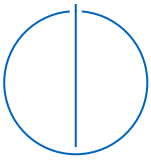
SCHOOL OF COMPUTATION, INFORMATION AND TECHNOLOGY – INFORMATICS

TECHNICAL UNIVERSITY OF MUNICH

Master's Thesis in Robotics, Cognition, Intelligence

Sliding-Window Planning for Realtime Motion Optimization in Humanoid Robots

Daniel Sebastian Ostermeier



SCHOOL OF COMPUTATION, INFORMATION AND TECHNOLOGY – INFORMATICS

TECHNICAL UNIVERSITY OF MUNICH

Master's Thesis in Robotics, Cognition, Intelligence

Sliding-Window Planning for Realtime Motion Optimization in Humanoid Robots

Echtzeitfähige Bewegungsoptimierung für humanoide Roboter

Author:	Daniel Sebastian Ostermeier
Examiner:	Prof. Dr.-Ing. Alin Olimpiu Albu-Schäffer
Supervisor:	M.Sc. George-Adrian Mesesan, M.Sc. Robert Schuller
Submission Date:	03.11.2025

Abstract

Autonomous robotic locomotion, particularly in the context of humanoid robots, has been an ongoing research topic for several decades. In this context, the model-based three-dimensional Divergent Component of Motion (3D-DCM) motion planning framework demonstrates comprehensive capabilities in planning dynamically consistent center-of-mass trajectories for agile motions. However, current methods for 3D-DCM motion planning either assume a constant Centroidal Angular Momentum (CAM) throughout the planned motion, are only able to roughly estimate its influence, or are not applicable to long, incrementally built motion plans as required, e.g., for shared-autonomy walking. This thesis addresses this issue by presenting a sliding-window approach for CAM-based optimization of the centroidal dynamics within the 3D-DCM framework. The multi-body dynamics of the robot are thereby simulated for short segments of the motion plan prior to their actual execution. These simulated preview windows enable a real-time capable motion plan optimization, even in shared autonomy scenarios, where short-term plan extensions are common and input time delays are undesirable. Furthermore, a conceptual path-search algorithm is proposed that is envisioned to enable short-term replanning of stepping sequences, for example, to avoid obstacles given a set of feasible alternative contact areas. The CAM-based optimization is validated both in simulation and reality on the Torque-Controlled Humanoid Robot (TORO) of the German Aerospace Center (DLR).

Contents

Chapter 1

Introduction

1.1 Motivation

Bipedal locomotion, especially in the context of humanoid robots, has been an open research topic for multiple decades [1, Chapter 48]. In recent years, continuous improvements and novel developments on humanoid walking frameworks, as well as various enhancements in, e.g., computational capabilities or machine learning, have led to an increase in interest and available funding for humanoid robotic research in both the scientific community and the private sector. The interplay of currently available frameworks for motion planning, control methods, and hardware components is already able to perform challenging tasks like walking on compliant surfaces, climbing stairs, or counteracting external disturbances [2–5]. Still, the reliable, fast, and robust execution of highly dynamic motions, such as fast walking or running, remains a challenging task in the real world due to limited foot sizes, underactuated system dynamics, or constraints on generatable contact forces and torques.

Apart from physical hardware limitations such as torque and velocity constraints [6], certain assumptions are made in model-based planning and control of such systems. These assumptions are often necessary to simplify parts of the employed mathematical model in order to make its underlying computations real-time capable [1, Chapter 48]. This real-time, or quasi-real-time, execution enables fast reactions to unforeseen events, such as unplanned interactions with the environment, reduces preparation times during planning, or ensures on-time execution of safety measures.

In the Divergent Component of Motion (DCM) locomotion framework [7], simplification is often obtained by neglecting changes in the Centroidal Angular Momentum (CAM), i.e., the torque acting about the Center of Mass (CoM) of the robot [7, 8]. Neglecting this torque—which is induced, e.g., through fast swing-leg motions—without compensation can lead to extraneous contact forces and torques, potentially leading to stumbling and falling. Hence, the usual approach is to compensate for all of the induced centroidal torque by extensive, upper-body motions through the whole-body controller [9], or not compensate for it and accept deviations in the desired Center of Pressure (CoP). While recent methods [9, 10] resemble a remedy for this issue, they lack the necessary flexibility to be generally applicable to arbitrarily extending motion plans, as encountered, e.g., within shared autonomy walking scenarios.

1.2 Related Work

The CAM is long known to play a crucial role in human and humanoid locomotion [11]. Leveraging CAM for posture balancing has been proposed multiple times, e.g., within [5, 12]. The authors in [13] perform a kino-dynamic planning procedure, where they simultaneously optimize the desired task-space trajectories alongside the centroidal linear and angular momentum to reduce discrepancies between the planned and executed motions. However, this method is performed offline and takes multiple minutes to compute trajectories for the horizon of a few seconds. In contrast to [13], where the full multi-body CAM is computed, simplified mass models lead to faster computation times as shown in [14], where a three-mass model is employed to roughly approximate the CAM within the Divergent Component of Motion (DCM) framework. Other approaches extend the linear inverted pendulum model to account for the CAM [15], leverage a preview controller for Zero Moment Point (ZMP) control [16, 17], or use a model-free iterative learning controller [18]. While the authors in [19] showed that an iterative learning controller can likewise be used for Virtual Repellent Point (VRP) optimizations within the DCM framework, e.g., to account for model uncertainties, they do this with an assumption of constant CAM. An accurate, real-time capable estimation for the CAM trajectories within the DCM framework is proposed in [9]. The authors hereby employ an iterative, online learning approach, which leverages the instantaneous CAM samples that can be computed during live motion execution. Their method fits a CAM trajectory during execution to the previously recorded CAM samples and, thus, is able to extrapolate from these observations and update the Enhanced Centroidal Moment Pivot (eCMP) waypoints for the remaining part of its motion trajectories. On the downside, this method heavily relies on strictly repetitive motions and requires some initial observations before the CAM estimation can be performed. Recently, the authors in [10] proposed an iterative optimization of the eCMP and subsequently, the VRP trajectory, based on the full multi-body CAM along with an optimization of the robot's foot trajectories with respect to the exerted CAM. The proposed method is thereby accurate, fast in its computation, and not constrained to any specific gaits or motions. However, it assumes optimization of the 3D-DCM motion plan as a whole, leading to constraints in its applicability to long-term motion planning or incrementally built plans, e.g., as needed for applications in shared autonomy walking.

1.3 Contribution

This thesis proposes a sliding-window approach to incrementally optimize the VRP trajectory of a given 3D-DCM motion plan [2] based on the robot's CAM. The CAM is thereby obtained by computing a short preview window of the robot's multi-body dynamics during the planned motion, which enables an accurate ad-hoc estimate of the actual CAM fluctuations experienced by the system. The obtained CAM quantities are subsequently used for adaptations to the eCMPs as proposed in [10]. In contrast to a

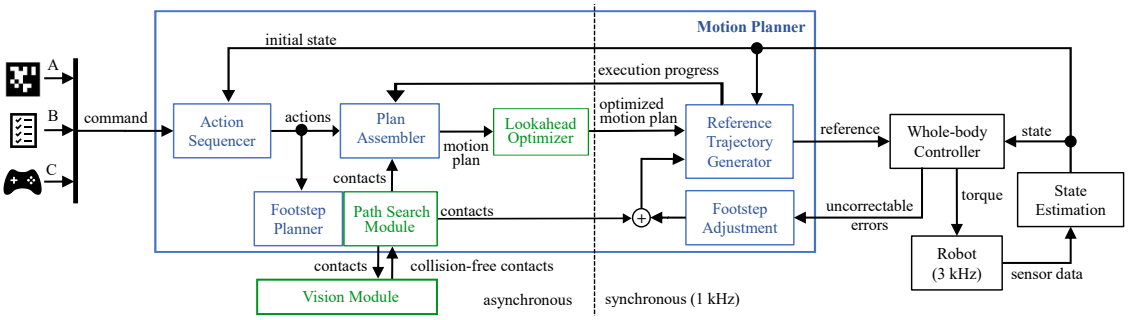


Figure 1.1: Extension of the motion planner architecture presented in [4] by the modules presented in this thesis (highlighted in green).

single optimization conducted in this manner on the entire motion plan, as proposed in previous work [10], the sliding-window approach facilitates the use of CAM-based VRP optimization in long planning. Its reactivity and low input delay make the proposed algorithm particularly well-suited for shared autonomy applications.

The introduced computation of kinematic and dynamic quantities during the planning procedure is subsequently leveraged alongside an informed graph search to enable short-term step-replanning, e.g., for obstacle avoidance, given a sensible set of alternative contacts. This algorithm joins the existing step-adjustment feature for stumble and push-recovery [20] to enhance the motion planner's reactivity in reacting to uncorrectable external events. An overview of the total resulting changes to the underlying motion planning architecture [4] is visualized in Fig. 1.1.

Chapter 2

Mathematical Foundations

The following sections establish a consistent notation and tooling for the mathematical framework used in subsequent chapters. The mathematical concepts presented are largely based on the established literature in [1, 21, 22].

2.1 Notation

The following basic conventions for mathematical notations are employed throughout this thesis:

- Scalars are denoted by lower-case letters, e.g., $c \in \mathbb{R}$.
- Vectors are denoted by bold lower-case letters, e.g., $\boldsymbol{v} \in \mathbb{R}^3$.
- Matrices are denoted by bold upper-case letters, e.g., $\boldsymbol{A} \in \mathbb{R}^{3\times3}$. Identity and zero matrices are denoted by $\mathbf{I}^{n\times m} \in \mathbb{R}^{n\times m}$ and $\mathbf{0}^{n\times m} \in \mathbb{R}^{n\times m}$, where $n, m \in \mathbb{N}_+$.
- Reference frames (or simply *frames*) are denoted as Σ_i, where the subscript corresponds to the name of the frame. A frame consists of an *origin*—a point in 3D Euclidean space—as well as a triplet of orthonormal basis vectors $(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z})$. If not explicitly stated otherwise, right-handed Cartesian coordinate systems are used to represent reference frames. Inertial frames are not distinguished from non-inertial frames by notation, but through the context surrounding the respective equation.
- The notation for points in 3D Euclidean space contains their reference frame as a superscript and their name as a subscript: ${}^{k}\boldsymbol{p}_i$. The frame notation is dropped if it is clear from the surrounding context.
- Newton's notation for differentiation is used for differentiation by time. Leibniz's notation is employed for differentiation by any other variable.
- The skew-symmetric matrix representation for a vector $\boldsymbol{r} \in \mathbb{R}^3$ is given by:

$$\hat{\boldsymbol{r}} = \begin{bmatrix} 0 & -r_z & r_y \\ r_z & 0 & -r_x \\ -r_y & r_x & 0 \end{bmatrix} \in \mathbb{R}^{3\times3} \quad \text{given} \quad \boldsymbol{r} = \begin{bmatrix} r_x \\ r_y \\ r_z \end{bmatrix} \in \mathbb{R}^3 . \tag{2.1}$$

The following physical quantities are defined by the use of the aforementioned convention on mathematical notation:

- Time is represented by the scalar quantity $t \in \mathbb{R}$.
- A translational displacement between two points ${}^{k}\boldsymbol{p}_i$, ${}^{k}\boldsymbol{p}_j$ is denoted as ${}^{k}\boldsymbol{t}_{i,j} \in \mathbb{R}^3$.
- The orientation of Σ_m with respect to Σ_n is parametrized via a three-dimensional vector ${}^{n}\boldsymbol{\theta}_m^{\alpha\beta\gamma} \in \mathbb{R}^3$ of Euler angles.
- Linear velocities are represented by ${}^{k}\boldsymbol{v}_{i,j}$, which denotes the velocity of Σ_j relativ to Σ_i expressed in the coordinate frame Σ_k. If Σ_k coincides with Σ_i, the abbreviated notation ${}^{k}\boldsymbol{v}_j$ is employed. This frame convention is employed for all the following quantities.
- Linear accelerations are represented by ${}^{k}\boldsymbol{a}_{i,j}$.
- Angular velocities are denoted by ${}^{k}\boldsymbol{\omega}_{i,j}$.
- Angular accelerations are denoted by ${}^{k}\dot{\boldsymbol{\omega}}_{i,j}$.
- Forces are denoted by ${}^{e}\boldsymbol{f}_i \in \mathbb{R}^3$.
- Moments are denoted by ${}^{k}\boldsymbol{h}_i = \begin{bmatrix} {}^{k}\boldsymbol{k}_i{}^{T} & {}^{k}\boldsymbol{l}_i{}^{T} \end{bmatrix}^{T} \in \mathbb{R}^6$, with ${}^{k}\boldsymbol{k}_i \in \mathbb{R}^3$ resembling the linear part and ${}^{k}\boldsymbol{l}_i \in \mathbb{R}^3$ resembling the angular part of the momentum.
- Torques are represented via ${}^{k}\boldsymbol{\tau}_i$.
- A pose ${}^{k}\mathbf{P}_i = \begin{bmatrix} {}^{k}\boldsymbol{p}_i{}^{T} & {}^{k}\boldsymbol{\theta}_i^{\alpha\beta\gamma T} \end{bmatrix}^{T} \in \mathbb{R}^6$ resembles the combined position and orientation of a body with respect to a given reference frame.
- Twist coordinates are used to represent linear and angular velocities in a single vector. They are denoted by ${}^{k}\boldsymbol{\xi}_{i,j} = \begin{bmatrix} {}^{k}\boldsymbol{v}_{i,j}{}^{T} & {}^{k}\boldsymbol{\omega}_{i,j}{}^{T} \end{bmatrix}^{T} \in \mathbb{R}^6$.
- Wrenches are denoted by ${}^{k}\boldsymbol{w}_i = \begin{bmatrix} {}^{k}\boldsymbol{f}_i & {}^{k}\boldsymbol{\tau}_i \end{bmatrix}^{T} \in \mathbb{R}^6$.

The individual components of the vector quantities listed above are referenced by adding an additional axis reference as a subscript. E.g., the individual entries of the force vector, i.e., the force in direction of the Cartesian axes (x, y, z), are represented by the scalars ${}^{0}\boldsymbol{f}_{example,x}$, ${}^{0}\boldsymbol{f}_{example,y}$, ${}^{0}\boldsymbol{f}_{example,z} \in \mathbb{R}$. Individual rows or columns of a matrix are referenced according to $\boldsymbol{M}_{i,j}$, where i resembles the referenced row and j resembles the referenced column. The use of : in this context implies all of the respective rows/columns.

The frame notation in the above quantities is dropped whenever the respective frame is clear from the equation's surrounding context.

2.2 Frame Transformations

The location of a body in Euclidean space, with respect to a second reference frame, requires at least six parameters to be fully defined—three of which are used to define its position and another three are used to define its orientation.

2.2.1 Motion in $\mathrm{SO}(3)$

Literature on robot kinematics exhibits a large variety of possible parameterizations for rotations ${}^{m}\boldsymbol{R}_{n} \in \mathrm{SO}(3)$ of the reference frame Σ_n into the orientation of Σ_m.

Euler angles parametrize the rotation ${}^{m}\boldsymbol{R}_{n} \in \mathrm{SO}(3)$ via a three-dimensional vector of angles ${}^{m}\boldsymbol{\theta}_{n}^{\alpha\beta\gamma} = \begin{bmatrix}\alpha & \beta & \gamma\end{bmatrix}^{T} \in \mathbb{R}^3$, where each angle parametrizes one of three consecutive rotations about the Cartesian axes $\boldsymbol{x}$, $\boldsymbol{y}$, and $\boldsymbol{z}$ of the reference frame Σ_n[1]. One hereby distinguishes between two different modes of rotation: *extrinsic* and *intrinsic* rotations. For extrinsic rotations, each individual rotation is applied consecutively, but with respect to the orthonormal basis of a static reference frame that is invariant to the applied rotations. Intrinsic rotations, on the other hand, assume consecutive rotations of a moving reference frame about its own axes. Hence, each of the three individual rotations can change the basis vectors about which the succeeding rotations are applied. It holds that any intrinsic rotation sequence is equal to the reversed sequence performed as extrinsic rotations and vice versa [1, Chapter 2.2.2]. Popular choices among the twelve possible sequences of rotation are $\boldsymbol{z}$-$\boldsymbol{y}'$-$\boldsymbol{x}''$ ($\boldsymbol{x}$-$\boldsymbol{y}$-$\boldsymbol{z}$, extrinsic), $\boldsymbol{z}$-$\boldsymbol{x}'$-$\boldsymbol{z}''$ ($\boldsymbol{z}$-$\boldsymbol{x}$-$\boldsymbol{z}$, extrinsic), and $\boldsymbol{z}$-$\boldsymbol{y}'$-$\boldsymbol{z}''$ ($\boldsymbol{z}$-$\boldsymbol{y}$-$\boldsymbol{z}$, extrinsic). Sequences that contain each of the three axes are sometimes referred to as the *Tait-Bryan*, *Cardan*, or *Yaw, Pitch, Roll* angle convention. Throughout this thesis, the intrinsic $\boldsymbol{z}$-$\boldsymbol{y}'$-$\boldsymbol{x}''$ ($\boldsymbol{x}$-$\boldsymbol{y}$-$\boldsymbol{z}$, extrinsic) rotation convention will be used, as it places the singularities at $\beta = \pm\frac{\pi}{2}$. One representative example where this particular placement of the singularities is useful is the orientation of the robot's base with respect to its surroundings. For bipedal walking, it is highly unlikely that the base orientation reaches $\beta = \pm\frac{\pi}{2}$ given the reference frame's $\boldsymbol{x}$ axis is positive toward the front of the robot, while the $\boldsymbol{z}$ axis is positive in the direction of displacement between the robot's base and its head—i.e., its natural upward direction.

Let in the following c_α, c_β and c_γ resemble the abbreviation for the cosine of α, β, γ and s_α. Further, let s_α, s_β and s_γ be the abbreviation for the sine of α, β and γ respectively. Then the orientation of a reference frame Σ_m with respect to Σ_n can be defined in terms of a rotation matrix acting on the orthonormal basis vectors of Σ_n. This rotation matrix is parametrized by α, β, and γ as shown in Eq. (2.2).

$$
{}^{m}\boldsymbol{R}_{n}(\alpha,\beta,\gamma) = \begin{bmatrix} c_\beta c_\gamma & c_\alpha s_\gamma + c_\gamma s_\alpha s_\beta & s_\alpha s_\gamma - c_\alpha c_\gamma s_\beta \\ -c_\beta s_\gamma & c_\alpha c_\gamma - s_\alpha s_\beta s_\gamma & c_\gamma s_\alpha + c_\alpha s_\beta s_\gamma \\ s_\beta & -c_\beta s_\alpha & c_\alpha c_\beta \end{bmatrix} \in \mathrm{SO}(3) \tag{2.2}
$$

The angle-axis convention defines rotations in Euclidean space via a single angle θ, and a unit vector $\boldsymbol{u} \in \mathbb{R}^3$ about which the orientation by θ is applied, where ${}^{m}\boldsymbol{\theta}_{n}^{\theta\boldsymbol{u}} = \theta\boldsymbol{u} \in \mathbb{R}^3$.

[1] Within this thesis, the three fundamental rotations are parametrized as $\boldsymbol{R}_{\boldsymbol{x}}(\alpha)$, $\boldsymbol{R}_{\boldsymbol{y}}(\beta)$, and $\boldsymbol{R}_{\boldsymbol{z}}(\gamma)$.

As the unit sphere $\mathbb{S}^2$ is fully defined by two parameters, and every 3D unit vector lies on the unit sphere, each unit vector can itself be fully defined by only two parameters [21, Chapter 6.1]. Hence, the total number of parameters of the angle-axis convention is three. Given the angle-axis parameters $\boldsymbol{u}$ and θ, one obtains the respective rotation matrix ${}^{m}\boldsymbol{R}_n \in \mathrm{SO}(3)$ via Rodrigues' rotation formular for the exponential map:

$$ {}^{m}\boldsymbol{R}_n(\boldsymbol{u},\theta) = e^{\hat{\boldsymbol{u}}\theta} = \sum_{k=0}^{\infty} \frac{(\hat{\boldsymbol{u}}\theta)^k}{k!} = \mathbf{I}^{3\times 3} + \hat{\boldsymbol{u}}\sin(\theta) + \hat{\boldsymbol{u}}^2(1-\cos(\theta)) \,. \tag{2.3} $$

While the angular velocity of a rotation parametrized by the angle-axis convention is simply determined via the time derivation of the vector $\theta\boldsymbol{u}$, the relation between the angular velocity and intrinsic $\boldsymbol{z}$-$\boldsymbol{y}'$-$\boldsymbol{x}''$ Euler angles is more intricate as shown in Eq. (2.4).

$$ {}^{m}\boldsymbol{\omega}_n = \boldsymbol{G}\,{}^{m}\dot{\boldsymbol{\theta}}_n^{\alpha\beta\gamma} = \begin{bmatrix} c_\beta c_\gamma & -s_\gamma & 0 \\ c_\beta s_\gamma & c_\gamma & 0 \\ -s_\beta & 0 & 1 \end{bmatrix} \begin{bmatrix} \dot{\alpha} \\ \dot{\beta} \\ \dot{\gamma} \end{bmatrix} = \frac{\mathrm{d}(\theta\boldsymbol{u})}{\mathrm{d}t} \tag{2.4} $$

The angular acceleration ${}^{m}\dot{\boldsymbol{\omega}}_n$ follows from differentiation of Eq. (2.4) with respect to time:

$$ {}^{m}\dot{\boldsymbol{\omega}}_n = \dot{\boldsymbol{G}}\,{}^{m}\dot{\boldsymbol{\theta}}_n^{\alpha\beta\gamma} + \boldsymbol{G}\,{}^{m}\ddot{\boldsymbol{\theta}}_n^{\alpha\beta\gamma} = \frac{\mathrm{d}^2(\theta\boldsymbol{u})}{\mathrm{d}t^2} \,. \tag{2.5} $$

$$ \dot{\boldsymbol{G}} = \begin{bmatrix} -c_\beta s_\gamma \dot{\gamma} - s_\beta c_\gamma \dot{\beta} & -c_\gamma \dot{\gamma} & 0 \\ c_\beta c_\gamma \dot{\gamma} - s_\beta s_\gamma \dot{\beta} & -s_\gamma \dot{\gamma} & 0 \\ -c_\beta \dot{\beta} & 0 & 0 \end{bmatrix} \tag{2.6} $$

2.2.2 Homogeneous and Adjoint Transformations

Every point ${}^{i}\boldsymbol{p}_p$ within the reference frame Σ_i can be described in a different frame Σ_k given that the relative orientation ${}^{i}\boldsymbol{R}_k$ and translation ${}^{i}\boldsymbol{t}_{i,k}$ between Σ_k and Σ_i are known. This transformation can be either described by a rotation with subsequent translation, as shown in Eq. (2.7), or by introducing so-called *Homogeneous Coordinates*. Homogeneous Coordinates enable performing both rotation and translation in a single multiplication with the matrix ${}^{k}\boldsymbol{T}_i$. The respective point or vector is thereby extended by an additional, fourth coordinate as shown in Eq. (2.8).

$$ {}^{k}\boldsymbol{p}_p = {}^{k}\boldsymbol{R}_i\,{}^{i}\boldsymbol{p}_p + {}^{k}\boldsymbol{t}_{i,k} \tag{2.7} $$

$$ \begin{bmatrix} {}^{k}\boldsymbol{p}_p \\ 1 \end{bmatrix} = {}^{k}\boldsymbol{T}_i \begin{bmatrix} {}^{i}\boldsymbol{p}_p \\ 1 \end{bmatrix} \quad \text{where} \quad {}^{k}\boldsymbol{T}_i = \begin{bmatrix} {}^{k}\boldsymbol{R}_i & {}^{k}\boldsymbol{t}_{i,k} \\ \mathbf{0}^{1\times 3} & 1 \end{bmatrix} \tag{2.8} $$

$$ {}^{i}\boldsymbol{T}_k = {}^{i}\boldsymbol{T}_k^{-1} = \begin{bmatrix} {}^{k}\boldsymbol{R}_i^{T} & -{}^{k}\boldsymbol{R}_i^{T}\,{}^{k}\boldsymbol{t}_{i,k} \\ \mathbf{0}^{1\times 3} & 1 \end{bmatrix} \tag{2.9} $$

Similar to the transformation performed on points, twist coordinates can also be transformed between two reference frames Σ_k and Σ_i, employing the so-called *adjoint transformation* $\mathbf{Adj}_{k\boldsymbol{T}_i}$. This transformation is associated with ${}^k\boldsymbol{T}_i$ and it is defined in terms of Eq. (2.10).

$$ {}^k\boldsymbol{\xi}_m = \mathbf{Adj}_{k\boldsymbol{T}_i}\, {}^i\boldsymbol{\xi}_m \quad \text{where} \quad \mathbf{Adj}_{k\boldsymbol{T}_i} = \begin{bmatrix} {}^k\boldsymbol{R}_i & {}^k\hat{\boldsymbol{t}}_{i,k}\, {}^k\boldsymbol{R}_i \\ \mathbf{0}^{3\times3} & {}^k\boldsymbol{R}_i \end{bmatrix} \in \mathbb{R}^{6\times6} \tag{2.10} $$

$$ \mathbf{Adj}_{k\boldsymbol{T}_i}{}^{-1} = \mathbf{Adj}_{i\boldsymbol{T}_k} = \begin{bmatrix} {}^k\boldsymbol{R}_i{}^T & -{}^k\boldsymbol{R}_i{}^T\, {}^k\hat{\boldsymbol{t}}_{i,k} \\ \mathbf{0}^{3\times3} & {}^k\boldsymbol{R}_i{}^T \end{bmatrix} \tag{2.11} $$

Wrenches are dual to twist coordinates, and are consequently transformed using the transposed adjoint [21, Chapter 5.1]. It is important to note that this notation of the transposed adjoint refers to a transpose operation that does not act on the block matrices contained within the adjoint. Transposing the contained rotations and the respective cross-product would lead to the inverse of the desired transformation ${}^k\boldsymbol{T}_j$ and thus to further inconsistencies with respect to the employed superscript/subscript notation.

$$ {}^k\boldsymbol{w}_m = \mathbf{Adj}^T_{k\boldsymbol{T}_i}\, {}^i\boldsymbol{w}_m \quad \text{where} \quad \mathbf{Adj}^T_{k\boldsymbol{T}_i} = \begin{bmatrix} {}^k\boldsymbol{R}_i & \mathbf{0}^{3\times3} \\ {}^k\hat{\boldsymbol{t}}_{k,i}\, {}^k\boldsymbol{R}_i & {}^k\boldsymbol{R}_i \end{bmatrix}. \tag{2.12} $$

The time derivative of an adjoint transformation is obtained by leveraging the *Lie Bracket Matrix* $\mathbf{adj}\,(i)\, {}^k\boldsymbol{\xi}_k$ as proposed in [23].

$$ \mathbf{adj}\left({}^i\boldsymbol{\xi}_k\right) = \begin{bmatrix} {}^i\hat{\boldsymbol{\omega}}_k & {}^i\hat{\boldsymbol{v}}_k \\ \mathbf{0}^{3\times3} & {}^i\hat{\boldsymbol{\omega}}_k \end{bmatrix} \quad \text{and} \quad \mathbf{adj}^T\left({}^i\boldsymbol{T}_k\right) = -\begin{bmatrix} {}^i\hat{\boldsymbol{\omega}}_k & \mathbf{0}^{3\times3} \\ {}^i\hat{\boldsymbol{v}}_k & {}^i\hat{\boldsymbol{\omega}}_k \end{bmatrix} \tag{2.13} $$

$$ \dot{\mathbf{Adj}}_{i\boldsymbol{T}_k} = \mathbf{Adj}_{i\boldsymbol{T}_k}\, \mathbf{adj}\left({}^i\boldsymbol{\xi}_k\right) \quad \text{and} \quad \dot{\mathbf{Adj}}^T_{i\boldsymbol{T}_k} = \mathbf{Adj}^T_{i\boldsymbol{T}_k}\, \mathbf{adj}^T\left({}^i\boldsymbol{\xi}_k\right) \tag{2.14} $$

2.3 Kinematics and Dynamics

This thesis is solely concerned with serial-chain kinematic structures within humanoid robots. Information on the applicability of the presented topics to parallel kinematic chains is available in the respective literature [1, 21, 22]. A serial kinematic chain generally consists of a sequence of joints with links between them. The two respective ends of the chain are thereby connected to the base link and the end effector. The general kinematics of a humanoid can be described via a tree structure, where the root node represents a single base link and the leaves resemble individual end effectors. In the example of DLR's Torque-Controlled Humanoid Robot (TORO)—whose kinematic chain is visualized schematically in Fig. 2.1—this kinematic tree has five leaves representing the end effector orientation of the four limbs (left/right arm and leg), as well as that of the head. Through this tree, one can impose a partial order on the individual joints; i.e., a *parent frame* can be defined for any link except the base. The parent frame of the base link is an arbitrarily placed inertial frame Σ_w.

