw}-1}^T, W_{n_{sw}}^T]^T \in \mathbb{R}^{3n_{sw} \times 6} \quad (13)$$

contains the  $n_{sw}$  waypoints. The swing leg trajectory for each spatial dimension is computed as follows:

$$\begin{aligned} x_{sw}(t, W) &= P_c(t)W, \quad \dot{x}_{sw}(t, W) = V_c(t)W, \\ \ddot{x}_{sw}(t, W) &= A_c(t)W \quad \text{with } t \in [0, T_{sw}]. \end{aligned} \quad (14)$$

The matrices  $P_c(t), V_c(t), A_c(t) \in \mathbb{R}^{1 \times 3n_{sw}}$  are computed based on the polynomial coefficients of the 5th-order polynomials connecting the waypoints, as well as the constraints resulting from  $C^2$  continuity at the waypoints themselves. Further details on the computation of the polynomial coefficient matrices can be found in [7], [26].

### B. Waypoint-Based Inverse Kinematic Computation

To evaluate the CAM and joint velocities induced by the swing leg trajectory, the corresponding centroidal momentum matrices and Jacobians are required. These are obtained by computing the inverse kinematics (IK) at discrete waypoints along the swing phase, based on the known task-space trajectory. We solve the IK problem at the eCMP waypoint timings  $t$ , enabling the use of the resulting CMMs to also compute the eCMP offsets, as we will discuss in Section VI.

### Algorithm 1 Iterative Algorithm for Swing Leg Waypoint and eCMP Offset Computation

---

**Input:**  $t, p(t), l_c^d(t), \dot{l}_c^d(t), x_{task,0}(t), \dot{x}_{task,0}(t), \ddot{x}_{task,0}(t)$   
**Output:**  $E, W, L_c, \dot{L}_c$

- 1: **for**  $i \leftarrow 1$  to  $n_w$  **do**
- 2:      $t \leftarrow t(i)$
- 3:      $P(i) \leftarrow p(t)$
- 4:      $X_{task}(i) \leftarrow x_{task,0}(t), \dot{X}_{task}(i) \leftarrow \dot{x}_{task,0}(t), \ddot{X}_{task}(i) \leftarrow \ddot{x}_{task,0}(t)$
- 5:      $L_c^d(i) \leftarrow l_c^d(t), \dot{L}_c^d(i) \leftarrow \dot{l}_c^d(t)$
- 6: **end for**
- 7:  $\Delta E_{xy} \leftarrow \mathbf{0}_{n_w \times 2}, E \leftarrow P$
- 8: **repeat**
- 9:      $\Delta E_{prev} \leftarrow \Delta E$
- 10:      $(X, \dot{X}, \ddot{X}) \leftarrow (11) \text{ and } (12)$
- 11:      $(X_{task}, \dot{X}_{task}, \ddot{X}_{task}) \leftarrow \text{update}(X, \dot{X}, \ddot{X})$
- 12:     **for**  $i \leftarrow 1$  to  $n_w$  **do**
- 13:          $(\bar{A}_l(i), \dot{\bar{A}}_l(i), J(i), \dot{J}(i)) = \text{computeIKwithCMM}(X_{task}(i), \dot{X}_{task}(i))$
- 14:     **end for**
- 15:      $W \leftarrow (21)$       $\triangleright$  Swing Leg Optimization
- 16:      $(X_{sw}, \dot{X}_{sw}, \ddot{X}_{sw}) \leftarrow (18)$
- 17:      $(X_{task}, \dot{X}_{task}, \ddot{X}_{task}) \leftarrow \text{update}(X_{sw}, \dot{X}_{sw}, \ddot{X}_{sw})$
- 18:      $\dot{L}_c \leftarrow (20)$
- 19:      $\Delta E \leftarrow (10)$       $\triangleright$  eCMP Offset Computation
- 20:      $E \leftarrow (9)$
- 21: **until**  $\|\Delta E - \Delta E_{prev}\|_2 < \epsilon$
- 22:  $L_c \leftarrow (19), L_{c,z} \leftarrow \mathbf{0}_{n_w}, \dot{L}_{c,z} \leftarrow \mathbf{0}_{n_w}$

---

First, we evaluate the initial task space trajectory and swing leg trajectory at the corresponding timings (see Algorithm 1, line 4),

$$\begin{aligned} X_{task}(i) &= x_{task,0}(t), \quad \dot{X}_{task}(i) = \dot{x}_{task,0}(t), \\ \ddot{X}_{task}(i) &= \ddot{x}_{task,0}(t), \quad \text{for } i = 1 \dots n_w \text{ and } t = t(i). \end{aligned} \quad (15)$$

Afterwards, we update the task space vector with the computed CoM trajectory from (11) and (12)

$$(X_{task}, \dot{X}_{task}, \ddot{X}_{task}) \leftarrow \text{update}(X, \dot{X}, \ddot{X}), \quad (16)$$

where the function “update” replaces the CoM waypoints with the newly computed ones (see line 11).