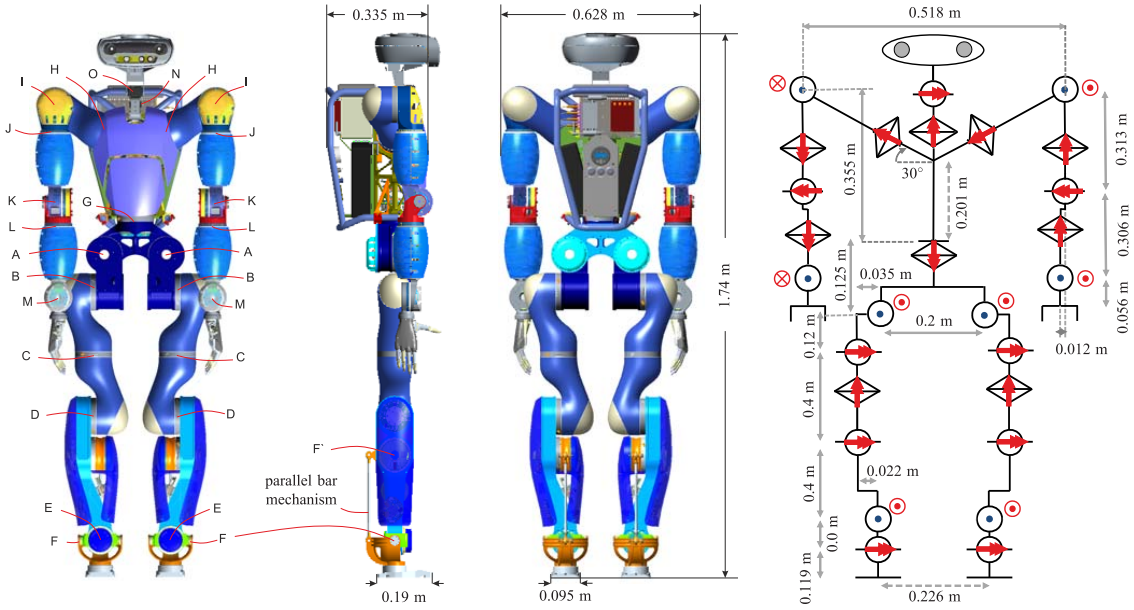


Figure 2.1: Schematic overview of DLR's TORO with its kinematic chain in zero-pose [6]. The parallel bar mechanism at the feet is actuated by a single motor at the joint location F' with the passive joint F attached. For simplification, a single active joint at the location of F is assumed in the robot's kinematic model throughout this thesis.

2.3.1 Forward Kinematics

The forward kinematics of a serial manipulator with a single end effector (abbreviated with "EE" in the subsequent equations), n revolute joints, and joint space $\mathbb{Q}$ define the pose of its end effector with respect to its base as a function of the joint angles $\boldsymbol{\theta} \in \mathbb{Q}$: $\mathrm{FK}(\boldsymbol{\theta}) : \mathbb{Q} \to \mathrm{SE}(3)$. The kinematic chain can thereby be fully defined through the poses of the n joint axes (in zero-position) with respect to the basis frame Σ_{base}, and the static pose of the end effector relative to the last joint axis. Each joint axis can be expressed through a unit vector $\boldsymbol{u}_i \in \mathbb{R}^3$ and a reference point ${}^{base}\boldsymbol{p}_i \in \mathbb{R}^3$ on the axis. Let ${}^{i}\boldsymbol{t}_{i,k}$ denote the translational offset between the reference points of axes i and k. The respective rotation induced by joint i is then given by the rotation matrix ${}^{i-1}\boldsymbol{R}_i(\boldsymbol{u}_i, \theta_i)$ according to Eq. (2.3). The position forward kinematics can be defined according to Eq. (2.15), while the orientation forward kinematics can be defined via Eq. (2.16). The two rotations ${}^{base}\boldsymbol{R}_0$ and ${}^{n}\boldsymbol{R}_{EE}$ are hereby static, i.e., independent of the joint positions, and resemble the orientations of the base frame and the end effector with respect to either start or end of the kinematic chain.

$$ {}^{base}\boldsymbol{t}_{base,EE} = {}^{base}\boldsymbol{t}_{base,1} + \left(\sum_{i=1}^{n-1} {}^{base}\boldsymbol{R}_i \, {}^{i}\boldsymbol{t}_{i,i+1} \right) + {}^{base}\boldsymbol{R}_n \, {}^{n}\boldsymbol{t}_{n,EE} \tag{2.15} $$

$$ {}^{base}\boldsymbol{R}_{EE} = {}^{base}\boldsymbol{R}_0 \left(\prod_{i=0}^{n-1} {}^{i}\boldsymbol{R}_{i+1} \right) {}^{n}\boldsymbol{R}_{EE} \tag{2.16} $$

An alternative approach to the above definition of the forward kinematics is made possible by considering the velocity ${}^{i}\boldsymbol{\xi}_{i+1}$ of the reference point of joint axis $i+1$ induced by the movement of joint i. One obtains the *twist* ${}^{i}\tilde{\boldsymbol{\xi}}_{i+1} \in \mathrm{se}(3)$—an infinitesimal generator of $\mathrm{SE}(3)$—that corresponds to ${}^{i}\boldsymbol{\xi}_{i+1}$ by means of Eq. (2.17). Conversely, the twist coordinates that correspond to ${}^{i}\tilde{\boldsymbol{\xi}}_{i+1}$ are denoted via the *vee* operator "$\vee$" as shown in Eq. (2.18).

$$ {}^{i}\tilde{\boldsymbol{\xi}}_{i+1} = \begin{bmatrix} {}^{i}\hat{\boldsymbol{\omega}}_{i+1} & {}^{i}\boldsymbol{v}_{i+1} \\ \boldsymbol{0}^{1\times 3} & 0 \end{bmatrix} \in \mathbb{R}^{4\times 4} \quad s.t. \quad {}^{i}\boldsymbol{T}_{i+1} = e^{{}^{i}\tilde{\boldsymbol{\xi}}_{i+1}\theta_i} \in \mathrm{SE}(3) \tag{2.17} $$

$$ {}^{i}\boldsymbol{\xi}_{i+1} = {}^{i}\tilde{\boldsymbol{\xi}}_{i+1}^{\vee} = \begin{bmatrix} {}^{i}\hat{\boldsymbol{\omega}}_{i+1} & {}^{i}\boldsymbol{v}_{i+1} \\ \boldsymbol{0}^{1\times 3} & 0 \end{bmatrix}^{\vee} \tag{2.18} $$

The forward kinematic formulation can subsequently be obtained via Eq. (2.19), where ${}^{base}\boldsymbol{T}_0$ and ${}^{n}\boldsymbol{T}_{EE}$ again resemble the static transformation at either end of the kinematic chain.

$$ {}^{base}\boldsymbol{T}_{EE} = {}^{base}\boldsymbol{T}_0 \left(\prod_{i=0}^{n-1} e^{{}^{i}\tilde{\boldsymbol{\xi}}_{i+1}\theta_i} \right) {}^{n}\boldsymbol{T}_{EE} \tag{2.19} $$

2.3.2 End-Effector Jacobians

The spatial manipulator Jacobian for a serial manipulator with a single end effector relates the individual joint velocities to the respective end effector velocities according to Eq. (2.20). While this relation is generally linear in $\boldsymbol{J}(\boldsymbol{\theta})$, the Jacobian itself is typically non-linear with respect to the respective joint configuration $\boldsymbol{\theta}$. The parameter $\boldsymbol{\theta}$ is often omitted for the sake of brevity.

$$ {}^{base}\boldsymbol{\xi}_{EE} = \boldsymbol{J}(\boldsymbol{\theta})\ \dot{\boldsymbol{\theta}} \tag{2.20} $$

One obtains the spatial manipulator Jacobian through differentiation of the forward kinematics function $\mathrm{FK} : \mathbb{Q} \to SE(3)$ with respect to time [21, Chapter 4.1]:

$$ \begin{aligned} \boldsymbol{J}(\boldsymbol{\theta}) = \left[\left(\frac{\partial \mathrm{FK}(\boldsymbol{\theta})}{\partial \theta_1} {}^{EE}\boldsymbol{T}_{base} \right)^{\vee} \cdots \left(\frac{\partial \mathrm{FK}(\boldsymbol{\theta})}{\partial \theta_n} {}^{EE}\boldsymbol{T}_{base} \right)^{\vee} \right] = \begin{bmatrix} {}^{base}\boldsymbol{\xi}_1 & \cdots & {}^{base}\boldsymbol{\xi}_n \end{bmatrix} \\ \text{given} \quad {}^{base}\boldsymbol{\xi}_i = \left((\mathbf{Adj}_{{}^{base}\boldsymbol{T}_{i-1}} {}^{base}\boldsymbol{\xi}_i \right)^{\vee} \quad \text{where} \quad {}^{base}\boldsymbol{T}_{i-1} = \prod_{k=0}^{i-1} e^{{}^{k}\tilde{\boldsymbol{\xi}}_{k+1}\theta_k} . \end{aligned} \tag{2.21} $$

Each column in the Jacobian thus corresponds to the respective joint's twist coordinates transformed to the manipulator configuration given through $\boldsymbol{\theta}$.

Alternatively, one often chooses the body manipulator Jacobian to represent the end effector's relative velocity with respect to the end effector's own coordinate frame. Assuming a serial chain of interconnected joints, where joint p resembles the parent joint of joint k, the body manipulator Jacobian propagates by means of the individual joint's Jacobian ${}^{k}\boldsymbol{J}_{p,k}$ as shown in Eq. (2.22).

$$ {}^{k}\boldsymbol{J}_{base,k}(\boldsymbol{\theta}) = \mathbf{Adj}_{{}^{p}\boldsymbol{T}_k^{-1}} {}^{p}\boldsymbol{J}_{base,p} + {}^{k}\boldsymbol{J}_{p,k} \tag{2.22} $$

The time derivative of the propagated body Jacobian is further given via:

$$ {}^{k}\dot{\boldsymbol{J}}_{base,k}\left(\boldsymbol{\theta}, \dot{\boldsymbol{\theta}}\right) = \mathbf{Adj}_{{}^{p}\boldsymbol{T}_k^{-1}} {}^{p}\dot{\boldsymbol{J}}_{base,p} - \mathbf{adj}\left({}^{p}\boldsymbol{\xi}_k\right) \mathbf{Adj}_{{}^{p}\boldsymbol{T}_k^{-1}} {}^{k}\boldsymbol{J}_{p,k} + {}^{k}\dot{\boldsymbol{J}}_{p,k} . \tag{2.23} $$

The algorithm used for computing the individual joint Jacobians, their derivatives, and the forward kinematics of individual joints is given in Algorithm 1. Apart from the total joint configuration and the respective joint rates, this algorithm expects a matrix $\boldsymbol{\Xi} \in \mathbb{R}^{6 \times n}$ as an input, which contains the individual twist coordinates of a joint. While the algorithm, as stated below, is generally applicable to n-dimensional joints, computation times can be improved by only considering the non-zero dimensions of $\boldsymbol{\Xi}$—for the typical case of a revolute joint, this reduces the number of relevant columns, and thus the number of iterations, to one. This matrix is unique to the respective joint types—individual examples are shown in [1, Chapter 2.3].

The total manipulator body Jacobian is obtained via Algorithm 2, which assumes the zero-pose of the manipulator, and thus the individual joint positions are given in advance along with information on each joint's parent frame. By sorting the input arguments with respect to the partial order defined through the robot's kinematic tree, the correct order of computation is ensured.

Algorithm 1 Computation of the joint Jacobian in accordance with [23]

Input: Joint Configuration $\boldsymbol{\theta} \in \mathbb{R}^n$, Joint Rates $\dot{\boldsymbol{\theta}} \in \mathbb{R}^n$, Joint's Twist-Coord. $\boldsymbol{\Xi} \in \mathbb{R}^{6\times n}$
Output: Joint Jacobian $\boldsymbol{J}$, Joint Jacobian Derivative $\dot{\boldsymbol{J}}$, Joint Transformation ${}^{\mathbf{0}}\boldsymbol{T}_{\boldsymbol{\theta}}$

$\boldsymbol{T} \leftarrow \mathbf{I}^{4\times 4}$
for $i = n, \ldots, 1$ **do**
  $\boldsymbol{\xi}_i \leftarrow \boldsymbol{\Xi}_{:,i}$ ▷ $\boldsymbol{\xi}_i$ resembles the velocity induced by moving joint i
  $\theta_i \leftarrow \boldsymbol{\theta}_i$
  if $\boldsymbol{\xi}_i \neq 0$ **then**
    $\boldsymbol{J}_{:,i} \leftarrow \mathbf{Adj}_{\boldsymbol{T}^{-1}}\, \boldsymbol{\xi}_i$
    $\dot{\boldsymbol{J}}_{:,i} \leftarrow -\mathbf{adj}\left(\boldsymbol{J}\dot{\boldsymbol{\theta}}\right) \mathbf{Adj}_{\boldsymbol{T}^{-1}}\, \boldsymbol{\xi}_i$
    $\boldsymbol{T} \leftarrow \exp\left(\tilde{\boldsymbol{\xi}}_i\, \theta_i\right)\, \boldsymbol{T}$
  else
    $\boldsymbol{J}_{:,i} \leftarrow \mathbf{0}^{6\times 1}, \quad \dot{\boldsymbol{J}}_{:,i} \leftarrow \mathbf{0}^{6\times 1}$
  end if
end for
${}^{\mathbf{0}}\boldsymbol{T}_{\theta} \leftarrow \boldsymbol{T}$
return $\boldsymbol{J}$, $\dot{\boldsymbol{J}}$, ${}^{\mathbf{0}}\boldsymbol{T}_{\theta}$

2.3.3 CoM Jacobian of a Humanoid Robot

In contrast to conventional serial chain manipulators, the kinematic chain of a humanoid has multiple end effectors. Additionally, the base of the robot is not rigidly attached to the environment but free-floating. To circumvent the latter issue, one can think of the floating base as just another link that is rigidly attached via a *virtual* 6-Degrees of Freedom (DoF) joint to an arbitrary but fixed inertial world frame Σ_w. The joint variables are then represented by the six values of the base's pose ${}^{world}\mathbf{P}_{base}$. The Jacobian propagation for this joint then follows the same rules as previously discussed in Section 2.3.2.

Apart from the placement of multiple end effectors—usually one end effector per limb—the CoM kinematics and dynamics also play a crucial role in the motion planning of a humanoid robot (see Section 3). In general, each rigid body attached to a humanoid is either a link that is directly attached to at least one joint or is a third-party rigid body that is rigidly attached to a link. In the second case, the weight and inertia of the respective rigid body are considered to be part of the respective link. Coupled links are thereby not considered for simplicity. Let ${}^{w}\boldsymbol{p}_{\mathrm{CoM},i}$ resemble the position of the CoM of link i in a robot's kinematic chain that contains $N \in \mathbb{N}_+$ links. Further, let $m_{total} \in \mathbb{R}_+$ resemble the total weight of all links in this chain. Then the CoM of the entire robot is defined through Eq. (2.24).

$$ {}^{w}\boldsymbol{p}_{\mathrm{CoM}} = \frac{1}{m_{total}} \sum_{i=1}^{n} {}^{w}\boldsymbol{p}_{CoM_i} \tag{2.24} $$

In addition to the CoM of each link, its inertia (i.e., its mass distribution) must be considered for computations regarding the dynamics of a robot. Throughout this thesis, the inertia of each link is assumed to be given in terms of the constant body inertia matrix $\boldsymbol{M}_k \in \mathbb{R}^{6\times6}$ expressed in the body frame of joint k that is associated with link k. The values for each $\boldsymbol{M}_k$ can be obtained through parameter identification or by use of the robot's Computer-Aided Design (CAD) model [1, Chapter 6.3].

Similarly to the concept of the manipulator Jacobian, one can define a CoM Jacobian $\boldsymbol{J}_{\text{CoM}}$ that relates joint velocities to the velocity of the CoM ${}^{w}\boldsymbol{p}_{\text{CoM}}$ relative to the inertial world frame Σ_w. The virtual 6-DoF joint of the base must hereby be included in both the number of the joints n, as well as the joint configuration space $\boldsymbol{\theta}$ and its derivative $\dot{\boldsymbol{\theta}}$. Let in the following ${}^{w}\boldsymbol{T}_{i_\theta}$ denote the transformation from Σ_w to the frame associated with joint i in the joint configuration $\boldsymbol{\theta}$, and let ${}^{w}\boldsymbol{T}_{i_0}$ denote the same transformation for the zero pose. Further, let ${}^{i_\theta}\boldsymbol{J}_{w,i_\theta}$ denote the body Jacobian of joint i, relating the joint velocities $\dot{\boldsymbol{\theta}}$ to ${}^{i_\theta}\boldsymbol{\xi}_{w,i_\theta}$. The spatial CoM Jacobian is then given via Eq. (2.25) [23]. ${}^{w}\boldsymbol{J}_{CoM}$ hereby only relates the linear velocities of the CoM.

$$
\begin{gathered}
{}^{w}\boldsymbol{J}_{CoM} = \frac{1}{m_{total}} \sum_{i=1}^{n} \begin{bmatrix} \mathbf{I}^{3\times3} & \mathbf{0}^{3\times3} \end{bmatrix} \mathbf{Adj}^{T}_{{}^{w}\boldsymbol{T}_{i_\theta}} \boldsymbol{M}_i \, {}^{i_\theta}\boldsymbol{J}_{w,i_\theta} = \frac{1}{m_{total}} \begin{bmatrix} \mathbf{I}^{3\times3} & \mathbf{0}^{3\times3} \end{bmatrix} \boldsymbol{L} \\
\text{where} \quad {}^{w}\boldsymbol{J}_{CoM} \in \mathbb{R}^{3\times n} \quad \text{and} \quad \boldsymbol{L} \in \mathbb{R}^{6\times n}
\end{gathered}
\tag{2.25}
$$

2.3.4 Inverse Kinematics

Inverse Kinematics (IK) describes the problem of finding joint angles of a kinematic chain for one or more given end-effector pose(s). Let $\boldsymbol{\chi}_{task}$ represent the N-dimensional Cartesian task space of a humanoid robot. The forward and inverse kinematics are then formally defined as:

$$
\begin{aligned}
\text{FK}\left(\boldsymbol{\theta}\right) &: \mathbb{Q} \rightarrow \boldsymbol{\chi}_{task} \\
\text{IK}\left(\boldsymbol{\chi}_{task}\right) &: \boldsymbol{\chi}_{task} \rightarrow \mathbb{Q}\,.
\end{aligned}
\tag{2.26}
$$

In general, the IK problem is only known to be analytically solvable for manipulators with less than six DoF and certain geometric correspondences between their joint axes. In the case of a humanoid robot, neither of these necessary properties is usually fulfilled. Still, analytical methods can be applied in the context of partial sub-chains within the overall kinematic chain of the robot. For example, one can separate the tasks of the upper- and lower-body limbs such that the movement of each arm can be planned independently to follow a certain Cartesian trajectory. In the specific example of German Aerospace Center (DLR)'s TORO, however, at least one joint in each arm would need to be locked in place such that the individual kinematic chain of each arm becomes analytically solvable, e.g., through an automatic decomposition into subproblems as shown in [24]. As this thesis is primarily concerned with locomotion tasks—and thus planning of the CoM and lower-body limb trajectories—a purely numerical method is employed in the following IK computations.

The joint-space of a humanoid robot generally consists of all of its joints—including the virtual 6-DoF at its base. In the case of DLR's TORO, this joint-space can be defined according to Eq. (2.27). As there is no need to perform any motion with the head in the context of this thesis, the respective joint values q_{neck} are set to remain in their zero configuration at all times. Furthermore, the waist of the robot (q_{waist}) is also set to remain in its natural zero configuration. By these assumptions, one can formulate a reduced joint-space $\boldsymbol{\theta}_{reduced}$.

$$\boldsymbol{\theta} := \begin{bmatrix} {}^{w}\mathbf{P}_{base} \\ \boldsymbol{q}_{rightLeg} \in \mathbb{R}^6 \\ \boldsymbol{q}_{leftLeg} \in \mathbb{R}^6 \\ \boldsymbol{q}_{waist} \in \mathbb{R} \\ \boldsymbol{q}_{rightArm} \in \mathbb{R}^6 \\ \boldsymbol{q}_{leftArm} \in \mathbb{R}^6 \\ \boldsymbol{q}_{neck} \in \mathbb{R}^2 \end{bmatrix} \in \mathbb{R}^{33} \quad and \quad \boldsymbol{\theta}_{reduced} = \begin{bmatrix} {}^{w}\mathbf{P}_{base} \\ \boldsymbol{q}_{rightLeg} \\ \boldsymbol{q}_{leftLeg} \\ \boldsymbol{q}_{rightArm} \\ \boldsymbol{q}_{leftArm} \end{bmatrix} \in \mathbb{R}^{30} \tag{2.27}$$

In contrast to the joint space, which resembles a part of the robot's system state, the task-state is related to the motion planning objective. In other words, the motion planning procedure imposes a specific set of tasks that must be fulfilled—up to a certain accuracy—in order for the robot to successfully achieve its planned motion objective. The IK is thereby concerned with mapping a set of Cartesian tasks in the task-space to a valid joint-state configuration that fulfills this task. The set of feasible solutions to the IK problem is therefore given via Eq. (2.28).

$$\mathrm{IK}\left(\boldsymbol{\chi}_{task}\right) = \left\{\boldsymbol{\theta} \in \mathbb{Q} \mid \mathrm{FK}\left(\boldsymbol{\theta}\right) = \boldsymbol{\chi}_{task}\right\} . \tag{2.28}$$

This set is either empty, contains a finite number of solutions, or contains an infinite number of solutions. The solution set is empty if, and only if, the given tasks are placed outside the robot's workspace such that they can not be (simultaneously) fulfilled. The latter is the case if, and only if, the manipulator Jacobian becomes singular, such that $det\left(\boldsymbol{J}\left(\boldsymbol{\theta}\right)\right) = 0$.

For the remainder of this thesis, the task-space vector $\boldsymbol{\chi}_{task}$ of TORO is defined to consist of the CoM ${}^{w}\boldsymbol{p}_{CoM}$, the orientation of the base link in terms of ${}^{w}\boldsymbol{\theta}_{base}^{\alpha\beta\gamma}$, and the poses of the two feet. The tasks for the upper limbs can vary between those formulated directly in joint space and those in Cartesian space. As this thesis is primarily concerned with bipedal locomotion, a formulation of Cartesian tasks for the upper-body limbs often introduces unnecessary complexity if one desires a constant upper-body posture.

$$\boldsymbol{\chi}_{task} := \begin{bmatrix} {}^{w}\boldsymbol{p}_{CoM} \\ {}^{w}\boldsymbol{\theta}_{base}^{\alpha\beta\gamma} \\ {}^{w}\mathbf{P}_{rightFoot} \\ {}^{w}\mathbf{P}_{leftFoot} \\ \boldsymbol{q}_{rightHand} \\ \boldsymbol{q}_{leftHand} \end{bmatrix} \in \mathbb{R}^{30} \quad \text{or} \quad \boldsymbol{\chi}_{task} := \begin{bmatrix} {}^{w}\boldsymbol{p}_{CoM} \\ {}^{w}\boldsymbol{\theta}_{base}^{\alpha\beta\gamma} \\ {}^{w}\mathbf{P}_{rightFoot} \\ {}^{w}\mathbf{P}_{leftFoot} \\ {}^{w}\mathbf{P}_{rightHand} \\ {}^{w}\mathbf{P}_{leftHand} \end{bmatrix} \in \mathbb{R}^{30} \tag{2.29}$$

Based on the task-space $\boldsymbol{\chi}_{task}$ and the joint-space $\boldsymbol{\theta}_{reduced}$, one can further define the (stacked) system Jacobian. In the context of TORO, this system Jacobian is represented by Eq. (2.30). In the case where the upper body limbs' tasks are defined in joint space, one can define $\boldsymbol{J}_{rightArm} = \boldsymbol{J}_{rightArm} = \mathbf{I}^{6\times 30}$ such that the total dimensionality of $\boldsymbol{\chi}$ and $\boldsymbol{\theta}_{reduced}$ can be further reduced to $\mathbb{R}^{15}$.

$$\boldsymbol{J} = \begin{bmatrix} \boldsymbol{J}_{CoM} \\ \mathbf{I}^{3\times 30} \\ \boldsymbol{J}_{rightFoot} \\ \boldsymbol{J}_{leftFoot} \\ \boldsymbol{J}_{rightArm} \\ \boldsymbol{J}_{leftArm} \end{bmatrix} \in \mathbb{R}^{30\times 30} \tag{2.30}$$

Given an initial guess $\boldsymbol{\theta}^{init}$, which is located in the vicinity of the actual solution, the Newton-Raphson method [22, Chapter 6.2.2] is used to iteratively compute the IK solution. The joint-space step $\Delta\boldsymbol{\theta}$ is computed according to Eq. (2.31) in each iteration of the algorithm.

$$\boldsymbol{\theta}_{i+1} = \boldsymbol{\theta}_i + \boldsymbol{J}^{\#}\left(\boldsymbol{\chi}_{task} - FK\left(\boldsymbol{\theta}_i\right)\right) \tag{2.31}$$

One option for $\boldsymbol{J}^{\#}$ is the choice of the damped pseudoinverse as shown in Eq. (2.32). Using this pseudoinverse corresponds to the solution of a damped least-squares problem [1, Chapter 10.3].

$$\boldsymbol{J}^{\#} = \boldsymbol{J}^T\left(\boldsymbol{J}\boldsymbol{J}^T + \lambda^2\,\mathbf{I}^{30\times 30}\right)^{-1} \quad \text{where} \quad \lambda \in \mathbb{R} \quad \lambda << 1 \tag{2.32}$$

Typically, the IK is not only computed once, but repeatedly over time with $\boldsymbol{\chi}_{task}$ following a set of time-dependent trajectories where a good initial guess $\boldsymbol{\theta}_{init}$ at timestep t, can be obtained via the first-order Taylor expansion:

$$\boldsymbol{\theta}_{init,\,t+\Delta t} = \boldsymbol{\theta}_t + \Delta t\dot{\boldsymbol{\theta}}_i\,. \tag{2.33}$$

Chapter 3

DCM-Based Bipedal Locomotion

3.1 CoP and ZMP

The CoP and ZMP are two essential quantities for bipedal locomotion that are often used interchangeably—sometimes mistakenly—when discussing contact wrenches and properties of contacts with the environment. The work in [25] thoroughly describes the definitions, interpretations, and applicability of both quantities and serves as a reference for the following paragraphs.

The CoP is the point in the robot's support polygon, where the exerted moment of the contact's pressure field becomes zero (see Fig. 3.1). It therefore only exists if the robot is in contact with its environment. Given that Σ_C defines a reference frame that is attached to the center point of a robot's support area, it holds that:

$$ {}^{C}\boldsymbol{\tau}_{grf} = {}^{C}\boldsymbol{t}_{C,CoP} \times {}^{C}\boldsymbol{f}_{grf} \,. \tag{3.1} $$

Conversely, the pressure field of all forces and torques acting on the contact can be represented by a single force acting at the CoP. The CoP is obtained via the relation given in Eq. (3.2) between the contact forces/torques, and the contact's normal vector $\boldsymbol{n} \in \mathbb{R}^3$.

$$ {}^{C}\boldsymbol{p}_{CoP} = \frac{\boldsymbol{n} \times {}^{C}\boldsymbol{\tau}_{grf}}{{}^{C}\boldsymbol{f}_{grf} \cdot \boldsymbol{n}} \tag{3.2} $$

The ZMP, on the other hand, resembles the point on the ground where the *tipping moments* (i.e., gravity and inertia) of the robot become zero. It is thus directly obtained

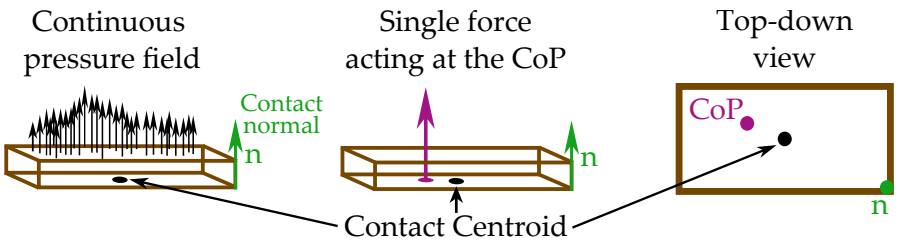


Figure 3.1: The continuous pressure field acting on a contact can be represented via a single force acting at the CoP.

through the dynamics of the robots and can also be computed, for example, during flight phases, e.g., during running, where no support area exists. However, in the context of this thesis, only walking on flat ground is considered, where the ground contacts of both legs appear within the same plane such that the ZMP and CoP always describe the same point.

3.2 Divergent Component of Motion (DCM)

The CoM dynamics of a robot, which are only affected by gravitational and external forces, are a common choice for the central planning and control entity in humanoid locomotion [1, Chapter 48].

In contrast to *simplified* models, such as the linear inverted pendulum (LIP) [1, Chapter 17.4.3], the 3D-DCM resembles a *reduced* model of the centroidal dynamics of a robot via a direct reformulation of Newton's second law of motion. It is generally not limited to bipedal walking, but can also be applied, e.g., to quadrupeds or multi-contact scenarios with humanoids [26]. The formulations within the 3D-DCM framework are applicable even in flight phases, enabling motion planning of centroidal dynamics for running, jumping, or skipping motions [2]. The 3D-DCM locomotion framework [7], which exhibited highly useful properties for planning of complex motions in the past [2–4], splits the unstable second-order CoM dynamics into two first-order dynamics: the stable CoM dynamics related to the DCM, and the unstable DCM dynamics related to the so-called VRP.

Let in the following $\boldsymbol{x} \in \mathbb{R}^3$ resemble the position of the CoM, ${}^{w}\boldsymbol{p}_{CoM}$, with respect to an inertial world frame Σ_w. The 3D-DCM is then defined via Eq. (3.3). The CoM dynamics, which become stable when expressed in terms of the DCM, are formulated according to Eq. (3.4).

$$\boldsymbol{\xi} = \boldsymbol{x} + b\dot{\boldsymbol{x}} \tag{3.3}$$

$$\dot{\boldsymbol{x}} = -\frac{1}{b}\left(\boldsymbol{x} - \boldsymbol{\xi}\right) \tag{3.4}$$

The DCM time constant $b \in \mathbb{R}_+$ is thereby defined according to Eq. (3.5), where $\Delta z \in \mathbb{R}_+$ denotes the design variable that defines the average CoM height above ground, and g resembles the gravitational constant. A generic adaptation of Δz over time is also possible, e.g., through the approach presented in [27].

$$b = \sqrt{\frac{\Delta z}{g}} \tag{3.5}$$

The sum of all external forces acting on the CoM can be encoded in terms of a *repelling force law*. This repelling force is defined through the difference between the CoM and the so-called eCMP $\boldsymbol{e} \in \mathbb{R}^3$. In contrast to the Centroidal Moment Pivot (CMP), the eCMP is not restricted to the robot foot's contact plane, which allows the eCMP to encode both direction and magnitude of the sum of external forces according to Eq. (3.6). The CMP

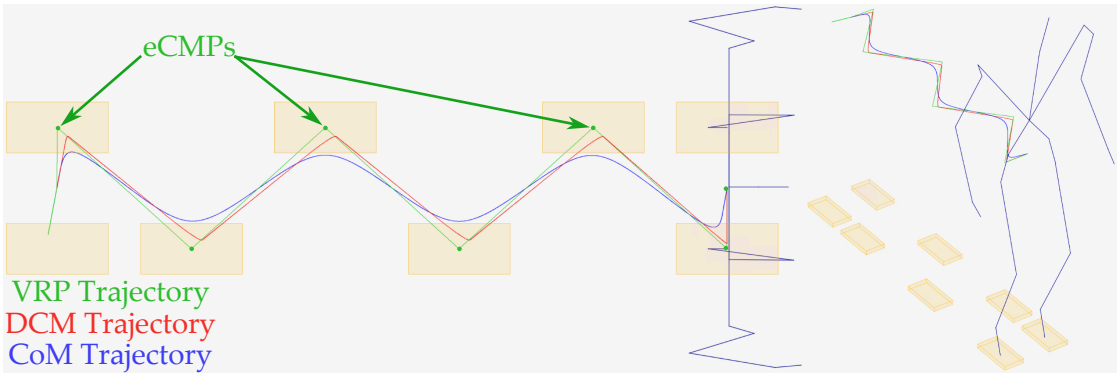


Figure 3.2: Quantities of the 3D-DCM framework visualized during straight walking.

resembles the intersection point between the ground and the line that connects the CoM with the eCMP.

$$\sum f_{external} = \frac{m}{b^2}(\boldsymbol{x} - \boldsymbol{e}) \tag{3.6}$$

In addition to the eCMP, the VRP $\boldsymbol{v} \in \mathbb{R}^3$ is used to concisely express the unstable dynamics of the DCM:

$$\dot{\boldsymbol{\xi}} = \frac{1}{b}(\boldsymbol{\xi} - \boldsymbol{v}) \ . \tag{3.7}$$

The trajectory of the VRP itself is defined through a spatial and temporal linear interpolation of the support points $\boldsymbol{v}_i \in \mathbb{R}^3$, which are located above each of the eCMPs by the vertical offset Δz:

$$\boldsymbol{v}_i = \boldsymbol{e}_i + \begin{bmatrix} 0 & 0 & \Delta z \end{bmatrix}^T \ . \tag{3.8}$$

Intuitively, one can think of the VRP as a quantity "pushing" the DCM away, while the CoM converges towards the DCM. All 3D-DCM quantities are visualized in Fig. 3.2 for the example of straight-walking on flat terrain. One of the major benefits of the previously discussed formulations of the 3D-DCM represents the relatively low-dimensional planning space. In the visualized setting of straight walking, the CoM dynamics can be fully defined through the set of eCMPs placed at the center of the respective foot contacts in addition to the average CoM height Δz.

In order to generate the reference trajectories for the DCM, one leverages the fact that the DCM dynamics in Eq. (3.7) become stable in the case of reversed time, as shown in Eq. (3.9). Intuitively, one thereby starts planning from the goal position of the DCM and integrates the dynamics backwards in time to obtain a DCM trajectory that is inherently stable by design.

$$\dot{\boldsymbol{\xi}}(-t) = -\frac{1}{b}(\boldsymbol{\xi}(-t) - \boldsymbol{v}(-t)) \tag{3.9}$$

Integrating the stable CoM dynamics forward in time—based on the stable DCM trajectory—subsequently allows for the planning of arbitrary long, stable, CoM dynamics. Convex properties of the resulting trajectories are discussed in [28]. An algorithm for

the efficient computation of the DCM and CoM trajectories, from a given set of VRP waypoints, is proposed in [2].

3.3 3D-DCM Motion Planning

The following elaborations on the concept of a motion plan in the context of the 3D-DCM framework were originally proposed in [2–4].

A *motion plan* consists of a set of sub-plans, which describe the individual task-space objectives of the robot's whole-body movement over time. In the context of this thesis, the task-space of the robot, and hence the set of subplans, consists of: the CoM dynamics, the individual limbs' movements, the CAM, as well as the orientation of the robot's base link. Each subplan thereby consists of a series of *motion phases*. Each phase is defined by a duration T_i, an initial phase state, and a final phase state. The initial phase state thereby resembles either the start of the subplan or the final state of the previous motion phase. Hence, every motion phase, except the first one, defines exactly one state in the overall task trajectory.

3.3.1 Phase-State Interpolations for Reference Trajectories

By interpolating between the individual phase states, one obtains the desired task-space *reference trajectory* of the respective subplan. Different types of motion phases thereby require different modes of interpolation between the individual states. The most crucial phases and their interpolations used in this thesis are briefly outlined below.

CoM-Phases
Interpolation between different CoM-phases is done by linearly interpolating the eCMP waypoints—and subsequently the VRP—between the phase's initial and final state. In the context of this thesis, the VRP waypoints are thereby associated with the centroid of a stance, i.e., a set of weighted contacts. The CoM and DCM trajectory for all subphases is then obtained by employing the reverse time computation via the efficient waypoint algorithm proposed in [2]. In contrast to the resulting VRP trajectories, the obtained CoM trajectory is thereby $\mathcal{C}^2$ consistent.

Limb Phases
The upper body limbs of a humanoid are, for the remained of this thesis, considered to be controlled in joint space. Their respective state trajectories are obtained by interpolating between the initial and final joint states via a fifth-order polynomial, which ensures matching velocities and accelerations at the phase boundaries, resulting in an overall $\mathcal{C}^2$-consistent trajectory. For bipedal walking, only the legs of the humanoid are considered to come into contact with the environment and thus exert forces. One thus distinguishes between a limb phase that maintains a ground contact and one where the limb is free to move. In between, attachment and detachment phases ensure a smooth transition between the exerted forces. Phases that resemble a "stepping motion" of the feet are interpolated in Cartesian space. While the Cartesian $\boldsymbol{x}$ and $\boldsymbol{y}$ position, along with

the orientation, can be directly interpolated using a fifth-order polynomial each, the z value involves two separate interpolations: The first one involves interpolating between the initial state's z value and an intermediate point located at the desired stepping height. The second interpolation step finalizes the motion by interpolating from the just-mentioned intermediate point to the phase's final z value. Velocities and accelerations at the beginning and at the end of a stepping motion are set to zero.

Orientation Phases
Analogous to the interpolation used for the orientation of the Cartesian states in the limb phases, the orientation of the robot's base is interpolated by the use of three fifth-order polynomials for roll, pitch, and yaw to match angular velocities and accelerations at the respective initial and final boundaries of the phase.

3.3.2 Planning for Shared Autonomy Walking

In shared autonomy, the high-dimensional action space of a humanoid robot is mapped to a low-dimensional input device, such as a joystick, keyboard, or gaming controller. This enables a human to take on the high-level reasoning process of planning the robot's motions while intuitively commanding the desired movements. That way, human intelligence and algorithmic capabilities complement each other to perform sophisticated and sensible movements on the robotic system. One of the major challenges in such scenarios is ensuring the system's high reactivity to commanded inputs while preventing undesired movements. A timely processing and execution of the commanded actions is thereby crucial to realizing low input delays.

With these requirements in mind, a motion plan can be incrementally built by iteratively sampling the input commands of the respective human interface device. In the context of a gaming controller, the direction of the joystick can be interpreted as the desired direction of the robot's next step and yaw orientation about its vertical axis [4]. By this approach, one continuously generates alternating steps for the left and right foot of the humanoid, one step at a time, based on the respective controller input. A new step is thereby added to the motion plan as soon as the robot starts executing the previous step. This ensures the high reactivity of the performed action to new inputs, thereby allowing the human to intuitively command the robot's motion. As the robot needs to come to a safe stop when no further action is required, an implicit safety step is planned to bring the robot to a safe stop along with each new stepping motion. A visual example of the instantaneous motion plan during walking with a gaming controller is shown in the Chapter on experiments in Fig. 6.11.

3.4 Whole-Body Control

The forces acting on the CoM, as planned within the 3D-DCM for a desired motion plan, must be generated by a coordinated interplay of the robot's individual joints and their respective torques. As previously denoted in Section 3.2, the DCM dynamics

are thereby only stable in reverse time. Thus, during the actual execution of a motion plan, these dynamics must be stabilized by a controller. Thus, the task of a whole-body controller is to generate ground reaction forces to track the desired DCM dynamics, while, in the best case, accomplishing all commanded additional tasks imposed on the system. These tasks may involve exerting a specific force on the robot's environment with one of its limbs, holding a specific whole-body pose, or following a commanded limb trajectory. The passivity-based whole-body torque controller proposed in [3] with additional information provided in [29] is employed within this thesis, along with the whole-body motion optimization proposed in [5]. The latter thereby enables tracking of a desired CAM trajectory by generating the respective momentum via a set of selected joints, e.g., within the upper-body limbs. One of the major benefits of a passivity-based whole-body torque-controlled robot is its capability for compliant and safe human-robot collaborations, as well as a precise exertion of forces, e.g., for interaction with uneven or compliant surfaces.

Chapter 4

Realtime VRP Optimization by Motion Preview

The CAM is long known to play a crucial role in human and humanoid locomotion [11]. Especially during dynamic and agile movements, such as fast walking or running, links with non-negligible masses, and considerable distances from the robot's CoM, are exposed to high velocities. This results in large fluctuations in the angular momentum introduced to the system. Previous work [9] proposed to optimize the VRP trajectory with respect to these changes in the robot's CAM as opposed to compensating for it through extensive, upper-body motions employed by the whole-body controller. However, in order to consider the CAM and its derivative in the motion planning procedure, an accurate estimate of the robot's kinematic and dynamic quantities during execution of the respective motions must be available during the planning process.

The following chapter proposes a sliding-window algorithm that incrementally optimizes the placement of eCMP waypoints during real-time execution, while ensuring a safe and consistent motion plan at all times. The robot's full kinematics and all quantities of its multi-body dynamics relevant to the CAM are thereby considered. As the optimization only acts on the components in x and y, the terms VRP and eCMP optimization may be interchangeably used in the following.

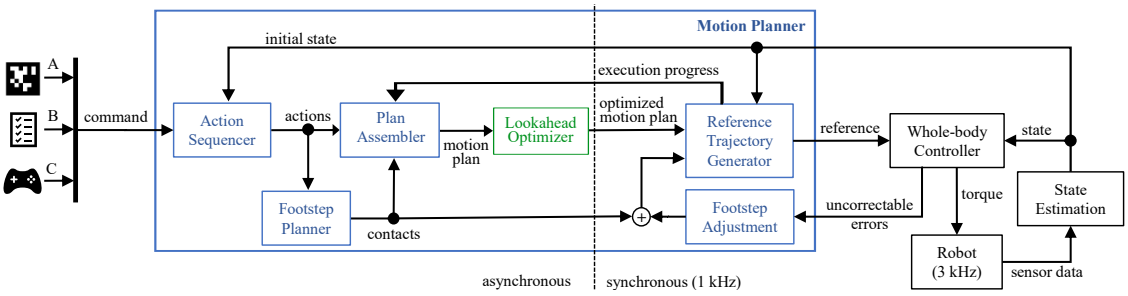


Figure 4.1: Overview of the motion planner architecture presented in [4]. This thesis extends the original architecture (highlighted in blue and black) to include a Lookahead Optimizer module (highlighted in green).

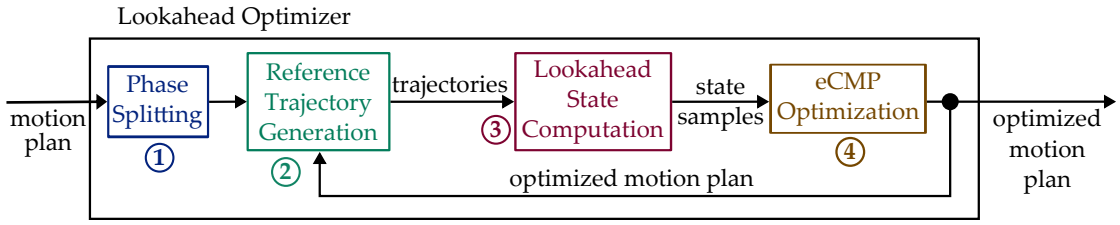


Figure 4.2: The Lookahead Optimizer consists of four distinct steps: Phase Splitting, Reference Trajectory Generation, Lookahead State Computation, and eCMP Optimization. All but the first step are repeated for multiple iterations.

4.1 System Architecture

The 3D-DCM motion planner presented in [4] serves as a baseline for the proposed implementation and consists of an asynchronous and a synchronous component. The objective of the asynchronous component is to provide the synchronous component with a safe, consistent, and feasible motion plan. The synchronous part computes time-continuous task-trajectories for the received motion plan, forwards them to the passivity-based whole-body controller (see Section 3.4), and notifies the asynchronous part about the execution progress. This thesis proposes an extension of the original architecture through a so-called *Lookahead Optimizer* module, as shown in Fig. 4.1. The input to the Lookahead Optimizer is the original motion plan provided through the *Plan Assembler*. The Lookahead Optimizer employs four distinct steps until the respective motion plan is sent to the synchronous part of the motion planner: Phase Splitting, Reference Trajectory Generation, Lookahead State Computation, and eCMP Optimization. As described in Section 3.2, the VRP trajectory is obtained via a piecewise linear interpolation of the eCMPs, in addition to the offset Δz. A schematic overview of the Lookahead Optimizer's structure is shown in Fig. 4.2.

4.1.1 CoM-Phase Splitting

Limited computing resources and real-time requirements do not allow for an arbitrary frequent computation of the robot's whole-body kinematics and dynamics. The Lookahead Optimizer, therefore, is limited to a finite number of sample points along the continuous motion plan at which the respective kinematic and dynamic computations are conducted. As previously discussed in Section 3.2, each CoM phase defines a single eCMP waypoint such that the eCMPs of two consecutive phases can be linearly interpolated. In order to optimize the resulting VRP trajectory based on a sampling density of $N \in \mathbb{N}_+$ samples per second, each CoM phase of duration $T \in \mathbb{R}_+$ has to be split up into $\left\lceil \frac{N}{T} \right\rceil$ individual subphases—each describing the position of a single eCMP.

The synchronous part of the motion planner employs the efficient waypoint computation algorithm presented in [2] to compute the centroidal dynamics for the received motion plan. Due to the synchronous execution of this code at 1kHz, the time available

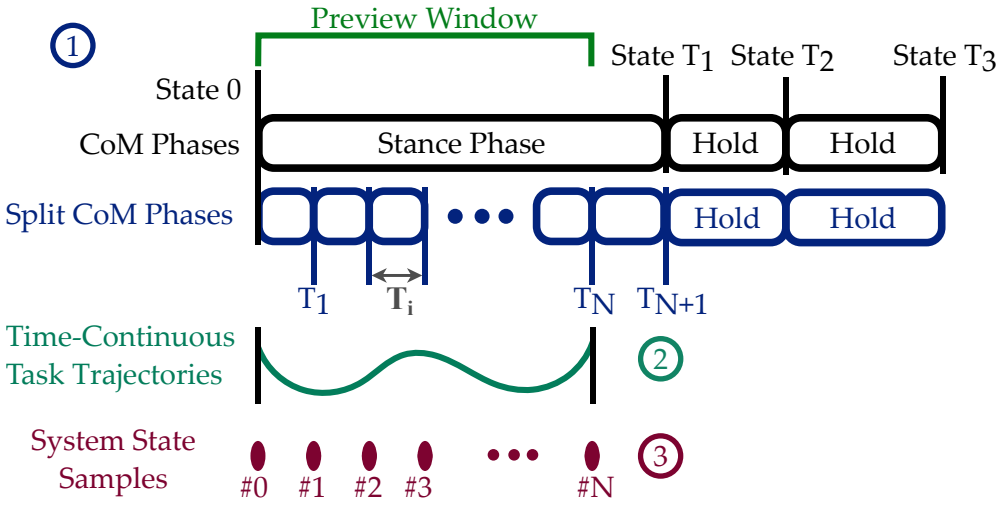


Figure 4.3: Each phase of the CoM plan that is located within the current preview window is split into multiple subphases of duration T_i. If the duration of the original subphase is not a multiple of T_i, the last subphase will be shorter.

for this waypoint computation is highly constrained, while the computational complexity of the efficient waypoint computation is $\mathcal{O}\left(n_\varphi^2\right)$, where n_φ denotes the total number of CoM phases. Based on the measured runtimes in Section 4.2, the total number of CoM phases provided to the synchronous part of the motion planner should not exceed 100 to not risk violating the system's real-time constraints. This significantly limits the total portion of the plan available for optimization, and, along with the input delay discussed in Section 4.2, motivates the use of a sliding window approach for the optimization. The sliding-window procedure is visualized in Fig. 4.3 with a more abstract visualization shown in Fig. 4.4. A discussion on the optimization window's respective parameters—length, sampling rate, and overlap—is conducted in Section 4.3.

The number of subphases obtained after splitting a CoM phase of duration $T \in \mathbb{R}_+$, $\#Subphases$, is obtained via:

$$\#Subphases = \left\lceil \frac{N}{T} \right\rceil \in \mathbb{N}_+ \,. \tag{4.1}$$

The individual duration of each subphase $i \in 1, \ldots, \#Subphases$ is further defined by Eq. (4.2).

$$T_i = \begin{cases} \left\lfloor \frac{T}{\#Subphases} \right\rfloor & , \; i \cdot \left\lfloor \frac{T}{\#Subphases} \right\rfloor < T \\ T - (\#Subphases - 1) \left\lfloor \frac{T}{\#Subphases} \right\rfloor & , \; else \end{cases} \tag{4.2}$$

The final eCMP position $\boldsymbol{e}_{i,T_i}$ at T_i of each subphase i is obtained from the linear interpolation between the original phase's start- and final eCMPs ($\boldsymbol{e}_0$ and $\boldsymbol{e}_T$ respectively) according to Eq. (4.3). The position of the individual eCMPs is thereby expressed relative

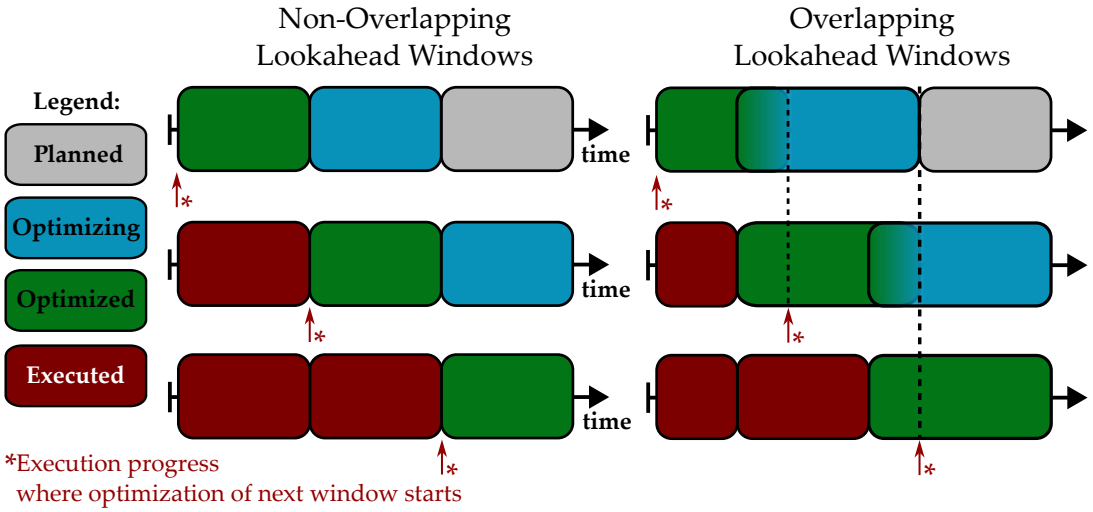


Figure 4.4: Visualization of the sliding-window approach of the Lookahead Optimizer. Each rectangle represents an optimization window on the motion plan, which is either planned, optimizing, optimized, or executing. Optimization of a new window starts when the execution progress reaches the beginning of the previous window. For overlapping windows, parts of the previously optimized plan are also included in the new optimization window.

to the respective foot stance's position to preserve the functionality of online footstep adjustment [20].

$$\boldsymbol{e}_{i,T_i} = \boldsymbol{e}_0 + (\boldsymbol{e}_T - \boldsymbol{e}_0)\,\frac{\sum_{k=1}^{i} T_k}{T} \tag{4.3}$$

4.1.2 Reference Trajectory Generation

The Lookahead Optimizer mirrors the behavior of the Reference Trajectory Generator in the synchronous part of the motion planner, in that it computes the explicit task-trajectories for all limbs and employs the efficient waypoint algorithm to obtain the continuous CoM dynamics of the robot.