Based on the task space vector, we evaluate the inverse kinematics at the waypoints for  $i = 1 \dots n_w$

$$\begin{aligned} (\bar{A}_l(i), \dot{\bar{A}}_l(i), J(i), \dot{J}(i)) & \\ &= \text{computeIKwithCMM}(X_{task}(i), \dot{X}_{task}(i)). \end{aligned} \quad (17)$$

The function “computeIKwithCMM” numerically computes the joint-space configuration from the given task-space vector and outputs the CMM transformed into task space, the Jacobian, and their time derivatives from the final forward kinematics step (see lines 12-14). This computation can be performed using any rigid-body dynamics algorithm.

In the next step, we also evaluate the swing leg trajectory in (14) at the  $n_w$  eCMP waypoint timings

$$\begin{aligned} X_{sw}(i) &= P_c(t)W, \quad \dot{X}_{sw}(i) = V_c(t)W, \\ \ddot{X}_{sw}(i) &= A_c(t)W, \quad \text{for } i = 1 \dots n_w \text{ and } t = t(i). \end{aligned} \quad (18)$$

As the last component before formulating the optimization, we define the CAM as a function of the swing leg trajectory. We introduce  $\mathbf{L}_c \in \mathbb{R}^{n_w \times 3}$  and  $\dot{\mathbf{L}}_c \in \mathbb{R}^{n_w \times 3}$  as matrices containing the CAM and its rate of change, respectively, sampled at the waypoint timings  $\mathbf{t}$ . Based on (2), we obtain

$$\mathbf{L}_c(i) = \bar{\mathbf{A}}_l(i) \dot{\mathbf{X}}_{task}(i) = \bar{\mathbf{A}}_{l,sw}(i) \dot{\mathbf{X}}_{sw}(i) + \mathbf{L}_{c,\chi}(i), \quad (19)$$

where we split the task space vector and the CMM into the swing leg task and the remaining tasks of the task vector by using a selection matrix  $\mathbf{S} \in \mathbb{R}^{\bar{n} \times \bar{n}}$  containing ones and zeros, such that  $\bar{\mathbf{A}}_l \mathbf{S}^T = [\bar{\mathbf{A}}_{l,sw}, \bar{\mathbf{A}}_{l,\chi}]$ ,  $\mathbf{S} \dot{\mathbf{X}}_{task} = [\dot{\mathbf{X}}_{sw}^T, \dot{\mathbf{X}}_{\chi}^T]^T$ , and  $\mathbf{S}^T \mathbf{S} = \mathbf{I}_{\bar{n} \times \bar{n}}$ . The known CAM contributions from all tasks, except the swing leg tasks, are summarized in  $\mathbf{L}_{c,\chi}(i) = \bar{\mathbf{A}}_{l,\chi}(i) \dot{\mathbf{X}}_{\chi}(i)$ . The rate of change of CAM is obtained as the time derivative of (19):

$$\dot{\mathbf{L}}_c(i) = \bar{\mathbf{A}}_{l,sw}(i) \ddot{\mathbf{X}}_{sw}(i) + \dot{\bar{\mathbf{A}}}_{l,sw}(i) \dot{\mathbf{X}}_{sw}(i) + \dot{\mathbf{L}}_{c,\chi}(i). \quad (20)$$

### C. Swing Leg Waypoint Optimization

We solve a quadratic program (QP) in which the swing-leg waypoints  $\mathbf{W}$  serve as optimization variables. The time horizon of the optimization is chosen as the duration of a single swing phase,  $T_{sw}$ , from the contact detachment to the contact reattachment of the same leg. One objective of the QP is to realize a desired CAM trajectory. Accordingly, the trajectory  $\mathbf{l}_c^d(t)$  presented in Section III is sampled at the waypoint timings  $\mathbf{t}$  to yield the matrices  $\mathbf{L}_c^d \in \mathbb{R}^{n_w \times 3}$  and  $\dot{\mathbf{L}}_c^d \in \mathbb{R}^{n_w \times 3}$  (see line 5).