4.1.3 Lookahead State Computation

The generated $\mathcal{C}^2$-consistent task-trajectories are evaluated at the timesteps $\{0, T_1, \ldots, T\}$ to obtain a total of $\#Subphases + 1$ consecutive samples of the robot's task-space $\boldsymbol{\chi}_{task,i}$ and its derivative $\dot{\boldsymbol{\chi}}_{task,i}$ within the current optimization window. For each of these samples, the IK is computed according to Section 2.3.4 to obtain the robot's respective joint-space configurations $\{\boldsymbol{\theta}_i\}$. The respective joint rates are then obtained via:

$$\dot{\boldsymbol{\theta}}_i = \boldsymbol{J}^{\#}(\boldsymbol{\theta}_i)\;\dot{\boldsymbol{\chi}}_{task,i}\,. \tag{4.4}$$

Based on the tuple $\left(\boldsymbol{\theta}_i, \dot{\boldsymbol{\theta}}_i\right)$, the CAM ${}^{CoM}\boldsymbol{l}$ and its derivative ${}^{CoM}\dot{\boldsymbol{l}}$ can be computed to perform the eCMP adjustments in Section 4.1.4. In general, the spatial momentum of a single rigid body k is computed as:

$$ {}^{w}\boldsymbol{h}_k = \begin{bmatrix} {}^{w}\boldsymbol{k}_k \\ {}^{w}\boldsymbol{l}_k \end{bmatrix} = \mathbf{Adj}^{T}_{{}^{w}\boldsymbol{T}_k} \boldsymbol{M}_k \, {}^{k}\boldsymbol{\xi}_{w,k} \,. \tag{4.5} $$

Transforming Eq. (4.5) into the CoM's reference frame via the appropriate adjoint transformation yields the centroidal momentum contributed by a single body k to the entire system. The total momentum acting about the CoM is thus computed by simply adding up all moments contributed by the individual links in the robot as shown in Eq. (4.6).

$$ {}^{CoM}\boldsymbol{h} = \sum_{k=1}^{n} \mathbf{Adj}^{T}_{{}^{CoM}\boldsymbol{T}_w} \mathbf{Adj}^{T}_{{}^{w}\boldsymbol{T}_k} \boldsymbol{M}_k \, {}^{k}\boldsymbol{\xi}_{w,k} = \mathbf{Adj}^{T}_{{}^{CoM}\boldsymbol{T}_w} \boldsymbol{L}\,\boldsymbol{\theta} \tag{4.6} $$

The CAM thereby corresponds to the angular part of Eq. (4.6). The CAM-Matrix $\boldsymbol{A} \in \mathbb{R}^{3\times n}$, which maps joint velocities to the whole-body angular momentum expressed in the CoM, can be defined according to Eq. (4.7).

$$ {}^{CoM}\boldsymbol{l} = \begin{bmatrix} \mathbf{0}^{3\times 3} & \mathbf{I}^{3\times 3} \end{bmatrix} \mathbf{Adj}^{T}_{{}^{CoM}\boldsymbol{T}_w} \boldsymbol{L}\,\boldsymbol{\theta} = \boldsymbol{A}\,\boldsymbol{\theta} \tag{4.7} $$

The CAM's time derivative ${}^{CoM}\dot{\boldsymbol{l}}$, which resembles the torque acting about the CoM, is obtained via Eq. (4.8). In practice, $\dot{\boldsymbol{A}}$, is thereby obtained using numerical differentiation in contrast to the analytical solution provided in (4.9), as the latter requires a significantly larger computational effort, while the gained increase in accuracy is negligible in the context of the tracking errors exhibited by the real system (see Section 6.3.1).

$$ {}^{CoM}\dot{\boldsymbol{l}} = \dot{\boldsymbol{A}}\,\boldsymbol{\theta} + \boldsymbol{A}\,\dot{\boldsymbol{\theta}} \tag{4.8} $$

$$ \begin{aligned} {}^{CoM}\dot{\boldsymbol{l}} &= \begin{bmatrix} \mathbf{0}^{3\times 3} & \mathbf{I}^{3\times 3} \end{bmatrix} \left(\left(\dot{\mathbf{Adj}}^{T}_{{}^{CoM}\boldsymbol{T}_w} \boldsymbol{L} + \mathbf{Adj}^{T}_{{}^{CoM}\boldsymbol{T}_w} \dot{\boldsymbol{L}} \right) \boldsymbol{\theta} + \boldsymbol{A}\,\dot{\boldsymbol{\theta}} \right) \\ \text{where} \quad \dot{\boldsymbol{L}} &= \sum_{i=1}^{n} \left(\dot{\mathbf{Adj}}^{T}_{{}^{w}\boldsymbol{T}_{i_\theta}} \boldsymbol{M}_i \, {}^{i_\theta}\boldsymbol{J}_{w,i_\theta} + \mathbf{Adj}^{T}_{{}^{w}\boldsymbol{T}_{i_\theta}} \boldsymbol{M}_i \, {}^{i_\theta}\dot{\boldsymbol{J}}_{w,i_\theta} \right) \end{aligned} \tag{4.9} $$

An algorithm that implements all of the above-mentioned properties for a humanoid's kinematic chain is given in Algorithm 2. The input to this algorithm is the current joint configuration $\boldsymbol{\theta}$, the joint velocities $\dot{\boldsymbol{\theta}}$, transformations from the world frame to each joint frame in zero-pose, the body inertia matrices, as well as the "PARENT" index of each joint, trough which the partial order within the chain is defined. All but the first two parameters are thereby static and independent of the current state of the system.

4.1.4 eCMP Optimization

The desired CoP of a foot contact is typically located within the center of the respective support area, as this ensures a maximum error margin for the CoP until the foot starts to tilt. In an unoptimized motion plan, there typically is one eCMP placed at the center

Algorithm 2 Computation of the manipulator kinematics and spatial CAM matrices [23]

Input: Joint Configuration $\boldsymbol{\theta} \in \mathbb{R}^n$, Joint Rates $\dot{\boldsymbol{\theta}} \in \mathbb{R}^n$, Zero-Pose Transforms ${}^{w}\boldsymbol{T}_{i_0}$, Body Inertias $\boldsymbol{M}_i \in \mathbb{R}^{\times 6\times 6}$, Parent Index Map $\textsc{Parent}(i)$, Body CoM ${}^{i_0}\boldsymbol{p}_{CoM}$.

Output: Body Jacobians $\mathcal{J}$, Link Transforms $\mathcal{T}$, Spatial CAM Matrix $\boldsymbol{L}$, CoM Jacobian $\boldsymbol{J}_{\text{CoM}}$, Multi-body CoM ${}^{w}\boldsymbol{p}_{CoM}$.

$\mathcal{T}[1..n]$, $\mathcal{J}[1..n]$ ▷ initialize as empty arrays
for $i = 1, \ldots, n$ **do**
  $p \leftarrow \textsc{Parent}(i)$ ▷ parent joint index
  ${}^{w}\boldsymbol{T}_{p} \leftarrow \mathcal{T}[p]$ ▷ world-to-parent transform from previous iteration
  ${}^{p}\boldsymbol{J}_{w,p} \leftarrow \mathcal{J}[p]$ ▷ parent Jacobian from previous iteration

  ${}^{i_\theta}\boldsymbol{J}_{i_0,i_\theta}$, $\dot{\boldsymbol{J}}_{i_0,i_\theta}$, ${}^{i_0}\boldsymbol{T}_{i_\theta} \leftarrow$ Algorithm 1 ▷ compute local joint Jacobian and transform
  ${}^{i_\theta}\boldsymbol{\xi}_{i_0,i_\theta} \leftarrow {}^{i_\theta}\boldsymbol{J}_{i_0,i_\theta}\dot{\boldsymbol{\theta}}$ ▷ compute local joint twist

  ${}^{p}\boldsymbol{T}_{i_0} = {}^{w}\boldsymbol{T}_{p}^{-1}\,{}^{w}\boldsymbol{T}_{i_0}$ ▷ parent-to-current joint in zero pose
  ${}^{p}\boldsymbol{T}_{i_\theta} \leftarrow {}^{p}\boldsymbol{T}_{i_0}\,{}^{i_0}\boldsymbol{T}_{i_\theta}$ ▷ parent-to-current joint transform
  ${}^{w}\boldsymbol{T}_{i_\theta} \leftarrow {}^{w}\boldsymbol{T}_{p}\,{}^{p}\boldsymbol{T}_{i_\theta}$ ▷ world-to-current joint transform
  $\mathcal{T}[i] \leftarrow {}^{w}\boldsymbol{T}_{i_\theta}$ ▷ save transform in list

  ${}^{i_\theta}\boldsymbol{J}_{w,i_\theta} \leftarrow \mathbf{Adj}_{{}^{p}\boldsymbol{T}_{i_\theta}^{-1}}\,{}^{p}\boldsymbol{J}_{w,p} + {}^{i_\theta}\boldsymbol{J}_{i_0,i_\theta}$ ▷ Body Jacobian at joint i

  $\mathcal{J}[i] \leftarrow {}^{i_\theta}\boldsymbol{J}_{w,i_\theta}$ ▷ save Jacobian in list

  $\boldsymbol{L} \leftarrow \boldsymbol{L} + \mathbf{Adj}_{{}^{w}\boldsymbol{T}_{i_\theta}}^{T}\,\boldsymbol{M}_i\,{}^{i_\theta}\boldsymbol{J}_{w,i_\theta}$ ▷ accumulate spatial CAM matrix

  $m_i \leftarrow \boldsymbol{M}_{i,1,1}$ ▷ total mass of body i
  ${}^{w}\boldsymbol{p}_{CoM} \leftarrow {}^{w}\boldsymbol{p}_{CoM} + m_i \cdot {}^{w}\boldsymbol{T}_{i_\theta}\,{}^{i_\theta}\boldsymbol{T}_{i_0}\,{}^{i_0}\boldsymbol{p}_{CoM}$
  $m_{total} \leftarrow m_{total} + m_i$
end for
$\boldsymbol{J}_{\text{CoM}} \leftarrow \frac{1}{m_{\text{total}}} \cdot \boldsymbol{L}_{\mathbf{1:3},:}$ ▷ linear part of $\boldsymbol{L}$ (first three rows) resembles the scaled $\boldsymbol{J}_{CoM}$
${}^{w}\boldsymbol{p}_{CoM} \leftarrow \frac{1}{m_{\text{total}}} \cdot {}^{w}\boldsymbol{p}_{CoM}$

return $\mathcal{J}$, $\mathcal{T}$, $\boldsymbol{L}$, $\boldsymbol{J}_{\text{CoM}}$, ${}^{w}\boldsymbol{p}_{CoM}$

of any given contact area. This eCMP placement is justified through Eq. (4.10), which relates the contact torques $\boldsymbol{\tau}_{grf}$ to the CoP $\boldsymbol{p}_{CoP_{desired}}$ [9].

$$\begin{aligned}\boldsymbol{\tau}_{grf} &= (\boldsymbol{x} - \boldsymbol{p}_{CoP_{desired}}) \times \boldsymbol{f}_{grf} + {}^{CoM}\dot{\boldsymbol{l}} \\ &\overset{(3.6)}{=} \frac{m}{b^2}(\boldsymbol{x} - \boldsymbol{p}_{CoP_{desired}}) \times (\boldsymbol{x} - \boldsymbol{e}) + {}^{CoM}\dot{\boldsymbol{l}}\end{aligned} \tag{4.10}$$

By choosing coinciding $\boldsymbol{p}_{CoP_{desired}}$ and $\boldsymbol{e}$, the cross product (i.e., the predominant term) becomes zero, such that Eq. (4.10) simplifies according to Eq. (4.11).

$$\boldsymbol{\tau}_{grf} = \frac{m}{b^2}(\boldsymbol{x} - \boldsymbol{e}) \times (\boldsymbol{x} - \boldsymbol{e}) + {}^{CoM}\dot{\boldsymbol{l}} = {}^{CoM}\dot{\boldsymbol{l}}\,. \tag{4.11}$$

Thus, the contact torques are solely defined by means of the time derivative of the CAM. It follows from Eq. (4.10) that the relation between the actual and the desired CoP is defined in terms of Eq. (4.12), where $\ddot{z}$ corresponds to the vertical CoM acceleration.

$$\boldsymbol{p}_{CoP_{actual}} - \boldsymbol{p}_{CoP_{desired}} = \frac{1}{m(g + \ddot{z})} \begin{pmatrix} \tau_{grf,y} \\ -\tau_{grf,x} \\ 0 \end{pmatrix} \tag{4.12}$$

Inserting Eq. (4.11) into Eq. (4.12) subsequently yields a direct relation between the time derivative of the CAM ${}^{CoM}\dot{\boldsymbol{l}}$ and the CoP deviation, given that the eCMPs were placed at the desired CoPs. One option to minimize the deviation between the desired and actual CoP per this formula is an active regulation of the CAM through the whole-body controller, e.g., through motion in the upper-body limbs [5]. Alternatively, one can alter the respective eCMPs, i.e., optimize the external forces, such that the cross-product term cancels out with the derivative of the CAM in Eq. (4.10). The latter is proposed in [10] on the basis of Eq. (4.13) [27].

$$\boldsymbol{e} = \boldsymbol{p}_{CoP_{desired}} + \frac{1}{m\,(g + \dddot{z})} \begin{bmatrix} \dot{l}_{CoM,y} \\ -\dot{l}_{CoM,x} \\ 0 \end{bmatrix} \tag{4.13}$$

4.1.5 CAM Reference Trajectories

Without CAM-optimized eCMP placements, the desired trajectories of the CAM are either not tracked at all, or are set to zero to reduce the contact torques according to (4.11). In practice, only the CAM about the $\boldsymbol{z}$ axis is regulated to zero (in the case of straight walking) to avoid contact slippage without counteracting the CAM about the $\boldsymbol{x}$ and $\boldsymbol{y}$ axis. By shifting the eCMPs based on Eq. (4.10), according to the computed CAM in $\boldsymbol{x}$ and $\boldsymbol{y}$ direction, this computed CAM implicitly becomes the desired CAM:

$$\begin{aligned}{}^{CoM}\dot{l}_{x,desired} &= {}^{CoM}\dot{l}_{x} \\ {}^{CoM}\dot{l}_{y,desired} &= {}^{CoM}\dot{l}_{y} \\ {}^{CoM}l_{x,desired} &= {}^{CoM}l_{x} \\ {}^{CoM}l_{y,desired} &= {}^{CoM}l_{y}\,.\end{aligned} \tag{4.14}$$

The CAM acting about the z direction, however, must be explicitly computed. Recalling, (4.10), the resulting contact torque about the z axis is given as:

$$\begin{aligned} \tau_{grf,z} &= \left(x\, f_{grf,y} - y\, f_{grf,x}\right) - \left(p_{CoP_{desired},x}\, f_{grf,y} - p_{CoP_{desired},y}\, f_{grf,x}\right) + {}^{CoM}\dot{l}_z \\ \text{where} \quad f_{grf,x} &= \frac{m}{b^2}\left(x - e_x\right),\ f_{grf,y} = \frac{m}{b^2}\left(y - e_y\right) \quad \text{given} \quad \boldsymbol{x} = \begin{bmatrix} x & y & z \end{bmatrix}^T . \end{aligned} \tag{4.15}$$

Inserting $\tau_{grf,z} = 0$ into Eq. (4.15) yields the desired derivative of the CAM on the z axis. The corresponding CAM is obtained via numerical integration [9]:

$$\begin{aligned} {}^{CoM}\dot{l}_{z,desired} &= \left(p_{CoP_{desired},x}\, f_{grf,y} - p_{CoP_{desired},y}\, f_{grf,x}\right) - \left(x\, f_{grf,y} - y\, f_{grf,x}\right) \\ {}^{CoM}l_{z,desired} &= \int {}^{CoM}\dot{l}_{z,desired}\, \mathrm{dt}\,. \end{aligned} \tag{4.16}$$

The denoted CAM reference trajectories are tracked via the whole-body motion optimizer proposed in [5]. This motion optimizer utilizes a selected set of joints—usually the upper-body limbs—to generate any additional angular momentum necessary for tracking the respective CAM trajectories. A constraint quadratic optimization is thereby employed to generate the desired angular momentum while minimizing the error to the respective joint's task-space objective (e.g., maintaining a constant upper-body posture).

4.2 Runtime Environment

This thesis builds on top of multiple years of research accompanying the DLR's humanoid robot TORO. To benefit from existing implementations of previous work, the choice of runtime environments and programming languages for implementing the proposed concepts in this thesis is made with consideration for legacy code and existing workflows. The synchronous part of the motion planner proposed in [2, 4], as well as the whole-body controller of [3], is implemented using Matlab/Simulink (R2015b) and running at a frequency of 1 kHz. The asynchronous portion of the motion planner, and thus the system architecture in which the proposed Lookahead Optimizer module is contained, is implemented in Java SE 8. For an efficient computation of linear-algebra operations, the Efficient Java Matrix Library (EJML) (v0.41) is employed. For the following benchmarks, an Intel Core i7-10700K CPU (3.8 GHz) with 32 GiB of RAM was used, running the openSUSE Leap Linux distribution. All provided runtime measurements were conducted using the most precise available system timer.

The average computation time on this system for running the eCMP optimization for a single iteration—broken down to a single sample—consists of the computation time required for the IK, as well as the computation of the CAM and its derivative. Additionally, some time elapses during phase splitting, the actual adaptation of the eCMPs, and the computation of the CoM trajectory after each iteration. However, these computation durations are negligible individually and are combined into a single miscellaneous (MISC) duration $T_{\text{optimMISC}}$:

$$\begin{aligned} T_{\text{optimSample}} &\approx \#Iterations \cdot \left(T_{\text{IK}} + T_{\text{CAM}}\right) + T_{\text{optimMISC}} \\ &\approx \#Iterations \cdot \left(130 \cdot 10^{-6}\text{s} + 105 \cdot 10^{-6}\text{s}\right) + 20 \cdot 10^{-6}\text{s}\,. \end{aligned} \tag{4.17}$$

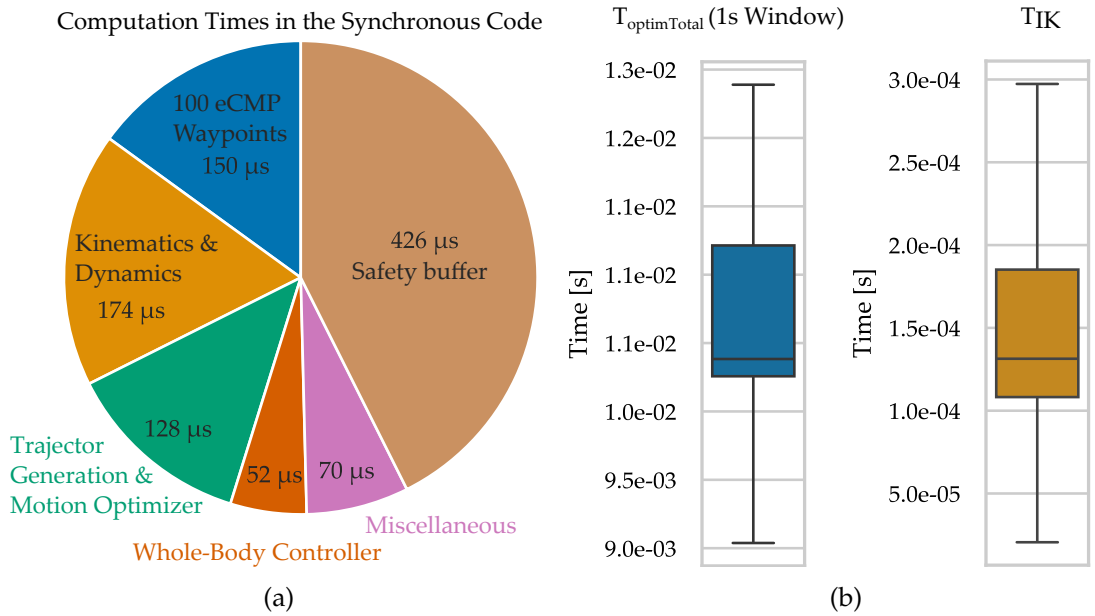


Figure 4.5: (a) Computation time of the motion planner's synchronous part, considering the upper limit of $n_\varphi = 100$ maximal CoM phases. (b) Optimization time of a Lookahead window of one second with 25 samples at two optimization iterations, next to the IK computation times of each point within it.

The average computation time of Algorithm 2 is 92μs. The computation of the CAM (and its derivative) requires a single run of Algorithm 2, computation of the joint rates according to (4.4), and taking the numeric derivative of the CAM matrix. The total average time spent for a single sample for this computation is $T_{\text{CAM}} = 105\mu$s. The IK leverages the joint rates computed from the previous CAM sample to start with an educated initial guess, as shown in (2.33). Often, this initial guess is good enough to avoid the expensive computation of the IK update step according to (2.31), which, along with the computation of the Jacobian's pseudo-inverse, contains a single run of Algorithm 2. The IK takes an average 0.79 iterations to converge with a resulting total average computation time of $T_{\text{IK}} = 130\mu$s (see Fig. 4.5b). The MISC computations take an average of $T_{\text{optimMISC}} = 20\mu$s. Assuming two optimization iterations are employed, this results in $T_{\text{optimSample}} \approx 0.5 \cdot 10^{-3}$s. At 25 samples per second, the total optimization of a one-second-long window takes about $T_{\text{optimTotal}} = 13 \cdot 10^{-3}$s as shown in Fig. 4.5b.

The synchronous code is being executed on one of TORO's onboard computers, which is equipped with an Intel Core i7-2715QE Processor (3 GHz) [6]. The waypoint computation algorithm [2] takes 150μs for a total of $n_\varphi = 100$ phases (corresponding to 100 eCMP waypoints). As the synchronous code needs to run at 1kHz, the added burden of these additional phases induced by the eCMP optimization is significant. However, the total runtime of the synchronous code in Fig. 4.5a still contains an average safety-buffer of more than 40% that keeps it from violating its real-time constraints.

4.3 Parametrization of Lookahead Windows

The essential component that differentiates this thesis from the previously proposed algorithm in [10] is the use of a variable-sized optimization window. To achieve real-time capability, e.g., for deployment in shared autonomy (Section 3.3.2), the parameters of the eCMP optimization algorithm must harmonize with those chosen for the optimization window and vice versa. The total available parameters of the algorithm are:

- *Window Size*: Temporal portion of the motion plan to be optimized
- *Window Overlap*: Percentage of overlap between two consecutive windows
- *Sampling Rate*: eCMP samples per second considered in the optimization
- *Optimization Iterations*: Optimization iterations undergone in each window .

The following sections elaborate on the effect of the respective parameters and provide reasonable default values for each.

4.3.1 Window Size

Due to the reverse time computation of the DCM (see Eq. (3.9)), every value along the DCM trajectory is only influenced by future values in the DCM and thus only through future values of the VRP/eCMP. Adjusting future eCMPs in a currently executing motion plan—as done by the eCMP optimization algorithm—thus induces a discontinuity between the *instantaneous* DCM that defines the dynamics of the running system at time t, and the *planned* DCM at the same time instance t in the updated motion plan. This effect becomes weaker the longer the time span T_{update} between t and the time instance of the motion plan where the adjustment happens, i.e., t_{adjust}. This effect plays a crucial role for large eCMP adaptions, e.g., as needed to extend a given motion plan by additional steps [2]. However, the DCM discontinuity induced by the eCMP optimization algorithm becomes negligibly small for any $T_{\text{update}} > 0.5$s as visualized for the example of an eCMP optimization in a straight walk in Fig. 4.6.

As the optimization procedure for a new window starts as soon as the system starts executing the first phase of the previously optimized window (see Fig. 4.4), T_{update} does correspond to the parametrized window size, but also depends on the time spent for the optimization of the window itself $T_{\text{optimTotal}}$, as well as the time it takes for the motion plan to arrive at the whole-body controller—including, e.g., communication delay or reference trajectory generation on the synchronous part of the motion planner—denoted T_{MISC}. A reasonable assumption for T_{MISC} is located in the magnitude of a few milliseconds or less—thus it is negligibly small as $T_{\text{MISC}} << T_{\text{optimTotal}} << 0.5$s. Therefore, it is reasonable to represent T_{update} only in terms of the window size and the runtime for the optimization itself:

$$T_{\text{update}} \approx T_{\text{window}} - T_{\text{optimTotal}} \tag{4.18}$$

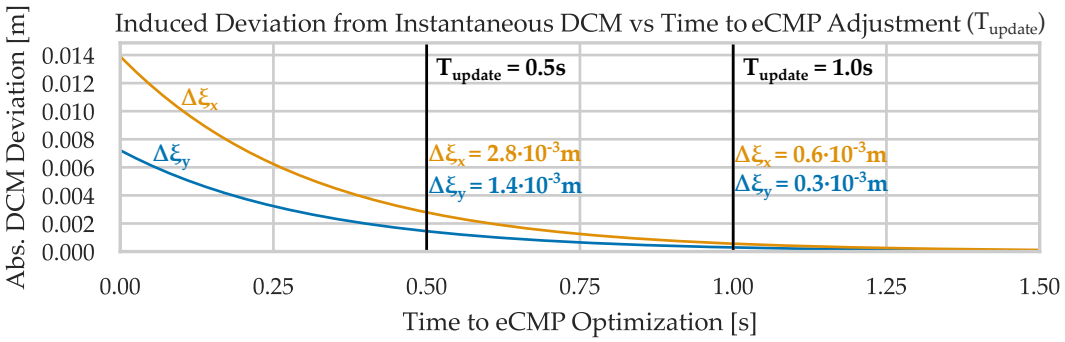


Figure 4.6: Deviation between the planned DCM and the instantaneous DCM in the system, induced through an optimization in the motion plan happening T_{update} seconds from the current system time t. (Motion) Parameters: Five steps at 15cm stepsize, single-support: 0.7s, double-support: 0.2s, 25 Samples/s, two iterations.

Given $T_{\text{update}} \overset{!}{>} 0.5\text{s}$, the minimum window size is defined according to (4.19), where $N \in \mathbb{N}_+$ resembles the number of samples per second and $T_{\text{optimSample}}$ resembles the time it takes for the eCMP optimization per sample (see Section 4.2).

$$\begin{aligned} T_{\text{window}} &\overset{!}{>} 0.5\text{s} + T_{\text{optimTotal}} \quad \text{where} \quad T_{\text{optimTotal}} \approx T_{\text{window}} \cdot N \cdot T_{\text{optimSample}} \\ &\text{s.t.} \quad T_{\text{window}} \overset{!}{>} \frac{0.5\text{s}}{1 - N \cdot T_{\text{optimSample}}} \end{aligned} \tag{4.19}$$

The upper limit of T_{window} is defined through the number of phases n_φ—corresponding to the individual samples—that the synchronous part of the motion planner can consider without running into the risk of violating its real-time constraints. In the "worst-case" scenario, the eCMP optimization is conducted more or less instantly[1], such that the synchronous part of the motion planner must consider all optimized CoM phases from the previous optimization window, along with the phases added by the current optimization window[2]:

$$T_{\text{window}} < \frac{0.5 \cdot n_{\varphi_{\max}}}{N}\,. \tag{4.20}$$

As discussed in the respective subsections of this chapter, reasonable values for the free optimization parameters are $N = 25$, $T_{\text{optimSample}} = 0.5 \cdot 10^{-3}\text{s}$, and $n_{\varphi_{\max}} = 100$. Thus, a reasonable range for the optimization window size is:

$$0.53\text{s} < T_{\text{window}} < 2\text{s}\,. \tag{4.21}$$

Note that T_{window} resembles the *desired* window size. The actual employed window size may deviate, e.g., if the duration T of the total motion plan is shorter than T_{window}. Also,

[1] As $T_{\text{optimTotal}} >> 0$, this is highly unlikely.
[2] Unoptimized (future) parts of the motion plan are not considered for $n_{\varphi_{\max}}$.

if the duration of the last window (see Eq. (4.22)) is shorter than the minimum required time according to Eq. (4.19), the previous window size is adopted accordingly.

$$T_{\text{LastWindow}} = T - \left\lfloor \frac{T}{T_{\text{window}}} \right\rfloor \cdot T_{\text{window}} \tag{4.22}$$

If not denoted otherwise, a default window size of $T_{\text{window}} = 2\text{s}$ is used for the remainder of this thesis.

4.3.2 Window Overlap

The previous section discussed the effect of future eCMP adaptions on the induced discontinuities between the planned and instantaneous DCM. However, this effect likewise influences the eCMP optimization itself, if the motion plan changes in the future vicinity of the optimization window. This occurs, for example, in the context of shared autonomy (see Section 3.3.2). In the specific context of shared autonomy walking via a gaming controller, the respective motion plan typically consists of only two moves: the move currently being executed and the immediate next move. The duration of a single step is thereby usually shorter than the default window size of two seconds, such that the respective optimization window will always[3] be capped to the length of the total step time, e.g., $T_{\text{window}} = T_{\text{DS}} + T_{\text{SS}} = 0.2\text{s} + 0.7\text{s}$. Fig. 4.7 visualizes this scenario, where a gaming controller is used to command a straight walk. The individual steps are thereby appended consecutively to the motion plan. Each new optimization window thus always optimizes the last step in the motion plan. Naturally, the eCMP optimization can only consider motions that are already contained in the motion plan, while the boundary conditions on the DCM's reverse time dynamics imply that the CoM stops at the final move [2]. However, by the time the motion plan is extended by an additional step, this assumption is no longer valid, which subsequently results in incorrect assumptions on the system's CAM and, thus, incorrect eCMP adjustments in each optimization where the motion is continued instead of stopped.

This problem can be resolved by employing overlapping optimization windows. That way, the major part of the motion plan is optimized twice: once during the first optimization with the assumption that the current step will be the final (stopping) step, and once as soon as the next move in the motion plan is available. This procedure results in the behavior visualized in Fig. 4.8, where a 70% overlap between the individual windows enables a correct optimization of the eCMPs. The overlapping percentage $p_{\text{overlap}} \in 0\% \ldots 100\%$ is expressed with respect to the previous optimization window size. The overlapping duration is added to the desired window size for the total window duration:

$$T_{\text{windowTotal}} = T_{\text{prevWindowTotal}} \frac{p_{\text{overlap}}}{100\%} + T_{\text{window}} \,. \tag{4.23}$$

This is generally possible, as the overlapping portion does not require any new phases/samples to be added to the plan, such that (4.20) always holds. However, there exists

[3] Ignoring the initial Stand-To-Move and final Step-To-Stand motions [2].

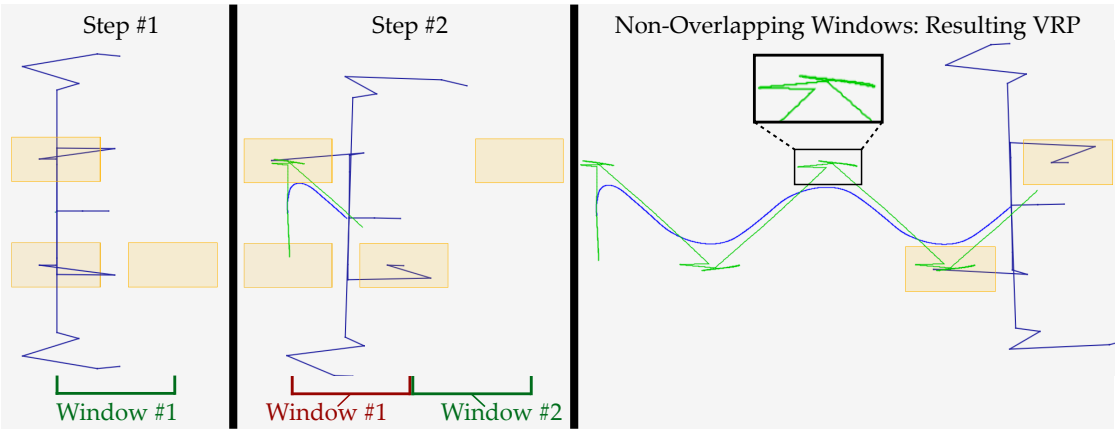


Figure 4.7: A straight walk commanded via gaming controller with a single step per optimization window. Non-overlapping optimization windows thereby result in partially incorrect eCMP adjustments.

a general upper limit for the overlap percentage. This is because the synchronous part of the motion planner has already started processing the previous window when the current window starts optimizing. As the optimization must be finished before the whole-body controller starts executing the respective part of the plan—ideally with a significant buffer T_{update} to reduce the induced DCM discontinuity—this upper limit is given per Eq. (4.24).

$$p_{\text{overlap}} < \frac{T_{\text{prevWindowTotal}} - \left(T_{\text{update}} + T_{\text{windowTotal}} \cdot N \cdot T_{\text{optimSample}}\right)}{T_{\text{prevWindowTotal}}} \cdot 100\% \tag{4.24}$$

While 0.5s is a reasonable value for T_{update} regarding the overall window length, this value is too conservative when talking about the maximum overlap percentage. The reason for this is that the second optimization run on the overlapping portion of the optimization typically has a significantly smaller effect on the overall VRP trajectory than the initial run. Also, it is reasonable to assume that the discontinuity induced by the overall plan extension is, in most cases, more significant than that induced by this second eCMP optimization. Justification for the latter is given by comparing Fig. 4.6 to the values provided in [2]. Assuming $T_{\text{prevWindowTotal}} = 0.9\text{s}$ for the first window[4], $T_{\text{windowTotal}} = 0.9\text{s}$, $N = 25$ and $T_{\text{optimSample}} = 0.5 \cdot 10^{-3}\text{s}$, one obtains the following range for a typical maximum overlap percentage in the first optimization window for $T_{\text{update}} = 0\text{s} \ldots 0.5\text{s}$:

$$41\% < p_{\text{overlapMax}} < 95\%\,. \tag{4.25}$$

In practice, $p_{\text{overlap}} = 50\%$ is chosen to achieve a reasonable large T_{update} even with fluctuating $T_{\text{optimSample}}$ for shared autonomy walking, while $p_{\text{overlap}} = 0\%$ for motion plans where the optimization window does not reach the end of the motion plan before it is extended.

[4]The first overall optimization window can not overlap with any prior window.

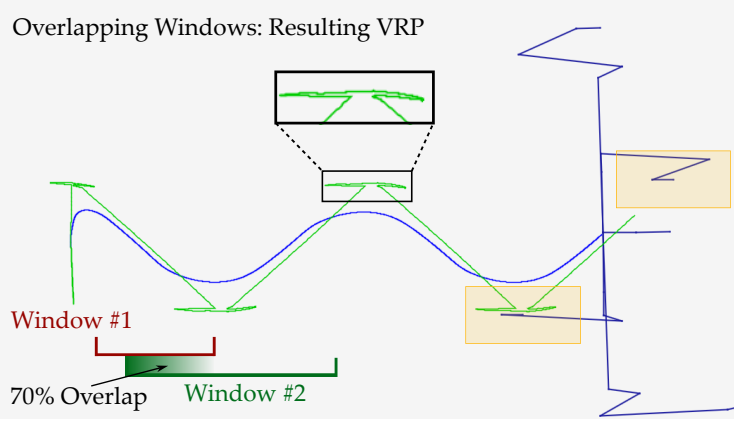


Figure 4.8: The same scenario as in Fig. 4.7, but with a 70% overlap between individual optimization windows. The resulting eCMP adaptions are thereby correct.

4.3.3 Sampling Rate

Setting the sampling rate for the optimization poses a trade-off between the maximum optimization window size and the accuracy obtained from the eCMP optimization. Discussions on the individual optimization times per sample are thereby conducted within Section 4.2. The number of samples required to accurately approximate the theoretical, continuous, non-linear eCMP trajectory required to cancel the CAM term in Eq. (4.10) is reasoned upon in this section. It is well known that—using equidistant samples—the error between the underlying trajectory and the trajectory resulting from linearly interpolating samples from that trajectory decreases quadratically with the number of samples [30]. The magnitude of the actual error thereby depends on the second derivative of the underlying trajectory via Rolle's theorem [30]. Intuitively speaking, the "curvier" the underlying trajectory, the larger the error obtained through an approximation via linearly interpolated samples.

Fig. 4.9 showcases the qualitative effect of different sampling rates on the example of a straight walk after optimizing the eCMPs. In this case, a rate of ten samples per second already provides a good approximation of the ideal eCMP trajectory. Little to no difference is visible when comparing the resulting trajectories at 25 or 50 samples. Contrarily, Fig. 4.10 visualizes the same sampling rates applied to a more complex movement: a slightly curved diagonal walk. In this case, the resulting eCMP adaptations follow a stronger curvature. At a rate of ten samples per second, this subsequently leads to significant deviations from the ideal trajectory. However, at 25 and 50 samples per second, there is, once again, little visual difference between the resulting trajectories. The CAM trajectories of both examples are visualized in Fig. 4.11 and Fig. 4.12, respectively.

Optimization window sizes that would require the use of only ten samples per second exhibit no particular benefit with respect to the induced DCM deviation, as opposed to, e.g., an optimization window of two seconds at 25 samples, either (see Fig. 4.6). Regarding the real-world tracking deviations shown in, e.g., Section 6.3.1, the gained

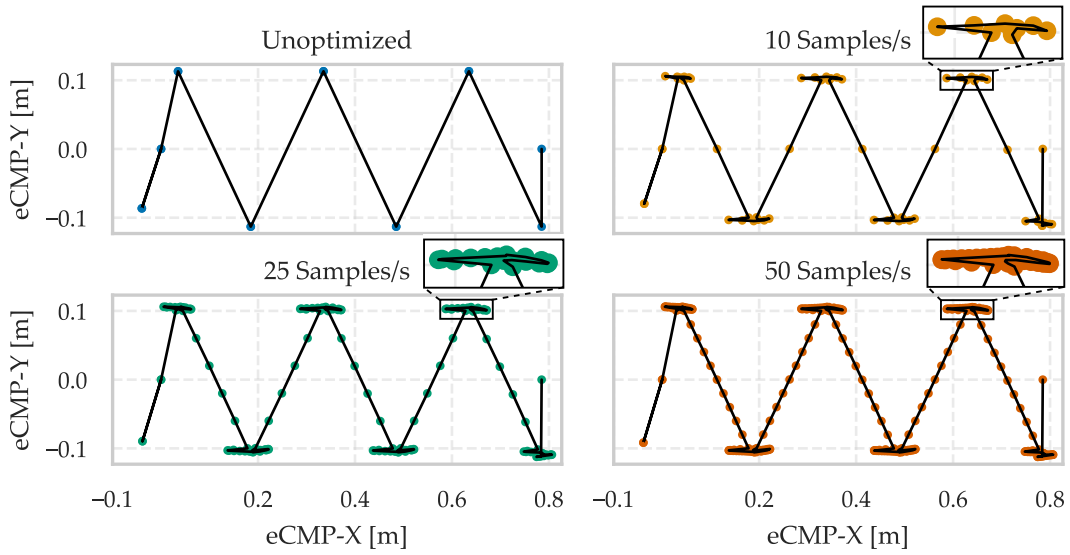


Figure 4.9: Different sampling rates for a straight walk after 2 optimization iterations. Parameters: Five steps at 15cm stepsize, single-support: 0.7s, double-support: 0.2s, Lookahead window: 5.7s.

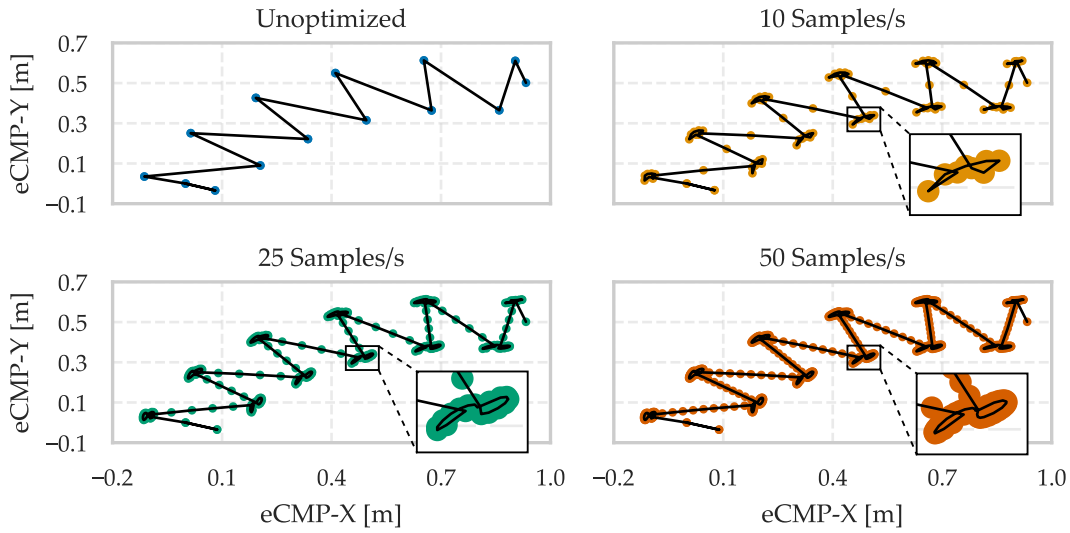


Figure 4.10: Different sampling rates for a curved diagonal walk after 2 optimization iterations. Parameters: Ten steps at X=15cm, Y=10cm, angle=−15°, single-support: 0.7s, double-support: 0.2s, Lookahead window: 10.2s.

accuracy through the use of 50 samples is typically not worth the added computational burden. Thus, rates of 25 ± 5 samples per second are reasonable for most applications, such that 25 samples is used as the default value for the remainder of this thesis.

4.3.4 Iteration Count and Convergence

The adaptation of the eCMPs with respect to the exerted CAM of the system subsequently induces a change to the CoM, and thus to the CAM. These coupled dynamics between the VRP, the CoM, and the CAM make it non-trivial to find a general proof of convergence for the algorithm proposed in [10]. However, empirical evidence suggests that the adapted eCMPs—and thus the CAM trajectory—always converge to a static configuration. Fig. 4.11 presents the respective evidence by the example of a straight walk. The sampling rate was thereby chosen at 25 samples per second, while the optimization window size was set to include the entire trajectory[5]. The system's CAM thereby converges to a stable trajectory on each axis; qualitative differences are only noticeable up to two iterations. A quantitative representation of the convergence is obtained by looking at the respective adjustments of the individual eCMPs. There, the median relative adjustment of a single eCMP with respect to the previous iteration becomes less than one millimeter after two iterations. The qualitative representation of the scattered eCMP supports this observation.

The second scenario for reference is shown in Fig. 4.12, where again the progress of the CAM trajector is analyzed along the relative eCMP adaptions between each iteration, but for the motion plan of a curved diagonal walk. Compared to the trajectory optimized in Fig. 4.11, the motion plan of the curved diagonal walk in Fig. 4.12 is nearly twice as long at 10.2 seconds and exhibits an overall less uniform motion of the eCMP trajectory. Still, just like in the example for straight-walk, the median relative eCMP adjustment converges after only two iterations to less than one millimeter.

Given the provided evidence, it is reasonable to conclude that the motion type (e.g., straight vs curved diagonal walk) and the overall length of the optimized trajectory has no substantial influence on the *rate*[6] of convergence. Considering the magnitude of the real-world tracking deviations shown in Section 6.3.1, deviations in the eCMP position of one millimeter or less can be considered negligible. Thus, a reasonable default value of 2 iterations is used for the remainder of this thesis.

[5] Lookahead windows of this size are generally only applicable to offline simulations as previously discussed in Section 4.3. In the provided scenario, this setting enables a comparison independent of the optimization window size and overlap.

[6] The total optimization *time* nonetheless varies with the length of the trajectory as discussed in Section 4.3.1.

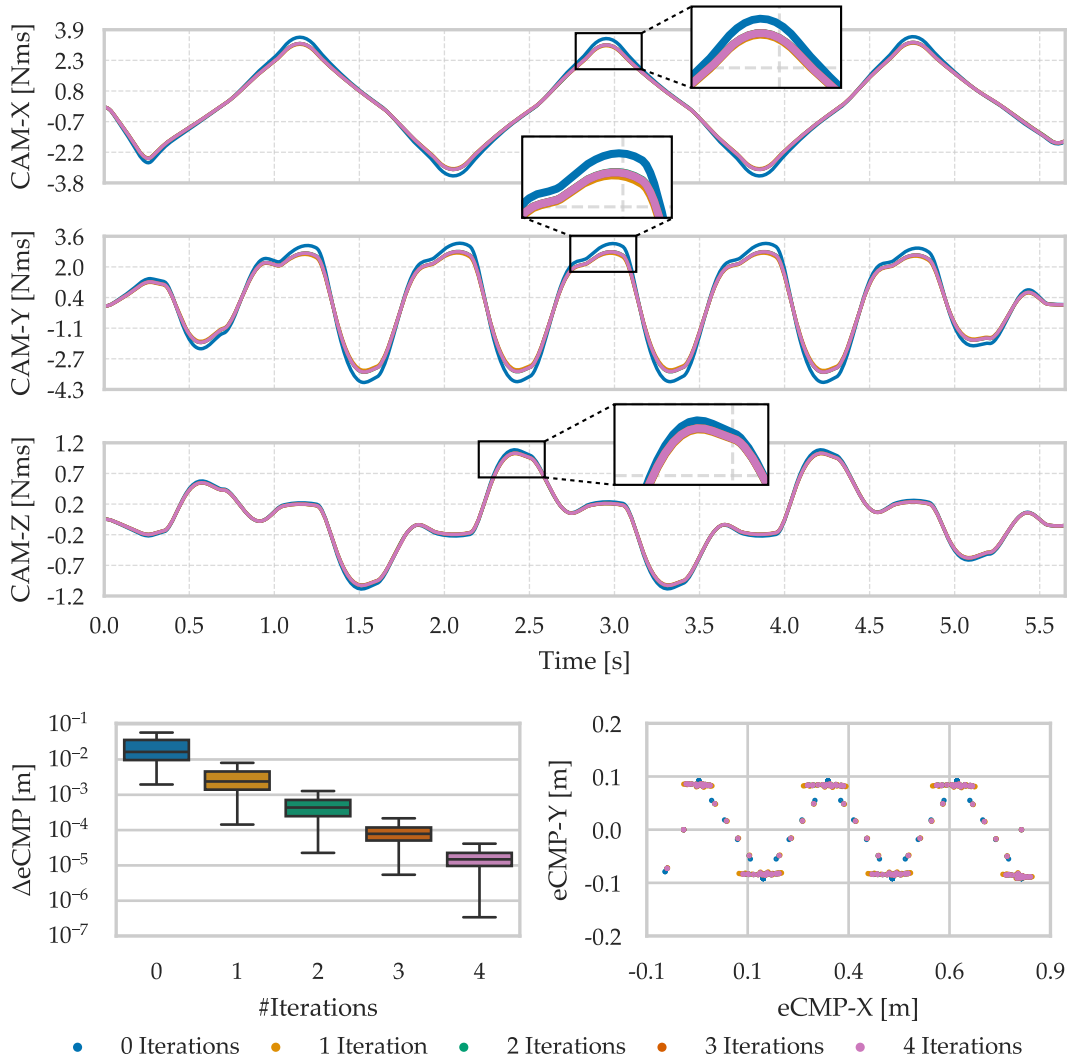


Figure 4.11: The CAM (simulated at 1kHz) in a straight walk after $0, \ldots, 4$ iterations of the eCMP optimization algorithm. The CAM trajectory quickly converges towards a stable solution. The median relative adjustments to the eCMPs become smaller than one millimeter after only two iterations. Parameters: Five Steps at 15cm stepsize, single-support: 0.7s, double-support: 0.2s, 25 Samples/s, Lookahead window: 5.7s.

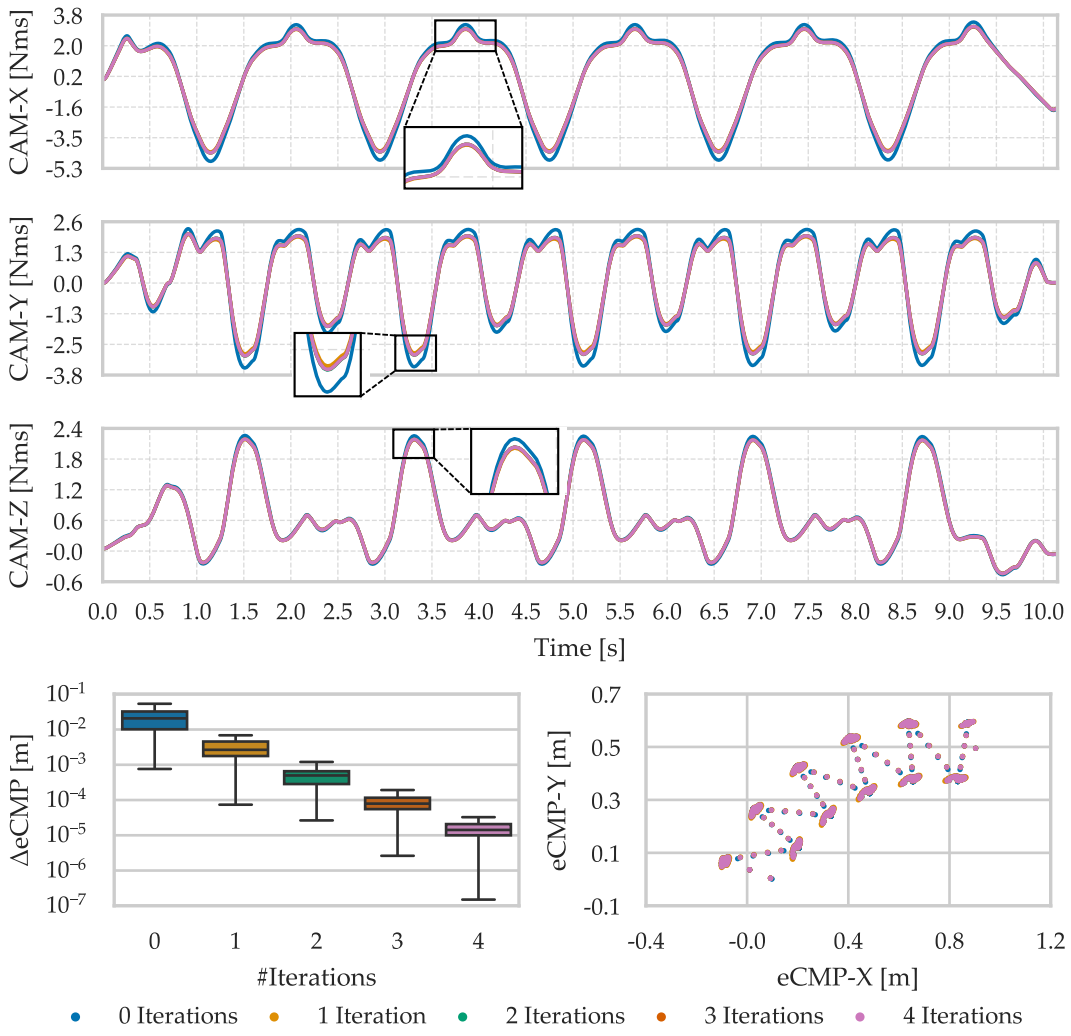


Figure 4.12: The CAM (simulated at 1kHz) in a curved diagonal walk after $0, \ldots, 4$ iterations of the eCMP optimization algorithm. The CAM trajectory quickly converges towards a stable solution. The median relative adjustments to the eCMPs become smaller than one millimeter after only two iterations. Parameters: Ten steps at X=15cm, Y=10cm, angle=$-15°$, single-support: 0.7s, double-support: 0.2s, 25 Samples/s, Lookahead window: 10.2s.

Chapter 5

Short-Term Contact Path-Planner

While the step adjustment [20] in the synchronous part of the 3D-DCM motion planner in [2, 4] (see Fig. 4.1) already enables a minimal form of replanning—triggered through an external event—it is intended to re-plan individual steps independently of the robot's surroundings. As shown in [4], stopping based on virtual boundaries is another already existing feature within the current motion planning framework. Inspired by the just-mentioned existing functionalities and the ongoing research in the field of (long-term) step and path-planning [31–37], this chapter conceptually proposes to extend the existing motion planner by a contact path planner to enable short-term replanning of the robot's movement based on its surroundings.

5.1 System Overview

The overall contact path planner consists of two separate modules: the *vision* module and the *path search* module, as depicted in Fig. 5.1. The vision module is assumed to obtain and maintain an internal representation of the robot's surroundings, e.g., by using an RGB-D camera, whose data is consistently merged into a (probabilistic) occupancy map [38–40]. The motion planner thereby communicates any planned contacts and their respective world coordinates with the vision module. If the vision module detects

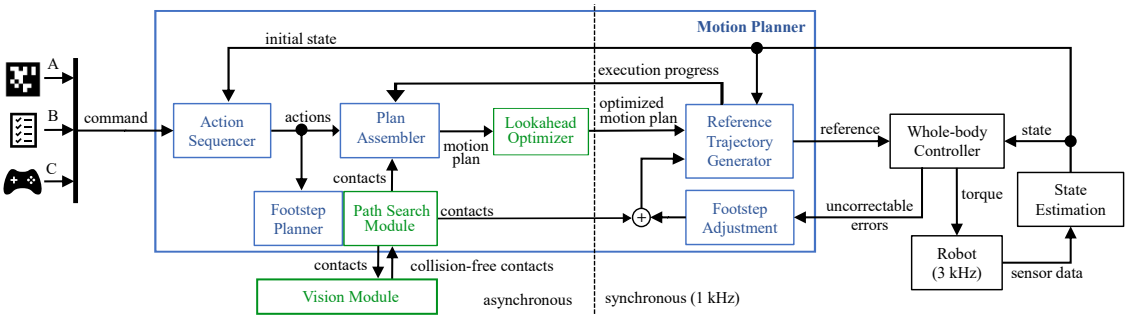


Figure 5.1: Extension of the motion planner architecture presented in [4] to include a vision and path search module along the Lookahead Optimizer (highlighted in green).

a potential collision between the current path and the environment, it sends a set of feasible alternative contacts to the path search module, thereby triggering the replanning procedure.

5.2 Vision Module

The development of a fully functional vision module for Simultaneous Localization And Mapping (SLAM) and collision detection is out of scope for this thesis. Therefore, an existing point cloud from a 3D scan is assumed to represent the robot's surroundings. The information on whether a path is colliding is, in this case, predetermined and hardcoded. In practice, a (hierarchical) occupancy map can be used in combination with a convex hull containing all of the robot's possible link positions for a given stance to distinguish between colliding and non-colliding path contacts. The further functionality of the vision module can be dissected into two different cases: colliding and non-colliding contact placements.