The following cost function for the swing leg waypoint optimization is introduced:

$$\min_{\mathbf{W}} \sum_{i=1}^{n_w} ( \|\mathbf{L}_c(i) - \mathbf{L}_c^d(i)\|_{Q_1} + \|\dot{\mathbf{L}}_c(i) - \dot{\mathbf{L}}_c^d(i)\|_{Q_2} + \|\dot{\mathbf{X}}_{sw}(i)\|_{Q_3} + \|\ddot{\mathbf{X}}_{sw}(i)\|_{Q_4} ), \quad (21)$$

where  $\mathbf{L}_c$  and  $\dot{\mathbf{L}}_c$  are linear functions of  $\mathbf{W}$  as derived in (18)-(20). The first two tasks in (21) aim to realize  $\mathbf{L}_c^d$  and  $\dot{\mathbf{L}}_c^d$ , while the second and third tasks regulate the swing leg velocity and acceleration, with individual task weightings  $Q_j > 0$ ,  $j = 1 \dots 4$ . The first and last waypoints in  $\mathbf{W}$  are fixed as they correspond to the desired start and end poses of the swing leg. The optimization is solved subject to the following constraints:

1) *Cartesian Space Constraints*: The Cartesian limits are particularly important for the vertical translation to ensure a minimum clearance during the swing phase:

$$\mathbf{X}_{sw}^{min}(i) < \mathbf{X}_{sw}(i) < \mathbf{X}_{sw}^{max}(i). \quad (22)$$

Here,  $\mathbf{X}_{sw}^{min}(i)$  and  $\mathbf{X}_{sw}^{max}(i) \in \mathbb{R}^{1 \times 6}$  denote the lower and upper limits, respectively.

2) *Joint Space Constraints*: The joint velocities and accelerations are obtained from the pseudoinverse of the Jacobian and the task space vector in (1) and its time derivative:

$$\dot{\mathbf{q}}^{min} < \mathbf{J}_{sw}^\#(i) \dot{\mathbf{X}}_{sw}(i) + \mathbf{J}_{\chi}^\#(i) \dot{\mathbf{X}}_{\chi}(i) < \dot{\mathbf{q}}^{max}. \quad (23)$$

The pseudoinverse of the Jacobian is partitioned as  $\mathbf{S}_q \mathbf{J}^\# \mathbf{S}^T = [\mathbf{J}_{sw}^\#, \mathbf{J}_{\chi}^\#]$  where  $\mathbf{S}_q \in \mathbb{R}^{n_q \times \bar{n}}$  is a joint selection matrix used to enforce joint-space constraints only on the selected  $n_q$  joints. The joint velocity lower and upper limits

are represented by  $\dot{\mathbf{q}}^{min}$ , and  $\dot{\mathbf{q}}^{max} \in \mathbb{R}^{n_q}$ , respectively. The joint acceleration constraints are realized corresponding to the time derivative of (23).

Once the QP is solved, we can compute the new swing leg trajectory  $\mathbf{X}_{sw}$ ,  $\dot{\mathbf{X}}_{sw}$ ,  $\ddot{\mathbf{X}}_{sw}$  based on (18) and  $\mathbf{W}$ , and update the task space vector with the new waypoints for the swing leg, accordingly (see lines 15 to 17).

## VI. ITERATIVE ALGORITHM

The proposed Algorithm 1 takes as input the waypoint timings, the desired CoP and CAM trajectory, and the initial task-space reference trajectory and outputs CAM-adjusted eCMP waypoints, optimized swing leg waypoints, and a CAM reference.

Once the task-space trajectory is updated, the rate of change of the CAM is computed (line 18), from which the eCMP offsets are derived (line 19). The eCMP waypoints are then obtained accordingly (line 20).

After executing lines 8-21, we obtain a task-space trajectory with updated CoM and swing-leg waypoints. However, the CMMs and Jacobians required to compute the swing-leg waypoints and eCMP offsets are derived from the previous iteration's task-space trajectory. Each update of the CoM and swing-leg waypoints alters the robot configuration, leading to corresponding changes in the CMMs and Jacobians. Therefore, we repeat lines 8-21 until the Euclidean norm of the difference between successive eCMP offsets becomes smaller than a predefined threshold  $\epsilon > 0$ . Based on (9), this condition corresponds to the convergence of the CoP to its desired value.