5.2.1 Vision Feedback for Non-Colliding Contacts

If the provided contacts are not colliding, the vision module performs a minor alignment of the contacts with respect to the actual surroundings of the robot. If, for example, the robot is walking on a slope, this information is quite valuable to the motion planner as it reduces any subsequent deviations between the commanded and the actual foot contacts. Although by the method proposed in [3], dynamic walking on uneven terrain is already possible, the motions appear much smoother if the respective terrain can be considered in advance,e.g., by reducing stomping of the heels during walking upwards a slope. This is achieved by extracting a Region Of Interest (ROI) around each contact from the point cloud, yielding a number of $n \in \mathbb{N}_+$ 3D points $\boldsymbol{p}_i$, which can be represented using a single matrix $\boldsymbol{R} \in \mathbb{R}^{3\times n}$. The points are mean-centered to obtain $\overline{\boldsymbol{R}}$ according to Eq. (5.1).

$$\begin{aligned} \overline{\boldsymbol{p}}_{\mathrm{ROI}} &= \frac{1}{n}\sum_{i=0}^{n-1} \boldsymbol{p}_i \\ \overline{\boldsymbol{R}} &= \boldsymbol{R} - \begin{bmatrix} \overline{\boldsymbol{p}}_{\mathrm{ROI}} & \cdots & \overline{\boldsymbol{p}}_{\mathrm{ROI}} \end{bmatrix}_{\in \mathbb{R}^{(3\times n)}} \end{aligned} \tag{5.1}$$

The objective now is to find a normal vector $\boldsymbol{n} \in \mathbb{R}^3$ describing a plane that minimizes the sum of squared distances to all points:

$$\min_{n} \sum_{i} \left(\boldsymbol{n}^T \boldsymbol{p}_i\right)^2 = \boldsymbol{n}^T \boldsymbol{R}\boldsymbol{R}^T \boldsymbol{n}\,. \tag{5.2}$$

A solution to (5.2) is obtained by computing the singular value decomposition of $\boldsymbol{R}$ [41, Chapter 12.1], where the Eigenvector corresponding to the smallest singular value corresponds to the best estimated normal:

$$\overline{\boldsymbol{R}} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^T \quad \text{s.t.} \quad \boldsymbol{n} = \boldsymbol{U}_{:,3}\,. \tag{5.3}$$

Projecting the initial position of the contact onto the plane defined by $(\overline{\boldsymbol{p}}_{\text{ROI}}, \boldsymbol{n})$ yields the new, terrain-adapted, reference position of the contact. The contact's orientation is thereby computed from the fitted plane's normal $\boldsymbol{n}$. The adapted contacts are then communicated back to the motion planner, along with the information that the provided contacts are non-colliding and replanning is not necessary.

5.2.2 Vision Feedback for Colliding Contacts

Given that the currently planned footsteps would cause a collision between the robot and its environment upon execution, the vision module is required to supply the motion planner, and thus the path search module, with a set of alternative contacts, along with a notification that the originally planned path is colliding. Obtaining a sensible, finite set of alternative contacts from the infinite set of possible positions and orientations of such contacts is generally non-trivial. One option is to choose a regular grid as a sampling strategy, while ignoring contacts that lead to potential collisions. Potential alternatives include sampling strategies based on hierarchical, probabilistic, and free-space representations that integrate point clouds into a multi-resolution occupancy map [38]. A concrete solution to this problem, however, is part of potential future work.

5.3 Path Search Module

Upon receiving the notice that replanning is required, along with a set of alternative contacts from the vision module, the path search module conducts an informed graph search to obtain a feasible motion plan from the last non-colliding stance to the original planned goal stance. A stance σ thereby resembles a set of contacts[1], each associated with a single limb of the robot. The objective of the path search is to find a sequence of stances $\left[\sigma_{start}, \ldots, \sigma_{goal}\right]$ that is feasible for the robot to traverse. The transition times between individual stances can be determined using the method proposed in [26].

The path search problem is represented as a graph structure, where each node represents a single stance σ_i. Edges between the nodes can be interpreted as the transition of a single limb to a different contact, resulting in a different stance σ_m. In order to avoid circles within the graph, each edge is thereby considered unidirectional, while each node—and thus, each stance—is unique. The last, non-colliding stance in the motion plan thereby resembles the start node. Let in the following $\#N \in \mathbb{N}_+$ resemble the number of a robot's limbs[2], while $\#c_0 \in \mathbb{N}_+$ resembles the number of alternative, non-colliding contacts provided by the vision module. Each contact is assumed to be associated with a single stance in each possible path, such that the number of available contacts for a node after $i \in \mathbb{N}_+$ transitions is $\#c_i = \#c_{i-1} - 1$. Each node thereby has up to $\#N \cdot \#c_i$ child nodes. From the start node onward, the graph is traversed via a

[1] Collision-free contacts received from the vision module, including the last non-colliding contacts in the motion plan.

[2] For the proof-of-concept, bipedal walking is assumed such that $\#N = 2$.

best-first search algorithm until the initial goal stance of the motion plan, i.e., the goal node is reached. It must thereby be ensured that only *feasible* edges are followed, i.e., only feasible nodes are expanded.

A node is considered feasible if the robot is kinematically and dynamically capable of transitioning between the stance defined through its goal node and the stance defined by the respective node itself. A hierarchical filter is thereby applied to sort out unfeasible child nodes: first, all contacts outside a conservative overestimate of the robot's reachable region are disregarded. For each of the remaining nodes, a 3D-DCM motion plan is constructed, which, in the case of bipedal walking, consists of two VRP waypoins, placed above the respective centroids of the stances σ_i and σ_{i-1}, as well as the foot trajectory of the transitioning limb. For multi-contact scenarios, a more elaborate placement strategy for the VRP waypoints must be employed, e.g., as proposed in [28]. The IK of the robot is then evaluated at an adjustable sampling rate along this plan. The dynamic feasibility can be assured by solving the wrench-distribution problem [26] along the respective motion plan. It is worth noting that a comprehensive dynamic feasibility check, along with potential VRP optimization for multi-contact applications, renders this filter operation computationally expensive and, using currently known methods, makes it unfeasible for use in a real-time capable path search algorithm.

Algorithm 3 showcases how the feasibility checks and node expansion can be included in a best-first graph search algorithm that utilizes a priority queue. The priority of a node thereby resembles a heuristic estimate on how likely a selection of the respective node leads to the goal stance.

Algorithm 3 Best-First Graph Search for Contact Path-Planning

Input: Goal stance *goalStance*, list of contacts *contacts*
Output: Path of stance nodes leading to *goalStance*, or `null` if no path exists

1: Initialize an empty priority queue Q
2: Enqueue the root node *stance* into Q with priority ComputePriority(*stance*)
3: **while** Q is not empty **do**
4: *currentNode* ← Q.Dequeue(()) ▷ Node with highest priority
5: **if** IsGoalReached(*currentNode*, *goalStance*) **then**
6: **return** Backtrack(*currentNode*)
7: **end if**
8: *unusedContacts* ← contacts not used in *currentNode.stance*
9: *children* ← CreateFeasibleChildNodes(*currentNode*, *unusedContacts*)
10: **for all** *child* in *children* **do**
11: *priority* ← ComputePriority(*child*)
12: Q.Enqueue(*child*, *priority*)
13: **end for**
14: **end while**
15: **return** null ▷ No path found

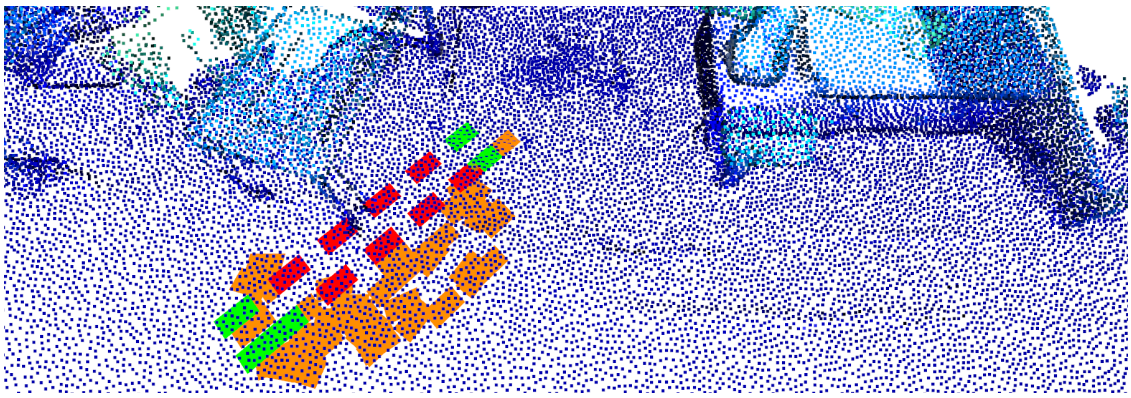


Figure 5.2: The vision module recognizes a set of contacts (red) that lead to collisions with the environment, and proposes a set of alternative, non-colliding contacts (orange) to the motion planner.

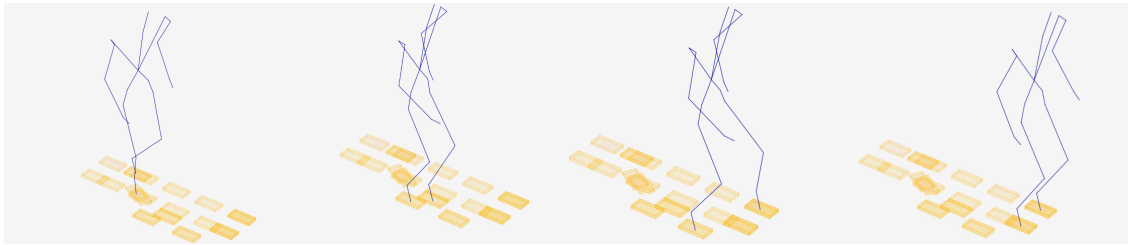


Figure 5.3: Visualization of the path search module's replanned motion.

5.4 Example Scenario

This thesis focuses on a proof-of-concept scenario for the proposed short-term contact-path planner's concept. For the sake of showcasing the path planner's intended functionality, the actions of the vision model are thereby manually predetermined and hard-coded. An offline prepared 3D scan is assumed to represent the robot's surroundings. Further, the feasibility of individual nodes is only verified with regard to the stance's kinematic reachability within the path search module.

The proposed scenario assumes that a bipedal straight walk is hindered by a potential collision with an obstacle—more precisely, a table. Fig. 5.2 illustrates the concrete setup in the vision module: the robot's desired path (green) collides with the table for parts of its trajectory. The "colliding" contacts are thereby highlighted in red. In response to this motion plan, which the vision module received from the motion planner prior to its execution, it proposes a set of alternative, non-colliding contacts (orange) to the path search module. The heuristic priority in the path search—for the proposed scenario—consists of the inverse distance between the limb's positions in the current stance and their respective positions in the goal stance, in addition to the weighted inverse of the number of all previous steps. The resulting motion plan is simulated via openHRP [42]

(see section 6.2). Fig. 5.3 visualizes the corresponding result of the simulation.

5.5 Discussion

The previous sections conceptually proposed extending the motion planner architecture with a path search module, which, in combination with an external vision module, is envisioned to enable real-time capable short-term replanning, e.g., for obstacle avoidance. A simplified implementation of the proposed graph search algorithm, which only accounts for kinematic reachability in its feasibility checks, was subsequently used in an example scenario to demonstrate a potential real-world application of the algorithm. However, to successfully deploy a similar algorithm in a real-world scenario, especially in the context of multi-contact locomotion, further investigations, e.g., towards a method for a more efficient validation of dynamic stance feasibility, are necessary. While the total computation time of the overall path search in the proposed scenario was approximately 100 ms, the set of alternative contacts was handcrafted and contained only 53 total contacts. Using a trivial regular grid for sampling alternative contacts from non-colliding regions would likely result in runtimes in the order of seconds without an additional tune in the heuristic priority computation. The terrain-adapted contacts, as proposed in section 5.2.1, however, already resemble a working part of the overall envisioned system—all with the assumption of a reasonably accurate SLAM implementation.

Chapter 6

Experiments and Evaluation

6.1 TORO: A System Overview

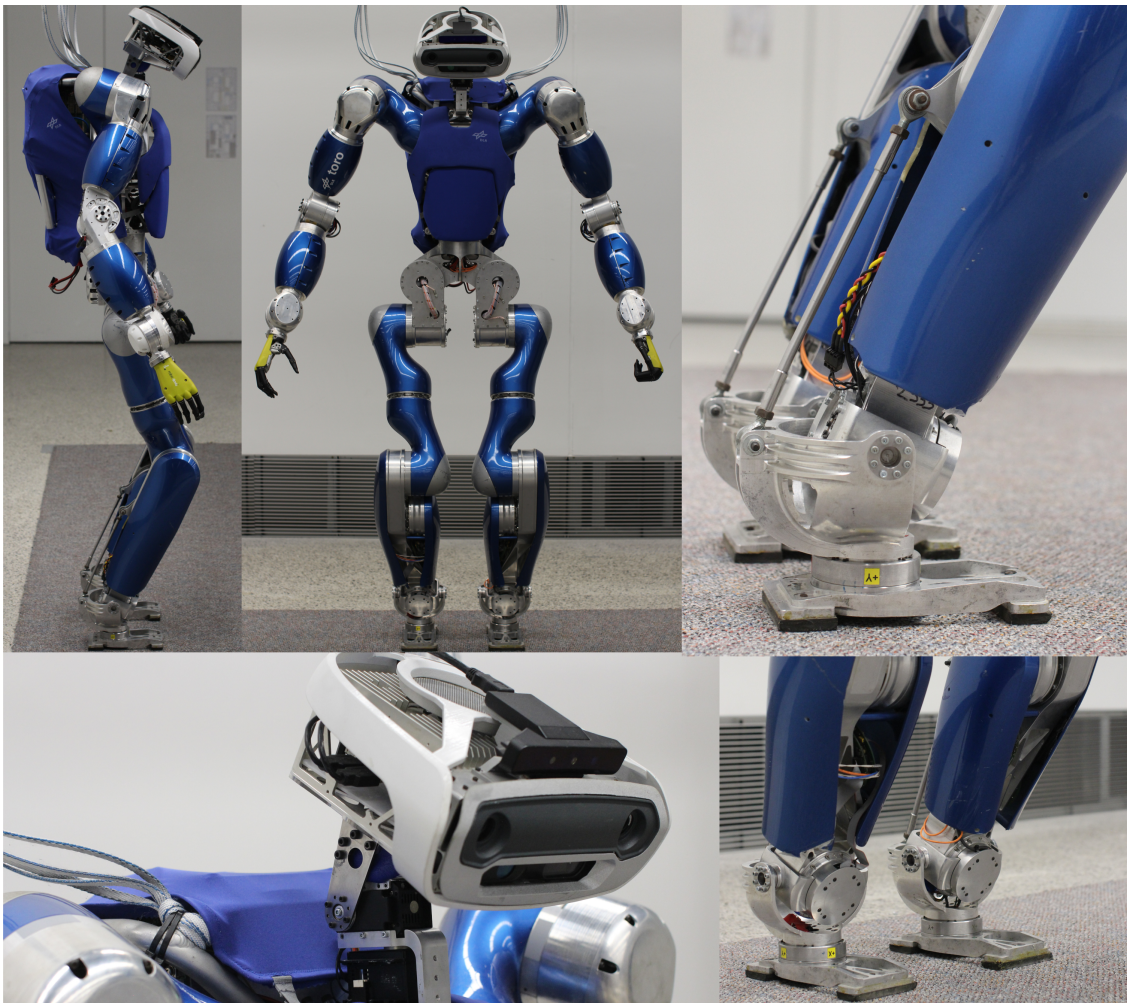


Figure 6.1: DLR's Torque-Controlled Humanoid Robot (TORO) is used for the experimental evaluation of the proposed algorithms.

TORO is a torque-controlled humanoid robot that was designed and built by the Robotics and Mechatronics Center of the DLR (see Fig. 6.1). It consists of a total of 27 joints—two of which are position-controlled and located in the robot's neck. The remaining 25 joints resemble variations of the torque-controlled Light Weight Robot (LWR) drive units [43], each of which contains a torque-sensor, position sensors, and a brake system. A detailed overview of TORO's kinematic chain is provided in Fig. 2.1. The system is powered by two battery packs, which last approximately 45-60 minutes in total. Except for a safety-crane system, TORO is able to move freely detached from its surroundings. Small gummy strips are attached to the footsoles to increase friction and absorb any potential (erroneous) impacts. Each of the footsoles is 19cm long and 9.5cm wide. The coordinate frames of the robot's feet are oriented as shown on the yellow stickers on the feet in Fig. 6.1. The $\boldsymbol{x}$ axis is thereby positive in the forward-facing direction of the robot, the positive $\boldsymbol{y}$ axis faces to the right (relative to the robot), and $\boldsymbol{z}$ is positive in the direction of the ground, facing away from the robot. For all of the following simulations and experiments, the standard parameters of 25 samples per second, two optimization iterations, and a lookahead window size of two seconds are chosen for the eCMP optimization algorithm (see Section 4.3). Diagrams on contact forces/torques or CoP deviations always represent the respective values of the right foot. The respective quantities on the left behave analogously.

6.2 Simulation

Simulations are essential to minimize the risks of damaging the system in actual experiments. Furthermore, they allow the isolation of the behavior of specific parts of the system—such as the motion planner—without the necessity of considering, for example, tracking deviations of the whole-body controller or any torque/velocity constraints exhibited by the hardware. For this thesis, the openHRP [42] simulation is chosen for this task due to its realistic contact model and overall accuracy in simulating the real-world behavior of the actual TORO system.

6.2.1 Straight Walking

This section considers a straight walk with eight steps, as shown before and after applying the eCMP optimization in Fig. 4.11. For the unoptimized baseline, Fig. 6.2 displays the quantities describing the robot's centroidal dynamics, alongside the exerted contact wrenches, when executing the unoptimized motion plan in the openHRP simulation. While the CAM about the z axis, and thus, per Eq. (4.10), subsequently the contact torques about the z direction, are regulated to zero by the whole-body controller, the contact torques about the remaining axes remain significant. Especially on the $\boldsymbol{y}$ axis, the torques nearly reach the contact constraints assumed by the whole-body controller [3]. Reciting (3.2), non-zero torques about the $\boldsymbol{y}$ axis, in this context, induce a CoP deviation along the $\boldsymbol{x}$ axis of the foot, i.e., in the walking direction. Torques about the $\boldsymbol{x}$ axis further

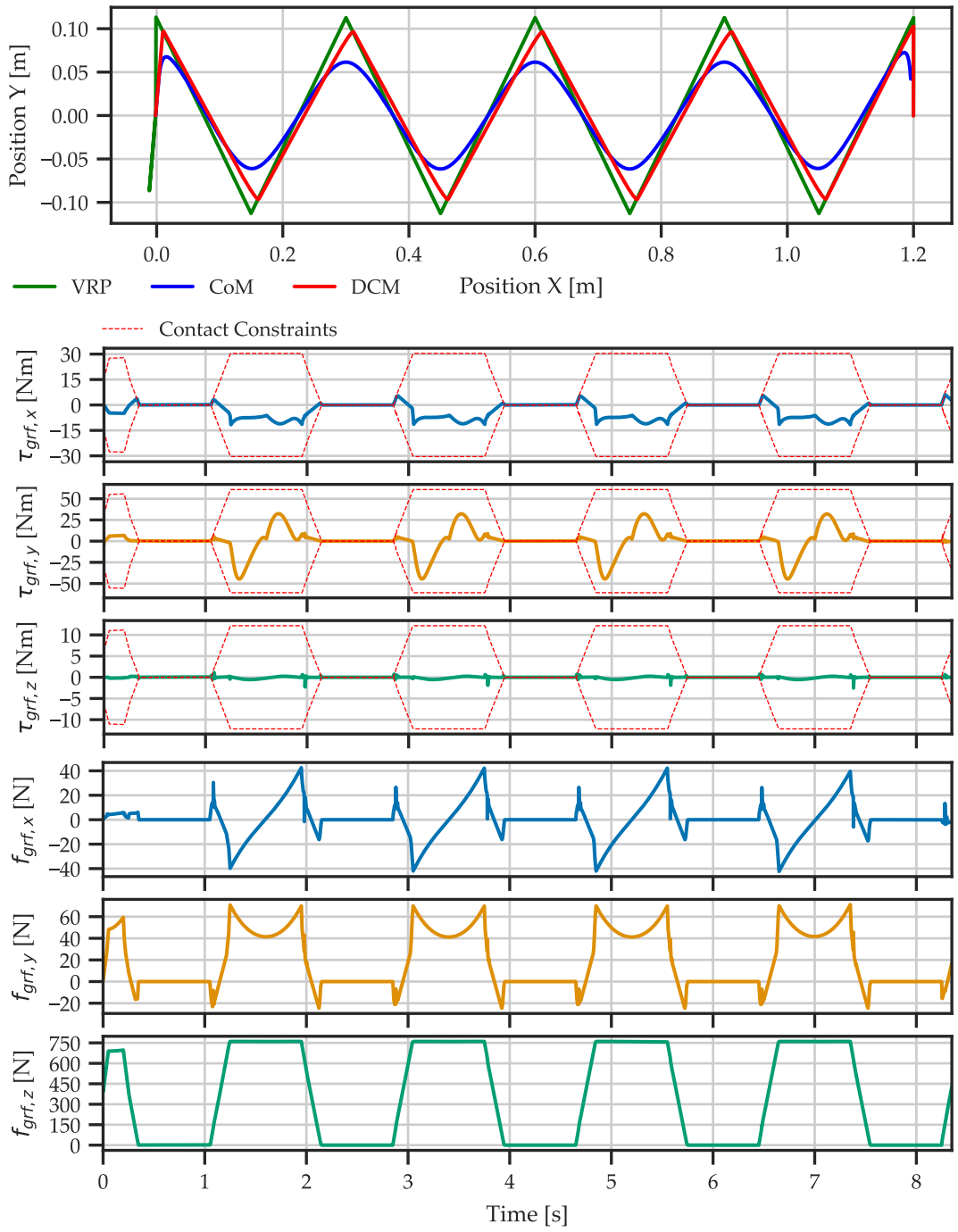


Figure 6.2: The planned CoM dynamics for a straight walk of eight steps without eCMP optimization along the exerted contact wrench at the right foot over time. As the CAM about the z direction is regulated to zero by the whole-body controller, the contact torque about z is also nearly zero. Parameters: step X=15cm, single-support: 0.7s, double-support: 0.2s.

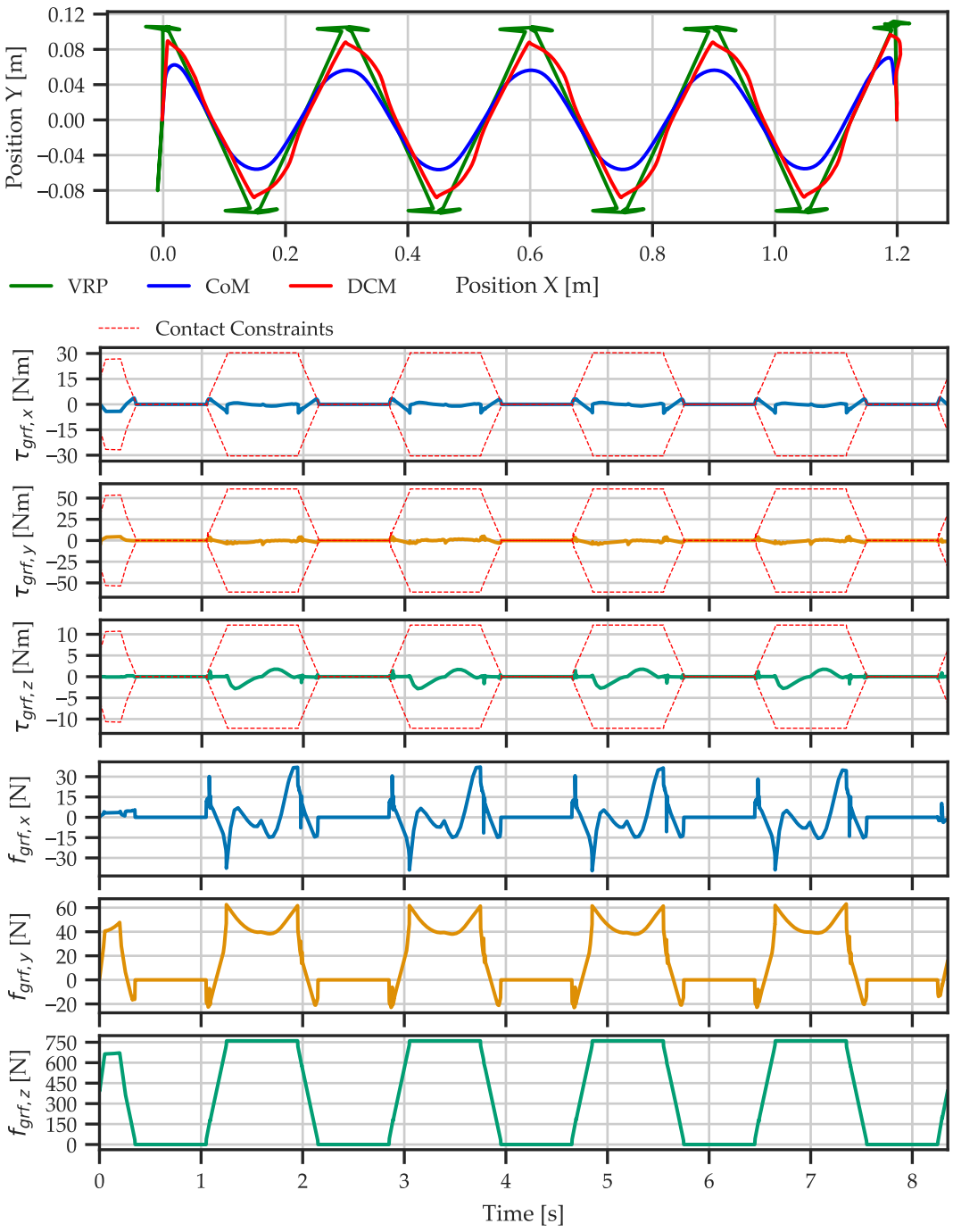


Figure 6.3: The planned CoM dynamics for a straight walk of eight steps with active eCMP optimization along the exerted contact wrench at the right foot over time. With respect to the baseline in Fig. 6.2, the contact torques about the x and y axes are successfully reduced, while the contact torque about the z axis increases as the CAM about this direction is regulated to zero by the whole-body controller. Parameters: step X=15cm, single-support: 0.7s, double-support: 0.2s.

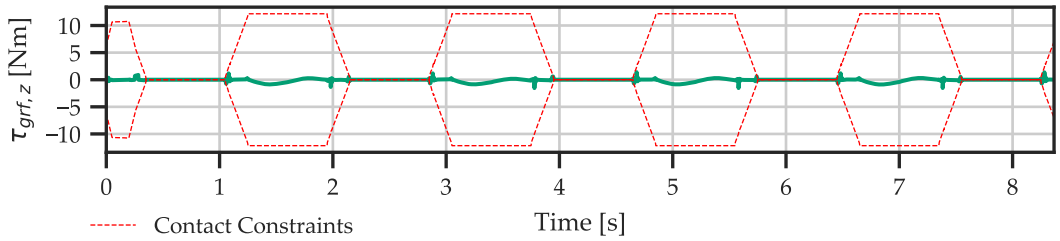


Figure 6.4: Contact torques on the z axis in the same scenario as in Fig. 6.3, but with an explicit tracking of the CAM-Z trajectory defined in Eq. (4.16).