As the final step, the reference CAM  $\mathbf{L}_c$  and its rate of change  $\dot{\mathbf{L}}_c$  are computed (line 22). Note that tracking the desired CAM is only a weighted task in (21); therefore, the resulting reference CAM may deviate from the desired one. Following biomechanical insights from [5], the vertical reference CAM and its rate of change are set to zero. The final trajectory is obtained by 3rd-order interpolation between the waypoints in  $\mathbf{L}_c$ , using  $\dot{\mathbf{L}}_c$  waypoints to ensure  $C^1$  continuity. As illustrated in Fig. 2, the motion optimizer computes the base and upper-body motions necessary to track the reference CAM, resulting in arm-swinging motions during both walking and running.

Note that, depending on the locomotion scenario, either the swing leg optimization or the eCMP offset computation can be independently deactivated by disabling lines 15-17 or lines 18-20, respectively. For example, during dynamic balancing motions, the swing leg optimization is not required.

Fig. 3 illustrates the eCMP offsets in the  $x$ -direction. In this example, Algorithm 1 is executed three times. Iteration #0 represents the initial trajectory described in Section III, with all eCMP offsets set to zero, corresponding to a constant CAM assumption. The eCMP trajectory is generated by linearly interpolating between the respective waypoints. It can be observed that the eCMP offsets are updated in each iteration based on the computed rate of change of the CAM. The magnitude of the eCMP offset updates diminishes with each successive iteration.

Fig. 4 presents the resulting CoP error for each iteration. Note that the CoP error is obtained from a multi-body dy-

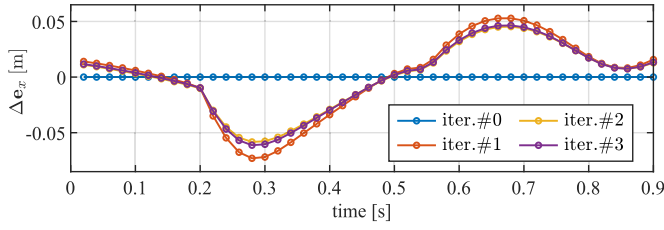


Fig. 3. eCMP offsets in  $x$ -direction for a double and single support phase for straight flat foot walking with a total swing time of  $T_{sw} = 0.7$  s. The time interval between the waypoints in  $\mathbf{E}$  is 20 ms, i.e.,  $\mathbf{t} = (0, 0.02, 0.04, \dots, 0.9)$  s, resulting in a total number of  $n_w = 45$  waypoints (indicated by the filled circles).

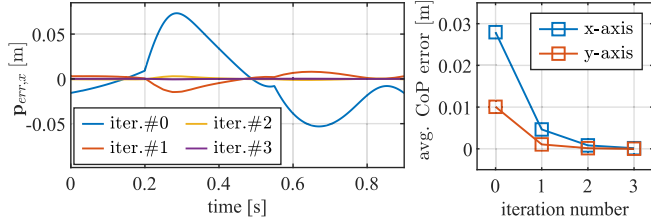


Fig. 4. The left-hand plot presents the CoP error corresponding to Fig. 3 in  $x$ -direction, defined as  $\mathbf{p}_{err}(t) = \mathbf{p}_{xy}^{mea}(t) - \mathbf{p}_{xy}(t)$ . Here,  $\mathbf{p}_{xy}(t)$  is the horizontal component of the desired CoP trajectory, and  $\mathbf{p}_{xy}^{mea}(t)$  is the measured CoP according to (7). The right-hand plot presents the average CoP error for each iteration computed as  $err_i = \frac{1}{T_{sw}} \int_0^{T_{sw}} |\mathbf{p}_{err}(t)| dt$ .

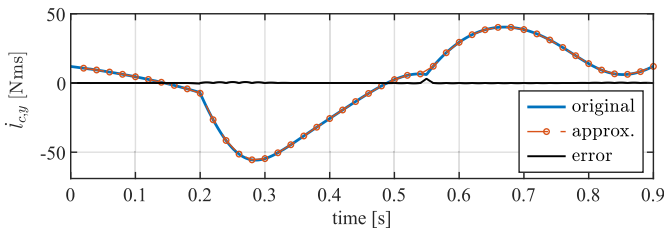


Fig. 5. Rate of change of CAM about the  $y$ -axis computed from the multi-body dynamics (blue). The waypoints in  $\dot{\mathbf{L}}_{c,y}$  are indicated by filled orange circles. The trajectory obtained by linearly interpolating between these waypoints is shown in orange. The difference between the original and interpolated trajectories is plotted as the error in black.

namics simulation, used solely for validation and not required by the algorithm itself. Additionally, the average CoP error across iterations is presented. It can be seen that the CoP error converges towards zero as the number of iterations increases.

Fig. 5 illustrates, as an example for iteration #0, the rate of change of CAM about the  $y$ -axis computed from the multi-body dynamics. The corresponding sampled values of  $\dot{\mathbf{L}}_{c,y}$  at the waypoint timings are also shown. Additionally, the error defined as the difference between the original trajectory and its approximation obtained by linearly interpolating between the waypoints in  $\dot{\mathbf{L}}_{c,y}$  is presented. It can be observed that the approximation closely reproduces the original trajectory, owing to the sufficient density of waypoints. This justifies the dense sampling of the original CoP trajectory, which ensures an accurate approximation of the rate of change of CAM.

We execute the iterative algorithm online in an event-based manner within the asynchronous module, each time new footsteps are added to the motion plan or an existing plan is modified. Typically, during walking, a motion plan consisting of four forward steps is transmitted to the synchronous module, which subsequently evaluates the received waypoints and generates the corresponding time trajectory.

The computational cost per iteration primarily arises from computing the inverse kinematics and the CMM at  $n_w$  waypoints, followed by solving a QP with  $3(n_{sw}-2) \times 6$  optimization parameters. Explicit timing measurements, presented in Section VII, demonstrate that the proposed method is highly efficient and well-suited for integration with an online footstep planner, enabling real-time adaptation to arbitrary step plans.

## VII. EXPERIMENTS

The proposed approach was evaluated in multiple locomotion scenarios using the humanoid robot TORO [27], which has 25 torque-controlled joints, a height of 1.74 m, and a weight of 79.2 kg. The whole-body controller [24] and the synchronous module of the motion planner run at a rate of 1 kHz. Recordings of the corresponding simulations and experiments are provided in the supplementary video.

### A. Experimental Comparison With the Baseline Framework

In the first scenario, the proposed method is compared against a standard 3D-DCM framework baseline with a constant CAM assumption [19], [21], [22], corresponding to iteration #0 in Section VI. In this case, all eCMP offsets are set to zero, and default 5th-order polynomials are employed for the swing leg trajectory, as discussed in Section III. The robot performs straight-line walking with a stride length of 0.3 m, a single support time of 0.7 s, and a double support time of 0.2 s.

The first two plots in Fig. 6 show the reference eCMP trajectory and the measured CoP position in the  $x$ - and  $y$ -directions for both the baseline and the proposed method during a single-support phase on the right foot. It can be observed that the baseline eCMP trajectory is obtained by linearly interpolating between two waypoints (green filled circles) at the beginning and end of the single-support phase, resulting in a constant eCMP position. In contrast, the eCMP trajectory in the proposed approach is generated by linearly interpolating between 20 waypoints spaced at 35 ms intervals.

The measured CoP is computed based on the contact wrench (6), with all quantities expressed relative to the center of the foot. Based on the foot dimensions, the maximum allowable CoP deviation is  $\pm 0.08$  m in the  $x$ -direction and  $\pm 0.04$  m in the  $y$ -direction, as indicated by the horizontal dashed lines. Using the baseline method, the CoP can be seen to deviate significantly from its desired position at the center of the foot, nearly reaching the edge in the  $x$ -direction at  $t = 0.46$  s. In such a case, the foot could tilt, potentially causing the robot to lose balance.

In contrast, the measured CoP error obtained with the proposed method is substantially reduced. As shown in Fig. 4, under ideal conditions without external disturbances, the CoP error can be reduced to almost zero using our approach. In real-world scenarios, however, several factors affect CoP tracking performance, including model uncertainties causing slight variations in CAM, velocity tracking errors, and external forces acting on the robot. Nevertheless, as seen in the right plot of Fig. 6, our method reduces the average CoP error from (4.33, 1.97) cm to (2.55, 1.03) cm, corresponding to