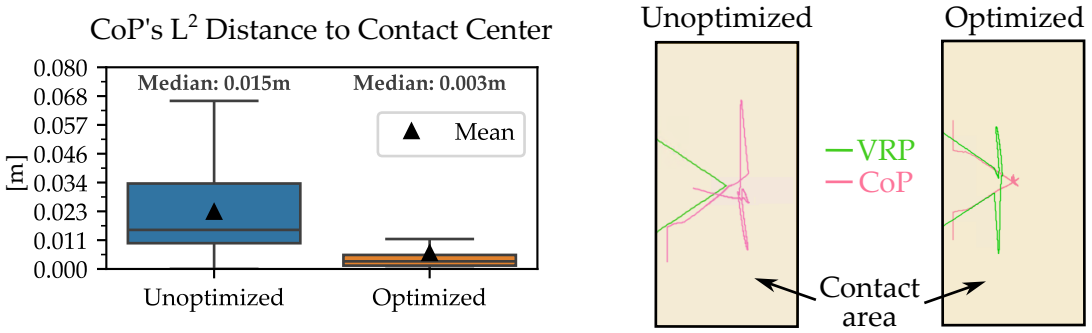


Figure 6.5: Comparison between the unoptimized motion plan's CoP deviations from the contact center to that of the eCMP optimized plan. The optimized plan exhibits a median improvement of 80%.

correspond to CoP deviations normal to the walking direction, within the contact plane. Contrarily, the torques about the x and y axes resulting from the optimized motion plan in Fig. 6.3 exhibit much smaller overall deflections. The main deviations from zero torque are exerted about the x axis and happen during the attachment and detachment phases of the contacts. During these transition phases, the robot remains in double support and shifts its weight away from the current point of contact. During this motion, the overall applied force is relatively small, resulting in a less rigid ground contact at the foot from which the weight is transferred away. The torque about the z axis, however, is increased with respect to the unoptimized plan. This torque is a direct result of the eCMP adjustments as denoted in Eq. (4.15). Setting the motion optimizer [5] to tracking the CAM's z reference trajectory obtained via Eq. (4.16) provides a remedy for this issue, as shown in Fig. 6.4.

Quantitative results on the improvement obtained through the eCMP optimization can be obtained via the overall deviation of the actual, simulated, CoP from the center of the contact (i.e., the desired CoP), as shown in Fig. 6.5 alongside the trajectory of the CoP and VRP within the actual foot's contact area.

6.3 Experiments

The proposed algorithm from the previous chapters is evaluated in a series of real-world scenarios on the experimental platform TORO [6]. The subsequent experiments thereby showcase a series of plots containing the contact forces and torques perceived by the system. As the force-torque sensors at the feet of the TORO system were not available for use during the experiments, the obtained values instead stem from the whole-body controller's perception of the system state. Nevertheless, the difference to an explicit sensor measurement can be considered small enough to be negligible in the context of showcasing the effectiveness of the sliding-window CAM-based eCMP optimization.

Several assumptions are made about the system executing the planned trajectories—in this case, the TORO system. Some of these assumptions are the overall structure of the robot's kinematic model, the inertia properties of individual links, friction within the joints, accurate sensor measurements of the individual joint torques, and many more. Simulations are, just like any motion planning or control algorithm itself, up to a certain point dependent on these assumptions. For example, the openHRP simulation employed within the previous section makes use of the same kinematic and dynamic model of TORO as employed for the computation of the CAM and the subsequent adaptation of the eCMPs. Thus, any model discrepancies in this context will only become apparent during real-world experiments, not during simulations. As a second example, the readings obtained from the torque sensors within the joints are, in the real world, temperature-dependent and require proper calibration. Consequently, the results in the subsequent sections are assumed to differ from those obtained in the previous section, which leveraged the computer simulation openHRP. Nonetheless, to conclude proper functionality of the eCMP optimization algorithm, the obtained values must exhibit a substantial improvement over the baseline performance obtained from an unoptimized motion plan.

6.3.1 Straight Walking

The motion parameters for the real-world experiment of straight walking are analogous to the simulation: eight steps are conducted with a step length of $X = 15$cm, single-support time of 0.7s, and double-support time of 0.2s. A baseline for the system performing the commanded motions without active eCMP optimization is shown in Fig. 6.7. This figure contains a comparison between the commanded and actual CoM dynamics, along with the exerted contact wrenches during the performed motions. It becomes apparent from Fig. 6.7 that the real-world system—without the eCMP optimization enabled—is much more frequently acting at the respective contact constraints. This not only limits the walking speeds and overall agility of the robot, but also makes walking less robust to external influences overall. In contrast to the contact torques obtained from simulation in Fig. 6.2, the real experiment shows significant contact torques being exerted about the z axis, even though the CAM's z portion is again commanded to be regulated to zero through the whole-body controller.

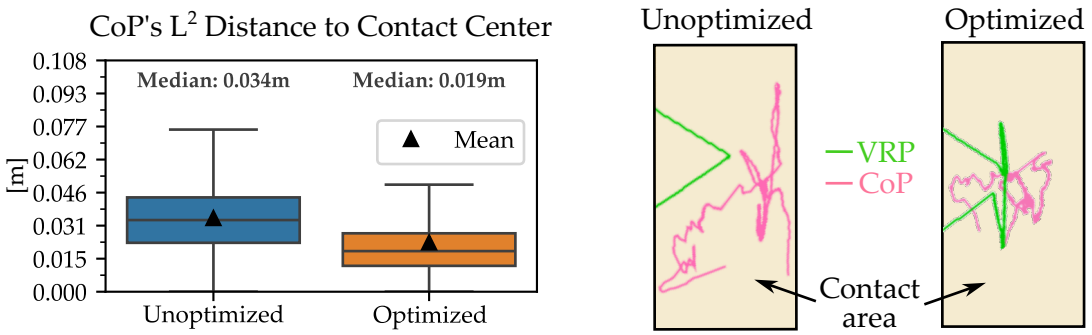


Figure 6.6: Comparison between the unoptimized motion plan's CoP deviations from the contact center to that of the eCMP optimized plan during the experiments on straight walking conducted with TORO. The optimized plan exhibits a median improvement of 44%.

Activating the eCMP optimization algorithm yields the quantities visualized in Fig. 6.8. Analogously to the results obtained in simulation, the contact torques are reduced—albeit less optimally—about both the x and y axes. In contrast to the behavior shown in Fig. 6.3, the contact torques about the z axis roughly remain the same whether the eCMP adaptation is active or not, even without explicitly tracking the respective CAM-Z trajectory in (4.16). The difference in the number of ground contacts, and hence in the total trajectory time between Fig. 6.8 and Fig. 6.7 results from the experimental setup: In Fig. 6.7, the motion plan starts with a right-foot stance, while the motion plan in Fig. 6.7 starts with a left-foot stance. The number of contact phases of the right leg with the ground is therefore five and four, respectively. This, however, does not impair the comparability of the two respective scenarios. The quantitative comparison of the CoP placements exhibited from both the unoptimized and optimized motion plan is shown in Fig. 6.6. Although the improvement is overall less notable than in the simulation, it is still indisputably significant at a median decrease of the CoP deviation by 44%.

6.3.2 CAM-Z Reference

In simulation, an explicit tracking of the CAM-Z trajectory in (4.16) severely decreased contact torques about the z axis that resulted from the optimized eCMPs (see Section 6.2.1). Without active eCMP optimization, the CAM-Z trajectory, and thus the resulting contact torques about the z axis, could thereby be successfully regulated to zero via the motion optimizer.

In the experiments on the real system, however, the exerted contact torques about the z axis—even for the baseline experiment in Fig. 6.7—were substantial. Interestingly, imposing eCMP adaptions to the respective motion plan without accounting for the altered CAM-Z trajectory according to (4.16) did not significantly increase the z axis contact torques either. In Fig. 6.9, the resulting z axis contact torques are shown for an eCMP optimized straight walk during the tracking of a zero-reference trajectory for

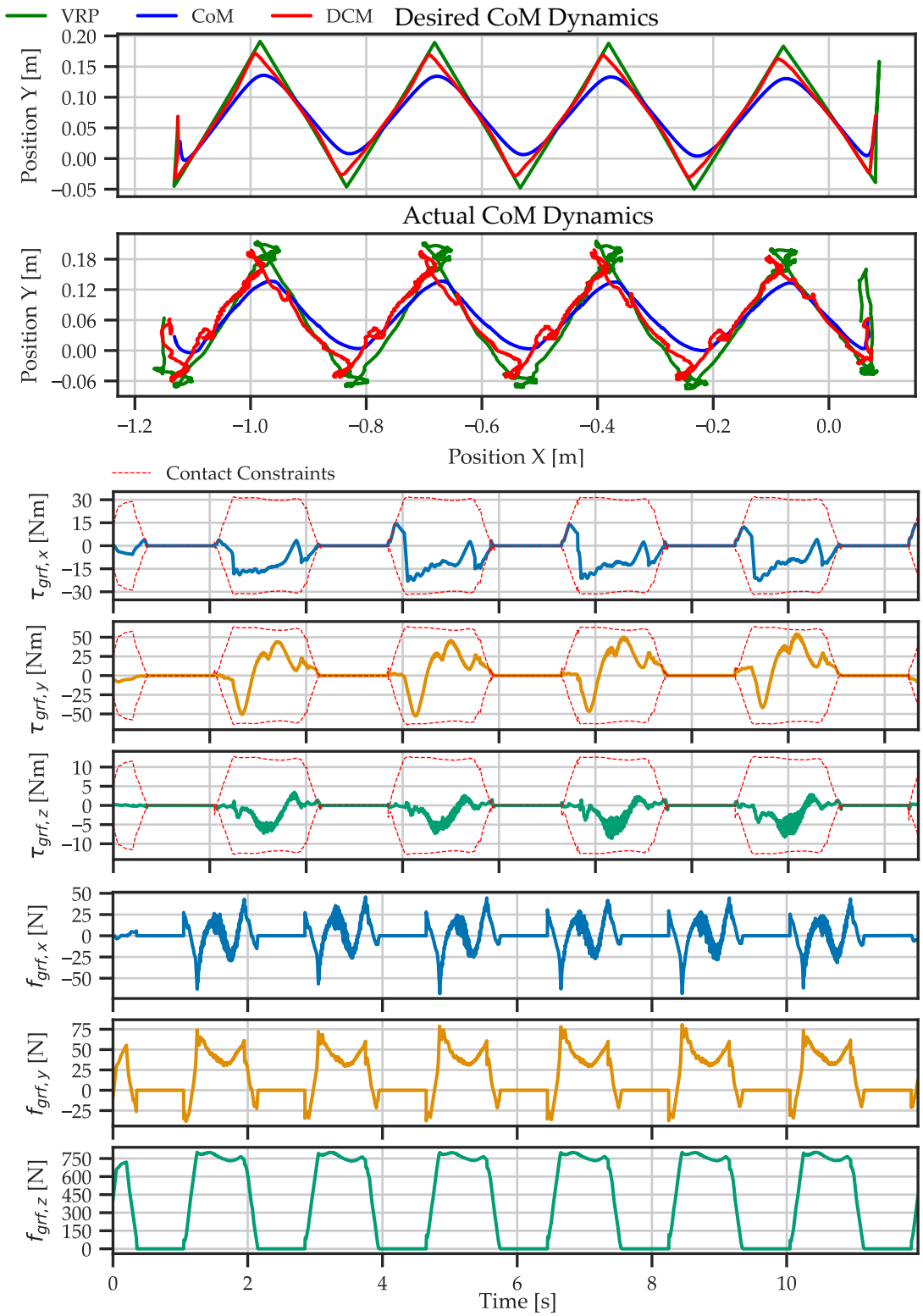


Figure 6.7: Straight walk experiment on TORO: Baseline CoM dynamics and contact wrenches.

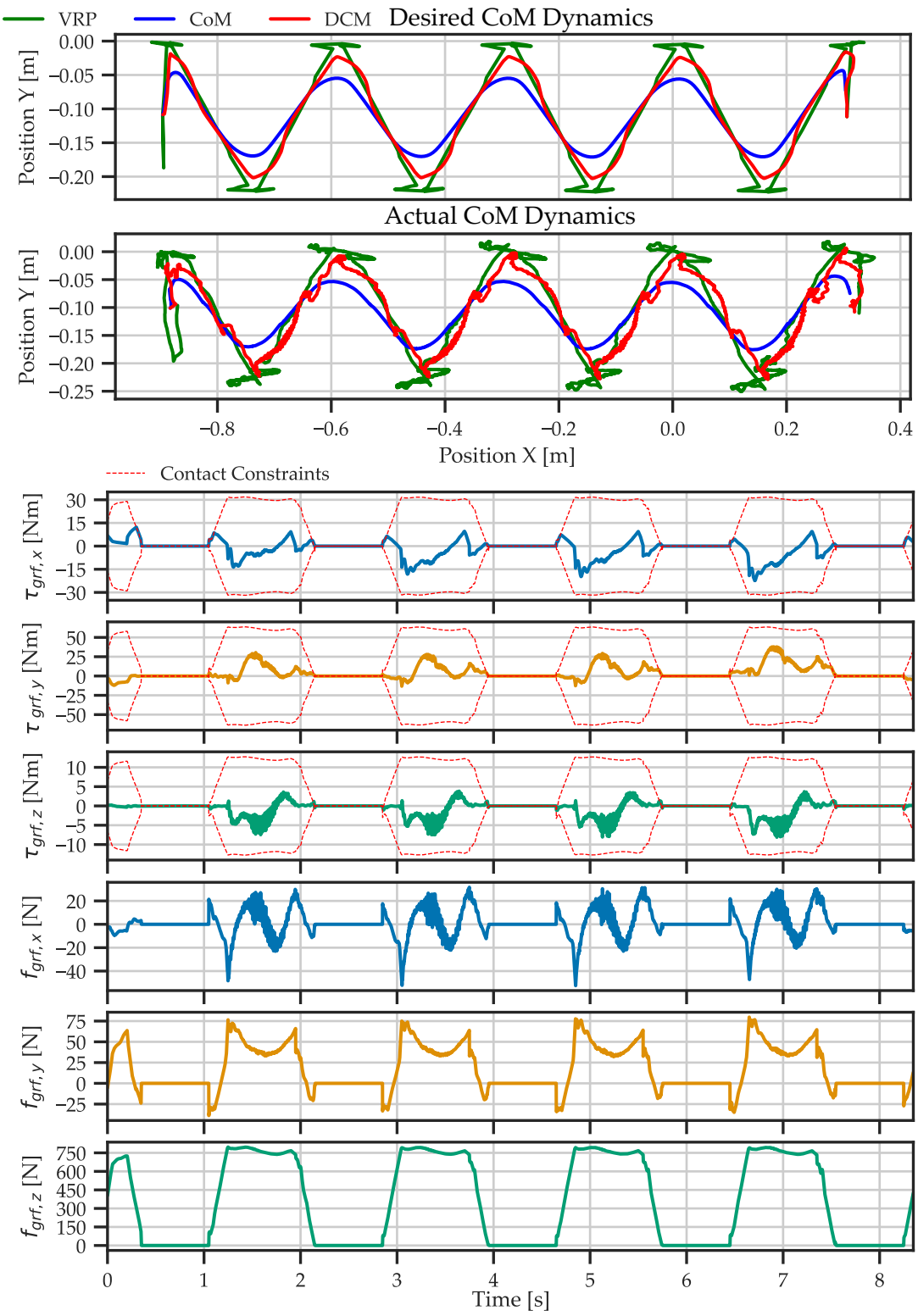
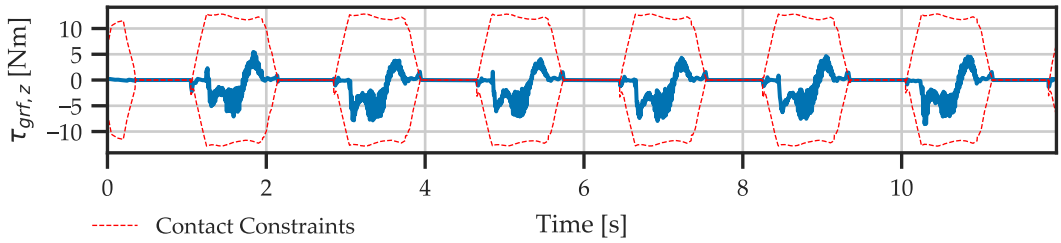
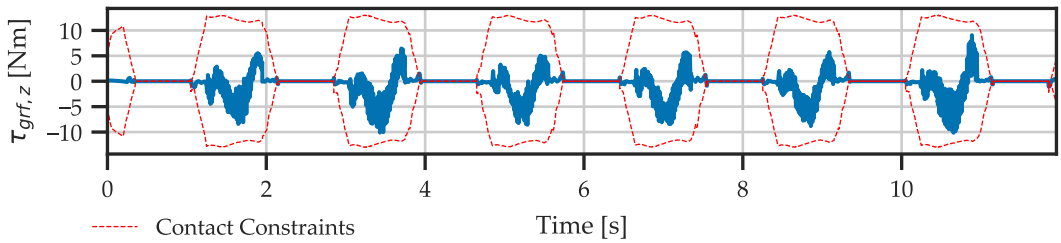


Figure 6.8: Straight walk experiment on TORO: CAM optimized CoM dynamics and contact wrenches.



(a) Contact torques about z using a zero reference trajectory for CAM-Z.



(b) Contact torques about z using the non-zero trajectory defined via (4.16) as reference trajectory for CAM-Z.

Figure 6.9: Contact torques about the z axis exerted by an eCMP optimized straight walk for different CAM-Z references.

the CAM's z component (Fig. 6.9a), versus tracking of the explicitly adapted trajectory resulting from (4.16). While in simulation, a substantial difference was observed when comparing these two scenarios; the non-zero CAM-Z trajectory for the real system only appears to increase the variance of the z axis contact torque. While one conceivable reason for this is given by the choice of gains in the motion optimizer in [5], further investigations are necessary on this behalf. Further, during any turning motion of the robot, the tracking of a desired CAM-Z trajectory is disabled by default for both this thesis as well as prior work. This is because, with the current implementation, the CAM about the z axis required for following a given motion plan can not be sufficiently dissected from the "undesired" Z-CAM induced, e.g., through swing leg motions. Solving this issue is part of the potential future work accompanying this thesis.

6.3.3 Shared Autonomy Walking

To experimentally evaluate CAM-based eCMP optimization during shared autonomy walking via a gaming controller, two individual experiments with the TORO system are conducted—one resembling the baseline performance of the system and one with eCMP optimization enabled. The motion parameters for both experiments consist of a maximal step-length of $X = Y = 15$cm, single-support time of 0.9s, and double-support time of 0.3s. The parameters of the eCMP optimization consist of 25 samples per second, with a

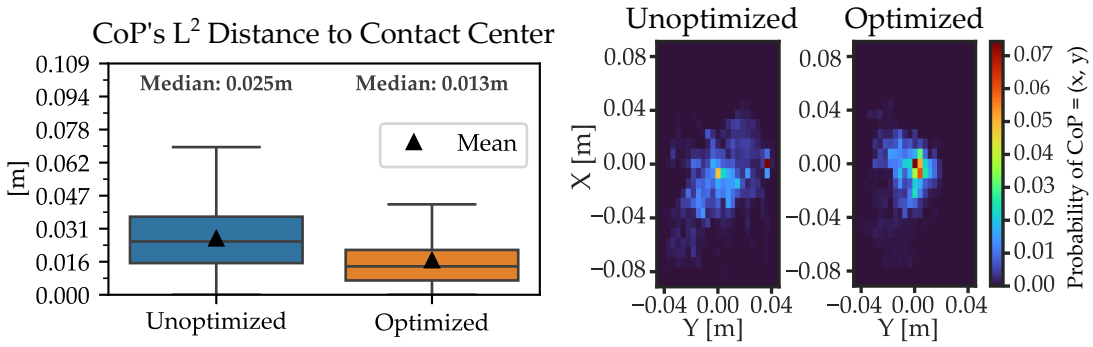


Figure 6.10: Comparison between the optimized and unoptimized shared autonomy walking in terms of the CoP deviations from the contact center (left). The right figure resembles a probabilistic representation of the CoP's position throughout the experiment.

total of two iterations, a window size of two seconds, and 50% overlap in between the optimization windows.

The contact torques in the right foot obtained from the baseline implementation are visualized in Fig. 6.11. Enabling the CAM-based eCMP optimization results in the quantities shown in Fig. 6.12. Both visualizations also contain a high-level overview of the motion, with only the desired VRP being plotted along the robot's actual poses in a top-down visualization. As the commanded motion plans are both commanded by a human via a gaming controller, the two distinct motions only roughly match. As the optimized motion plan is overall less uniform and contains sharper turns, one would generally expect higher overall contact torques than in the baseline experiment. Nevertheless, the experiment in Fig. 6.12 still shows a clear improvement over the baseline, and thereby further validates the effectiveness of the proposed algorithm. A quantitative comparison between the baseline and the optimized version, in terms of the CoP distances to the contact centers, is provided in Fig. 6.10. As the motion described by the shared autonomy walk, and thus the movement of the CoP within each contact, is non-repetitive throughout the motion plan, a probabilistic representation of the CoP's position within the contact is chosen over the CoP's movement in a single representative foot stance. The foot's contact is thereby divided into 25 by 25 *buckets*. The CoP is sampled and assigned to one of these buckets at the frequency used by the whole-body controller (1kHz). The number of samples within each bucket divided by the total number of samples then yields the probability of the CoP being located within each bucket, as visualized in Fig. 6.10. It can be concluded that the CoP in the optimized motion plan is much more likely to be located near the center of the contact than the CoP exerted by the reference plan.

For a second example of the eCMP optimization being used in a shared autonomy walk, the robot is commanded to turn in place by 180° and, directly after, should walk ten straight steps. The motion parameters are thereby set for an increase in

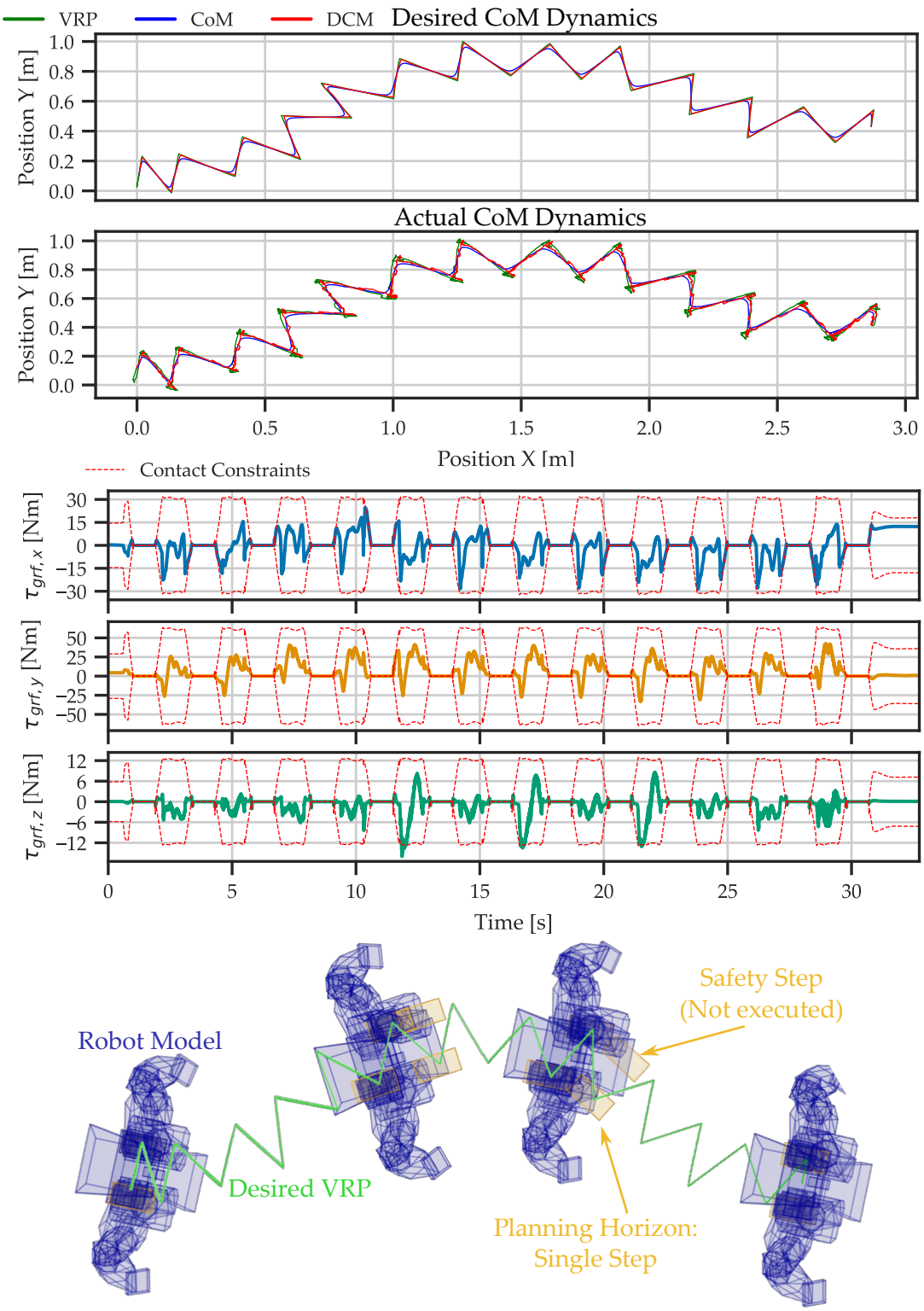


Figure 6.11: Baseline shared autonomy walking experiment on TORO: Desired and actual CoM dynamics, contact torques, and a high-level overview of the motion.