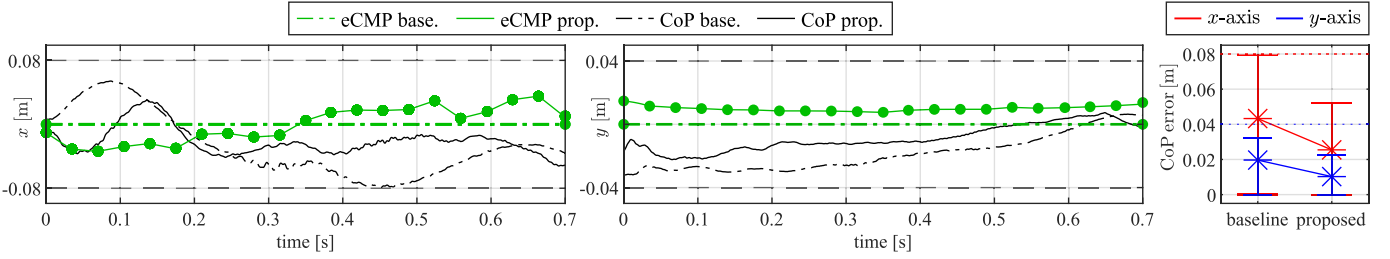


Fig. 6. Baseline comparison for flat-footed straight walking. The first two plots show the reference eCMP trajectories and the measured CoP for the baseline and proposed methods in the  $x$ - and  $y$ -directions, respectively. The right-hand plot presents the CoP error, computed as in Fig. 4, along with the minimum and maximum errors. The dotted lines for each respective axis indicate the limits of CoP deviation.

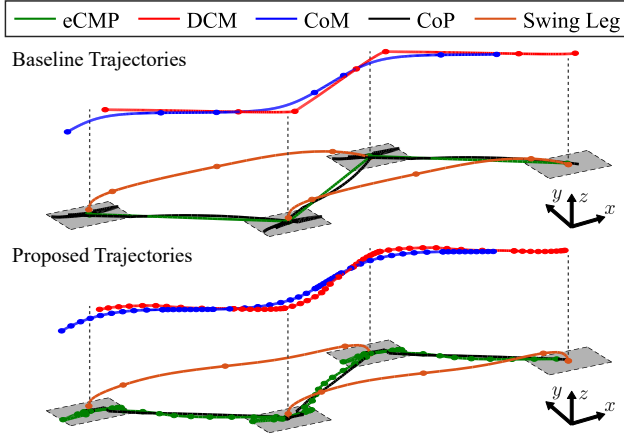


Fig. 7. 3D-DCM quantities and swing leg trajectories for walking with a toe-off phase. The corresponding waypoints are indicated by dots in their respective colors; for clarity, only a subset is shown. The baseline trajectories, generated using the standard 3D-DCM framework, are shown in the top row, while the proposed trajectories are shown in the bottom row. The gray areas indicate the footsteps.

reductions of 41% and 48%, respectively. More importantly, the maximum CoP deviation is significantly reduced, thereby increasing the safety margin relative to the foot edges.

For the baseline method, the walking parameters in this scenario are already at the maximum achievable velocity for flat-footed walking, as the CoP reaches almost the edge of the foot. At higher velocities, the rate of change of CAM increases, causing larger CoP deviation and potential tilting of the robot's feet. Consequently, higher walking speeds for flat-footed motion can only be achieved using our proposed method. Corresponding experiments are provided in the accompanying video.

### B. Rapid Walking

In the next scenario, the robot performs straight-line walking with a toe-off phase. The stride length is increased to 0.92 m, with a single support time of 0.7 s and a double support time of 0.2 s. A total of  $n_w = 30$  eCMP waypoints are used for each step, which consists of a double and consecutive single support phase. The reference trajectories for both the baseline and the proposed method are shown in Fig. 7. These walking parameters yield a speed of 0.51 m/s, representing a 38% improvement over the previous speed record of 0.37 m/s for the humanoid robot TORO [24]. This improvement is made possible by optimizing the swing leg trajectories to fully exploit the hardware capabilities, combined with the eCMP offset computation for enhanced CoP tracking.

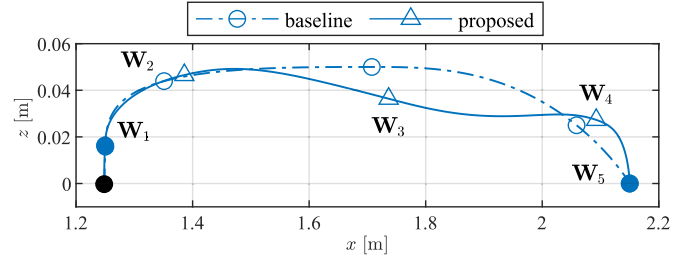


Fig. 8. The swing leg trajectory of the right foot in the  $xz$ -plane is shown. The blue dashed line represents the trajectory generated by the baseline approach, while the solid blue line illustrates the optimized trajectory obtained through waypoint optimization. The first and last waypoints remain fixed (solid blue circles), while the remaining waypoints (open markers) are included in the optimization.

The optimized swing leg trajectory is shown in Fig. 8. We found that five waypoints ( $n_{sw} = 5$ ) provide sufficient expressiveness for the scenarios considered. As shown in Fig. 9, the corresponding joint velocities remain within their limits, in contrast to the trajectory resulting from the baseline approach.

The proposed method involves computing the IK and the CMM at 30 waypoints, as well as solving the swing leg QP. The algorithm takes a total of 6.87 ms to optimize a trajectory of 0.9 s duration, including two iterations of Algorithm 1, on a desktop computer equipped with an Intel i7-11700K CPU.

### C. Further Locomotion Scenarios

The proposed method is not limited to straight-line walking but can be applied to a wide range of locomotion scenarios. We conducted several additional experiments and simulations, including highly dynamic balancing motions, curved walking, flat-footed, turning in place, and walking on rough terrain (see Fig. 1). Footage of these experiments is provided in the accompanying video. As observed in comparison with [7], [21], [24], these experiments would either be infeasible or could only be executed at reduced velocities without the proposed method, demonstrating the generality and effectiveness of our approach.

Finally, we applied the proposed method to running scenarios in simulation. In addition to improved CoP tracking and kinematically feasible swing leg trajectories, the CAM-aware swing leg optimization also reduces undesired upper-body motion by regulating the overall rate of change of CAM to zero (see Fig. 10). The angular momentum generated by the legs compensates for each other, with the CAM induced by the right leg counteracting that of the left leg, particularly during the flight phase.

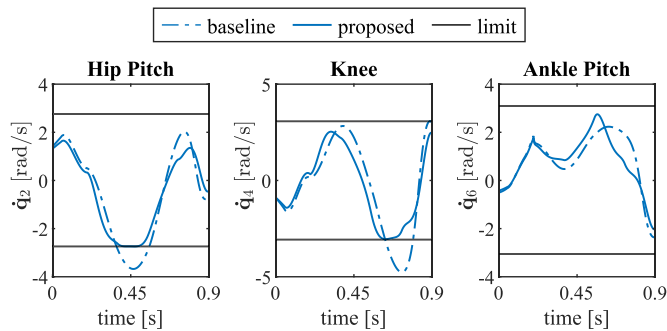


Fig. 9. Joint velocities of the leg joints in pitch configuration, primarily associated with swing leg motion, are shown for one double and single support phase. Velocities for both the baseline and proposed methods are displayed, along with the joint velocity limits.

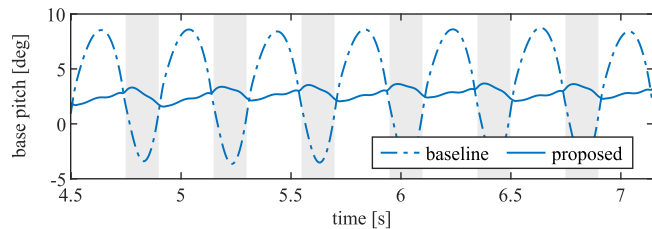


Fig. 10. Pitch motion of the base for both the baseline and proposed trajectories during running. The gray regions indicate the flight phases.

## VIII. CONCLUSION

In this work, we propose an online iterative algorithm that adjusts the center of mass trajectory based on the current rate of change of centroidal angular momentum to improve center of pressure tracking. Our method builds on the 3D-DCM framework, which provides a natural foundation for angular momentum planning through the eCMP. Additionally, an optimized swing leg trajectory ensures that the resulting motions fully exploit the robot's hardware capabilities, thereby maximizing locomotion speed.

Using this approach, we increased the robot's maximum walking speed by 38% compared to its previous record. We demonstrate improved execution speed across multiple scenarios, including balancing, straight walking, turning, and walking on uneven terrain. In running scenarios, undesired upper-body motions are substantially reduced by minimizing the angular momentum induced by the swing legs. In future work, we will further investigate the effect of angular momentum-optimized trajectories on running performance.

## REFERENCES

- [1] 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.*, 2001, pp. 239–246.
- [2] M. Vukobratovic, B. Borovac, and D. Surdilovic, "Zero moment point-proper interpretation and new applications," in *Proc. 1st IEEE-RAS Int. Conf. Humanoid Robots*, 2001, pp. 237–244.
- [3] P. Wensing and D. 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.
- [4] G. Secer and U. Saranlı, "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.