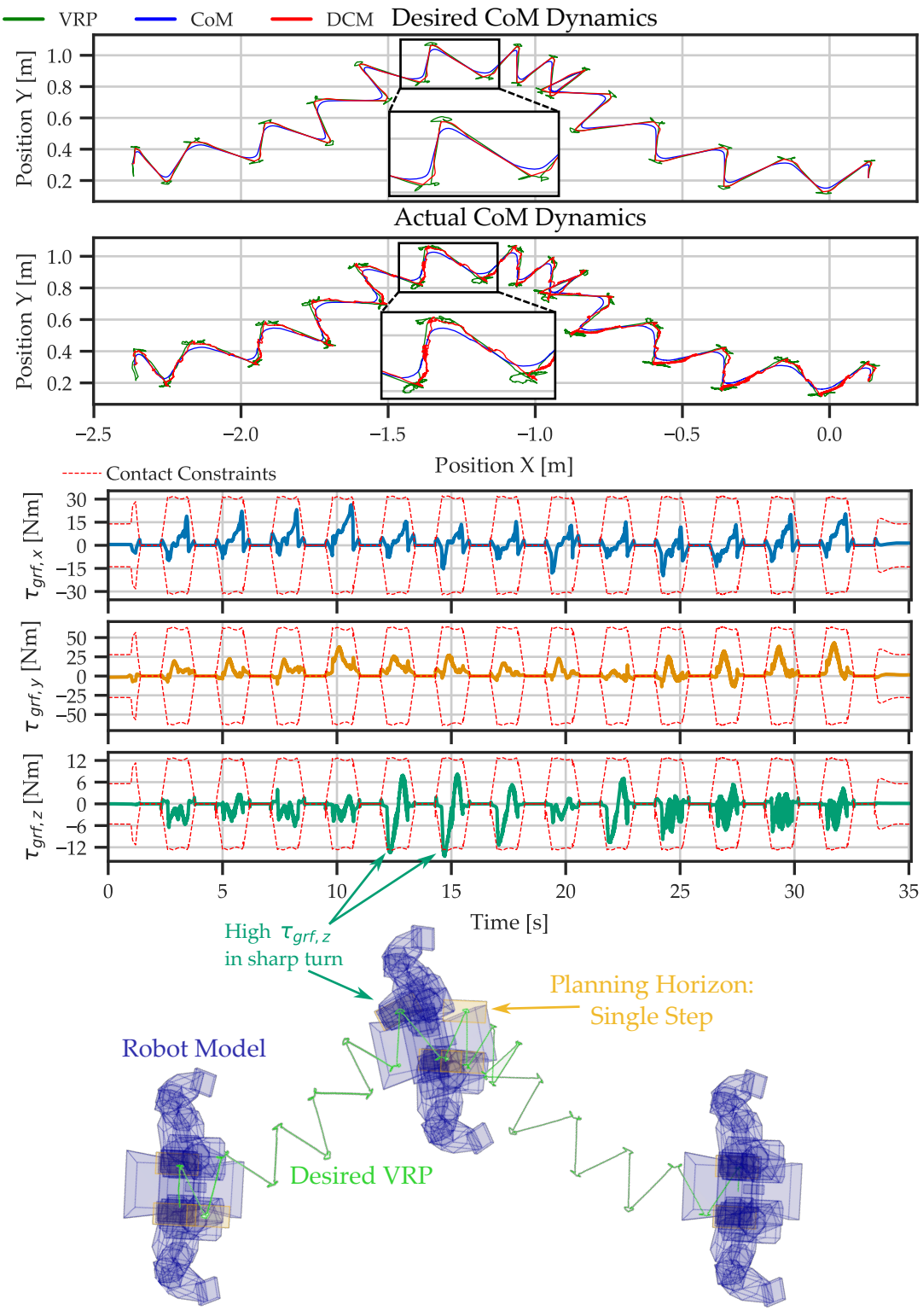


Figure 6.12: Optimized shared autonomy walking experiment on TORO: eCMP optimized desired and actual CoM dynamics, contact torques, and a high-level overview of the motion.

speed to a maximal step-length of $X = 20$cm, $Y = 15$cm, a single-support time of 0.7s, and a double-support time of 0.2s. The parameters of the eCMP optimization remain unchanged. The CoM dynamics with the resulting contact torques for this motion are depicted in Fig. 6.13. Note that the depicted contact torque about the z axis exceeds the assumed contact constraints when exiting the in-place turning motion to continue with the straight walk. The fact that the system continues normal operation after this movement implies a conservative assumption on the respective contact constraints.

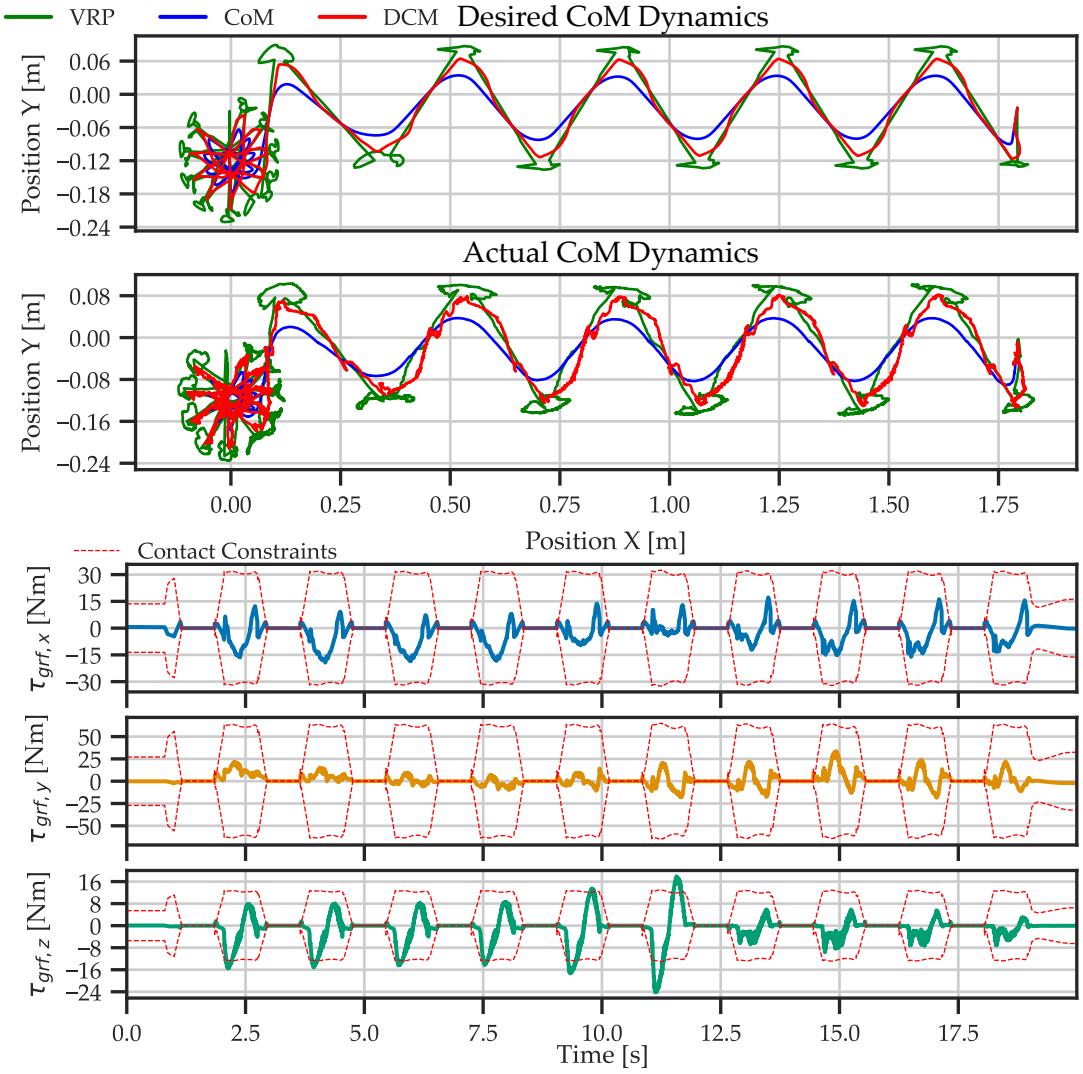


Figure 6.13: Second example of eCMP optimized shared autonomy walking: CoM dynamics and contact torques.

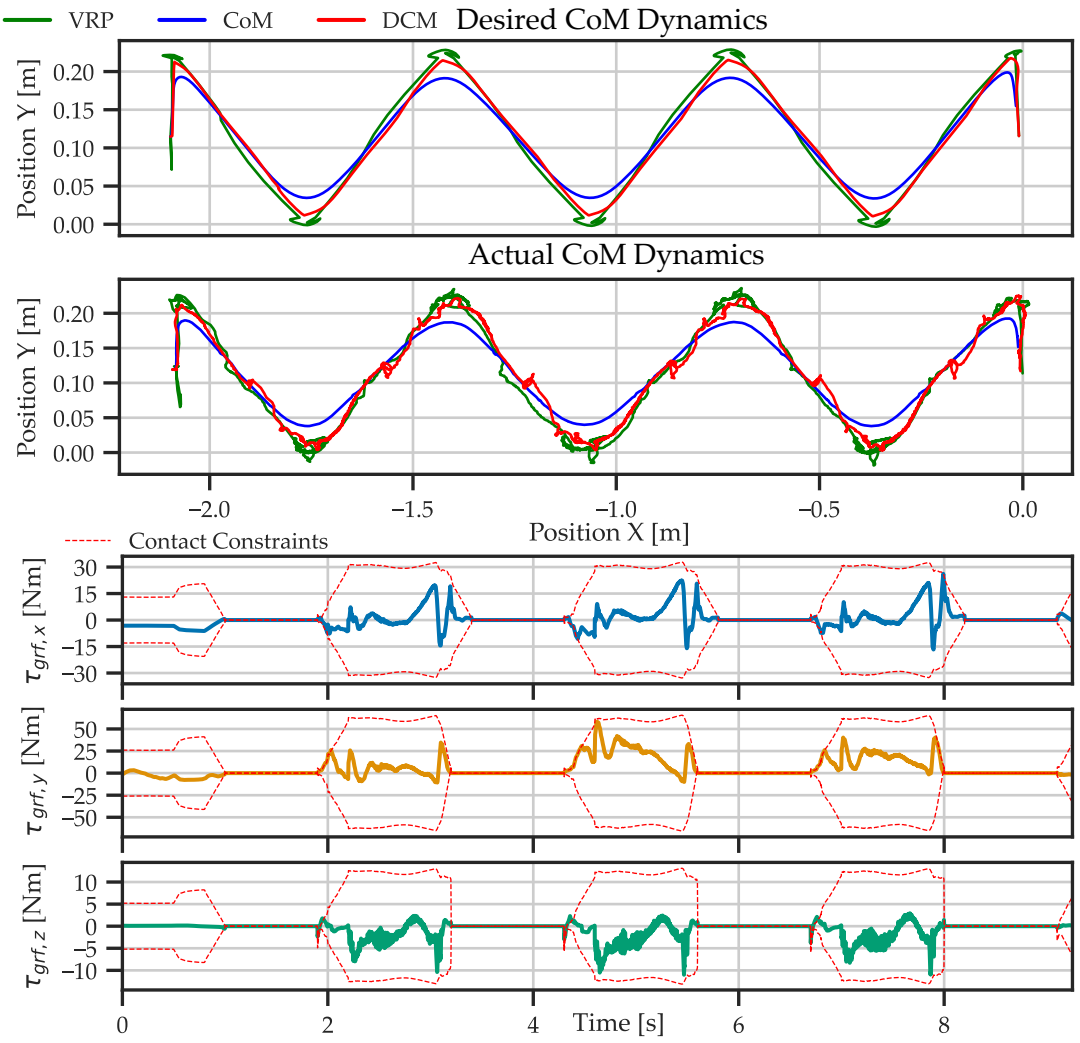


Figure 6.14: CoM dynamics and contact torques during eCMP optimized edge-walking.

6.3.4 Edge Walking

The previous sections were solely concerned with flat-foot walking. However, it can be beneficial or even necessary to employ human-like heel-to-toe motions, e.g., to avoid singularities in the knee joint when taking long steps [2, 3]. These heel-to-toe motions are realized by shifting the CoP to the front edge of the foot during the detachment process while simultaneously tilting the foot about its $\boldsymbol{y}$ axis. That way, only the foot edge is in contact during the detachment process. This walking gait is also referred to as *Edge-Walking*. Fig. 6.14 shows this motion employed along with the CAM-based eCMP optimization. The motion parameters are as follows: $\Delta z = 0.93$m, body pitch: 5°, single-support: 0.9s, double-support: 0.3s, step-length: 35cm. Comparing the previous experiment on straight flat-foot walking (see Fig. 6.8) with the results displayed in Fig. 6.14, the contact torques on the $\boldsymbol{x}$ and $\boldsymbol{y}$ axes appear quite high. Upon closer

inspection, it becomes clear that these high torques are primarily exerted during contact detachment phases, where the CoP is actively moved to the front edge of the foot. In this configuration, the foot's contact with the ground is overall less rigid, such that the CoP is likely to move sideways on the line defined through the foot's front edge.

6.3.5 Diagonal Walking

Large diagonal steps are a particularly challenging scenario for flat-foot walking, as they naturally involve large swing-leg trajectories that generate a substantial amount of CAM. Fig. 6.15 visualizes the effectiveness of the eCMP optimization in these movements, which, without optimization, would have likely led to foot tilting and subsequently to the robot falling. The motion parameters consist of a maximal step length of $X = 20$cm, $Y = 10$cm, a single-support time of 0.7s, and a double-support time of 0.15s.

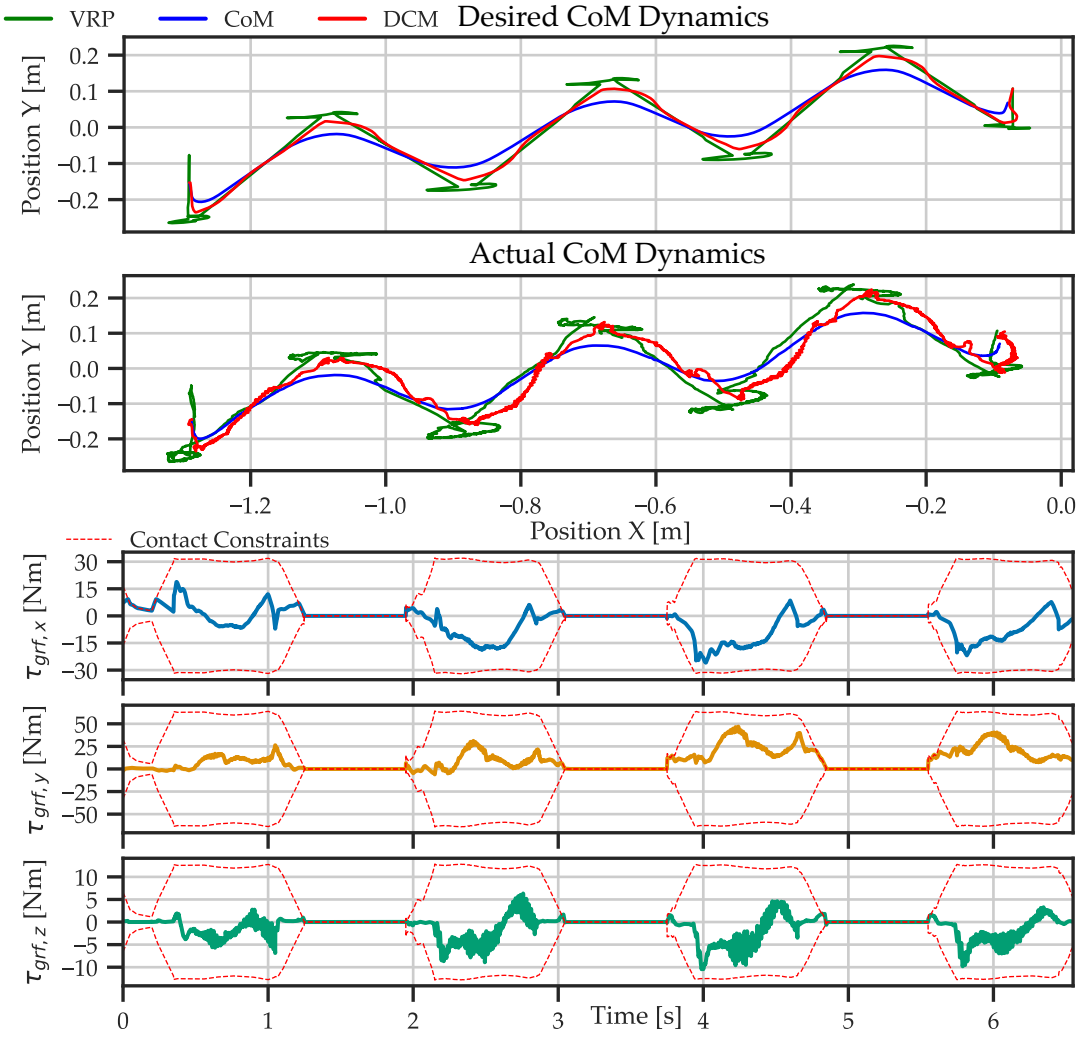


Figure 6.15: eCMP optimized diagonal walking with dynamic motion parameters.

Chapter 7

Conclusion and Future Work

This thesis proposed a sliding-window approach that enables an incremental optimization of the VRP trajectory of a given 3D-DCM motion plan based on the robot's CAM. The CAM is thereby obtained from a short preview window of the robot's multi-body dynamics during the planned motion, computed before its actual execution. The effectiveness of the proposed sliding-window approach was successfully evaluated on numerous motion examples. Especially the proposed algorithm's applicability to shared autonomy walking stands out in comparison to previous methods. A high amount of contact torques, acting about the z axis of the contact, was observed during the experiments, for both optimized and unoptimized motion plans. Additional investigations on this matter should be conducted to mitigate the respective effect, especially for motion plans that contain sharp turns. The internal structure of the torque-controlled joints in the TORO system prevents experiments on gaits that involve impact collisions, such as running, jumping, or skipping, without risking potential damage to the system. Thus, experiments on a different humanoid, capable of handling the respective impacts, would be desirable.

As a second contribution, a conceptual adaptation of the motion planner was proposed, which is envisioned to enable real-time capable short-term replanning, e.g., for active obstacle avoidance. The proposed path search module thereby depends on a set of non-colliding alternative contacts supplied by an external vision module. An efficient method for validating a stance's dynamic feasibility, along with a search algorithm that efficiently handles multi-contact settings, is an open research problem and could be explored in future studies.

Abbreviations

TORO Torque-Controlled Humanoid Robot

CoM Center of Mass

CoP Center of Pressure

ZMP Zero Moment Point

DCM Divergent Component of Motion

3D-DCM three-dimensional Divergent Component of Motion

CMP Centroidal Moment Pivot

eCMP Enhanced Centroidal Moment Pivot

VRP Virtual Repellent Point

CAM Centroidal Angular Momentum

DLR German Aerospace Center

LWR Light Weight Robot

CAD Computer-Aided Design

DoF Degrees of Freedom

IK Inverse Kinematics

EJML Efficient Java Matrix Library

SLAM Simultaneous Localization And Mapping

ROI Region Of Interest

List of Figures

List of Algorithms

Bibliography

[1] B. Siciliano and O. Khatib. *Handbook of Robotics*. Springer Berlin Heidelberg, 2007.

[2] G. Mesesan, R. Schuller, J. Englsberger, C. Ott, and A. Albu-Schäffer. "Unified Motion Planner for Walking, Running, and Jumping Using the Three-Dimensional Divergent Component of Motion." In: *IEEE Transactions on Robotics* 39.6 (2023), pp. 4443–4463.

[3] 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. on Humanoid Robots*. 2019, pp. 25–32.

[4] G. Mesesan, R. Schuller, J. Englsberger, M. A. Roa, J. Lee, C. Ott, and A. Albu-Schäffer. "Motion Planning for Humanoid Locomotion: Applications to Homelike Environments." In: *IEEE Robotics & Automation Magazine* 32.1 (2025), pp. 35–48.

[5] R. Schuller, G. Mesesan, J. Englsberger, J. Lee, and C. Ott. "Online Centroidal Angular Momentum Reference Generation and Motion Optimization for Humanoid Push Recovery." In: *IEEE Robotics and Automation Letters* 6.3 (2021), pp. 5689–5696.

[6] J. Englsberger et al. "Overview of the torque-controlled humanoid robot TORO." In: *Proc. IEEE-RAS Int. Conf. on Humanoid Robots*. 2014, pp. 916–923.

[7] J. Englsberger, C. Ott, and A. Albu-Schäffer. "Three-Dimensional Bipedal Walking Control Based on Divergent Component of Motion." In: *IEEE Transactions on Robotics* 31.2 (2015), pp. 355–368.

[8] J. Englsberger, C. Ott, and A. Albu-Schäffer. "Three-dimensional bipedal walking control using Divergent Component of Motion." In: *Proc. IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*. 2013, pp. 2600–2607.

[9] 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. on Robotics and Automation*. 2022, pp. 10442–10448.

[10] R. Schuller, G. Mesesan, J. Englsberger, J. Lee, and A. Albu-Schäffer. "Centroidal Angular Momentum-Aware Locomotion: Enabling Fast and Robust Walking and Running." In: *Submission to IEEE Robotics and Automation Letters* (2025).

[11] H. Herr and M. Popovic. "Angular momentum in human walking." In: *Experimental Biology* 211.4 (2008), pp. 467–481.

[12] D. E. Orin, A. Goswami, and S.-H. Lee. "Centroidal dynamics of a humanoid robot." In: *Autonomous Robots* 35.2 (2013), pp. 161–176.

[13] A. Herzog, N. Rotella, S. Schaal, and L. Righetti. "Trajectory generation for multi-contact momentum-control." In: *Proc. IEEE-RAS Int. Conf. on Humanoid Robots*. 2015, pp. 874–880.

[14] T. Seyde, A. Shrivastava, J. Englsberger, S. Bertrand, J. Pratt, and R. J. Griffin. "Inclusion of Angular Momentum During Planning for Capture Point Based Walking." In: *Proc. IEEE Int. Conf. on Robotics and Automation*. 2018, pp. 1791–1798.

[15] 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. on Robotics and Automation*. 2021, pp. 2832–2838.

[16] S. Kajita, F. Kanehiro, K. Kaneko, K. Fujiwara, K. Harada, K. Yokoi, and H. Hirukawa. "Biped walking pattern generation by using preview control of zero-moment point." In: *Proc. IEEE Int. Conf. on Robotics and Automation*. Vol. 2. 2003, pp. 1620–1626.

[17] K. Nishiwaki and S. Kagami. "Online Walking Control System for Humanoids with Short Cycle Pattern Generation." In: *Robotics Research* 28.6 (2009), pp. 729–742.

[18] K. Hu, C. Ott, and D. Lee. "Learning and Generalization of Compensative Zero-Moment Point Trajectory for Biped Walking." In: *IEEE Transactions on Robotics* 32.3 (2016), pp. 717–725.

[19] S. Wang, G. Mesesan, J. Englsberger, D. Lee, and C. Ott. "Online Virtual Repellent Point Adaptation for Biped Walking using Iterative Learning Control." In: *Proc. IEEE-RAS Int. Conf. on Humanoid Robots*. 2021, pp. 112–119.

[20] G. Mesesan, J. Englsberger, and C. Ott. "Online DCM Trajectory Adaptation for Push and Stumble Recovery during Humanoid Locomotion." In: *Proc. IEEE Int. Conf. on Robotics and Automation*. 2021, pp. 12780–12786.

[21] R. M. Murray, S. S. Sastry, and L. Zexiang. *A Mathematical Introduction to Robotic Manipulation*. 1st. CRC Press, 1994.

[22] K. M. Lynch and F. C. Park. *Modern Robotics: Mechanics, Planning, and Control*. 1st. Cambridge University Press, 2017.

[23] G. Garofalo, C. Ott, and A. Albu-Schäffer. "On the closed form computation of the dynamic matrices and their differentiations." In: *Proc. IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*. Tokyo, 2013, pp. 2364–2359.

[24] D. Ostermeier, J. Külz, and M. Althoff. "Automatic Geometric Decomposition for Analytical Inverse Kinematics." In: *IEEE Robotics and Automation Letters* 10.10 (2025), pp. 9964–9971.

[25] P. Sardain and G. Bessonnet. "Forces Acting on a Biped Robot. Center of Pressure–Zero Moment Point." In: *IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans* 34.5 (2004), pp. 630–637.

[26] 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. on Robotics and Automation*. 2017, pp. 4108–4115.

[27] 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. on Humanoid Robots*. 2014, pp. 266–272.

[28] 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." In: *IEEE Robotics and Automation Letters* 3.4 (2018), pp. 3449–3456.

[29] B. Henze, M. A. Roa, and C. Ott. "Passivity-based whole-body balancing for torque-controlled humanoid robots in multi-contact scenarios." In: *Robotics Research* 35.12 (2016), pp. 1522–1543.

[30] U. M. Ascher and C. Greif. "Chapter 11: Piecewise Polynomial Interpolation." In: *A First Course in Numerical Methods*. Society for Industrial and Applied Mathematics, 2011, pp. 331–363.

[31] R. J. Griffin, G. Wiedebach, S. McCrory, S. Bertrand, I. Lee, and J. Pratt. "Footstep Planning for Autonomous Walking Over Rough Terrain." In: *Proc. IEEE-RAS Int. Conf. on Humanoid Robots*. 2019, pp. 9–16.

[32] D. Calvert, B. Mishra, S. McCrory, S. Bertrand, R. Griffin, and J. Pratt. "A Fast, Autonomous, Bipedal Walking Behavior over Rapid Regions." In: *Proc. IEEE-RAS Int. Conf. on Humanoid Robots*. 2022, pp. 24–31.

[33] I. Tsikelis, E. Tsiatsianas, C. Kiourt, S. Ivaldi, K. Chatzilygeroudis, and E. Mingo Hoffman. "AHMP: Agile Humanoid Motion Planning With Contact Sequence Discovery." In: *Proc. IEEE-RAS Int. Conf. on Humanoid Robots*. 2025.

[34] H. Liu, Q. Sun, and T. Zhang. "Hierarchical RRT for humanoid robot footstep planning with multiple constraints in complex environments." In: *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*. 2012, pp. 3187–3194.

[35] D. Maier, A. Hornung, and M. Bennewitz. "Real-time navigation in 3D environments based on depth camera data." In: *Proc. IEEE-RAS Int. Conf. on Humanoid Robots*. 2012, pp. 692–697.

[36] A. Hornung, A. Dornbush, M. Likhachev, and M. Bennewitz. "Anytime search-based footstep planning with suboptimality bounds." In: *Proc. IEEE-RAS Int. Conf. on Humanoid Robots*. 2012, pp. 674–679.

[37] J. Chestnutt, K. Nishiwaki, J. Kuffner, and S. Kagami. "An adaptive action model for legged navigation planning." In: *Proc. IEEE-RAS Int. Conf. on Humanoid Robots*. 2007, pp. 196–202.

[38] N. Funk, J. Tarrio, S. Papatheodorou, M. Popović, P. F. Alcantarilla, and S. Leutenegger. "Multi-Resolution 3D Mapping With Explicit Free Space Representation for Fast and Accurate Mobile Robot Motion Planning." In: *IEEE Robotics and Automation Letters* 6.2 (2021), pp. 3553–3560.

[39] H. Oleynikova, Z. Taylor, M. Fehr, R. Siegwart, and J. Nieto. "Voxblox: Incremental 3D Euclidean Signed Distance Fields for on-board MAV planning." In: *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*. 2017, pp. 1366–1373.

[40] V. Reijgwart, C. Cadena, R. Siegwart, and L. Ott. "Efficient volumetric mapping of multi-scale environments using wavelet-based compression." In: *Proceedings of Robotics: Science and Systems*. 2023.

[41] C. Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006.

[42] F. Kanehiro, K. Fujiwara, S. Kajita, K. Yokoi, K. Kaneko, H. Hirukawa, Y. Nakamura, and K. Yamane. "Open architecture humanoid robotics platform." In: *Proc. IEEE Int. Conf. on Robotics and Automation*. Vol. 1. 2002, pp. 24–30.

[43] G. Hirzinger, N. Sporer, A. Albu-Schäffer, M. Hähnle, R. Krenn, A. Pascucci, and M. Schedl. "DLR's torque-controlled light weight robot III-are we reaching the technological limits now?" In: *Proc. IEEE Int. Conf. on Robotics and Automation*. Vol. 2. 2002, pp. 1710–1716.