- [5] H. Herr and M. Popovic, "Angular momentum in human walking," *J. Exp. Biol.*, vol. 211, no. 4, pp. 467–481, Feb. 2008.
- [6] R. N. Hinrichs, "Upper extremity function in running. II: Angular momentum considerations," *Int. J. Sport Biomech.*, vol. 3, no. 3, pp. 242–263, 1987.
- [7] R. Schuller, G. Mesesan, J. Engelsberger, J. Lee, and C. Ott, "Online learning of centroidal angular momentum towards enhancing DCM-based locomotion," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2022, pp. 10 442–10 448.
- [8] J. Pratt, J. Carff, S. Drakunov, and A. Goswami, "Capture point: A step toward humanoid push recovery," in *Proc. 6th IEEE-RAS Int. Conf. Humanoid Robots*, Jan. 2006, pp. 200–207.
- [9] K. Guan, K. Yamamoto, and Y. Nakamura, "Virtual-mass-ellipsoid inverted pendulum model and its applications to 3D bipedal locomotion on uneven terrains," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2019, pp. 1401–1406.
- [10] Y. Gong and J. Grizzle, "One-step ahead prediction of angular momentum about the contact point for control of bipedal locomotion: Validation in a LIP-inspired controller," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2021, pp. 2832–2838.
- [11] T. Seyde *et al.*, "Inclusion of angular momentum during planning for capture point based walking," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2018, pp. 1791–1798.
- [12] H. Kaminaga, J. Engelsberger, and C. Ott, "Kinematic optimization and online adaptation of swing foot trajectory for biped locomotion," in *Proc. 12th IEEE-RAS Int. Conf. Humanoid Robots*, 2012, pp. 593–599.
- [13] S. Sovukluk, R. Schuller, J. Engelsberger, and C. Ott, "Realtime limb trajectory optimization for humanoid running through centroidal angular momentum dynamics," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2025, pp. 404–410.
- [14] M. Posa, C. Cantu, and R. Tedrake, "A direct method for trajectory optimization of rigid bodies through contact," *Int. J. Robot. Res.*, vol. 33, no. 1, pp. 69–81, 2014.
- [15] H. Dai and R. Tedrake, "Planning robust walking motion on uneven terrain via convex optimization," in *Proc. 16th IEEE-RAS Int. Conf. Humanoid Robots*, 2016, pp. 579–586.
- [16] A. Herzog *et al.*, "Momentum control with hierarchical inverse dynamics on a torque-controlled humanoid," *Auton. Robots*, vol. 40, no. 3, pp. 473–491, 2016.
- [17] Z. Li *et al.*, "Reinforcement learning for versatile, dynamic, and robust bipedal locomotion control," *Int. J. Robot. Res.*, vol. 44, no. 5, pp. 840–888, 2025.
- [18] I. Radosavovic *et al.*, "Real-world humanoid locomotion with reinforcement learning," *Sci. Robot.*, vol. 9, no. 89, pp. 1–12, 2024.
- [19] J. Engelsberger, 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.
- [20] G. Mesesan, J. Engelsberger, C. Ott, and A. Albu-Schäffer, "Convex properties of center-of-mass trajectories for locomotion based on divergent component of motion," *IEEE Robot. Autom. Lett.*, vol. 3, no. 4, pp. 3449–3456, Oct. 2018.
- [21] G. Mesesan *et al.*, "Motion planning for humanoid locomotion: Applications to homelike environments," *IEEE Robot. Autom. Mag.*, vol. 32, no. 1, pp. 35–48, Mar. 2025.
- [22] G. Mesesan, R. Schuller, J. Engelsberger, C. Ott, and A. Albu-Schäffer, "Unified motion planner for walking, running, and jumping using the three-dimensional divergent component of motion," *IEEE Trans. Robot.*, vol. 39, no. 6, pp. 4443–4463, Dec. 2023.
- [23] D. E. Orin and A. Goswami, "Centroidal momentum matrix of a humanoid robot: Structure and properties," in *Proc. IEEE/RSJ Int. Conf. on Intell. Robots and Syst.*, 2008, pp. 653–659.
- [24] G. Mesesan, J. Engelsberger, 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. 19th IEEE-RAS Int. Conf. Humanoid Robots*, 2019, pp. 25–32.
- [25] R. Schuller, G. Mesesan, J. Engelsberger, J. Lee, and C. Ott, "Online centroidal angular momentum reference generation and motion optimization for humanoid push recovery," *IEEE Robot. Autom. Lett.*, vol. 6, no. 3, pp. 5689–5696, Jul. 2021.
- [26] J. Engelsberger, G. Mesesan, C. Ott, and A. Albu-Schäffer, "DCM-based gait generation for walking on moving support surfaces," in *Proc. 18th IEEE-RAS Int. Conf. Humanoid Robots*, 2018, pp. 1–8.
- [27] J. Engelsberger *et al.*, "Overview of the torque-controlled humanoid robot TORO," in *Proc. 14th IEEE-RAS Int. Conf. Humanoid Robots*, 2014, pp. 916–923.