Motion Planning for Humanoid Locomotion: Applications to Homelike Environments

George Mesesan, Robert Schuller, Johannes Englsberger, Máximo A. Roa, Jinoh Lee, Christian Ott, and Alin Albu-Schäffer

***Abstract*—"What can your humanoid robot do?" is probably the most commonly asked question that we, as roboticists, have to answer when interacting with the general public. Often, the question is framed in the familiar household or office setting, with implied expectations of robust locomotion on uneven and cluttered terrain, and compliant interaction with people, objects, and the environment. Moreover, the question implies the existence within the humanoid robot of a set of embodied loco-manipulation skills implemented by a motion planner, skills that are retrievable when given the corresponding commands. In this article, we formulate an answer to this question in the form of an efficient, modular, and extensible motion planner. We demonstrate its use with three challenging scenarios, designed to highlight both the robot's safe operation and its precise movement in unstructured environments. Additionally, we discuss key techniques derived from our experience in the practical implementation of torque-controlled humanoid robots.**

I. Introduction

HUMANOID robots are poised to take a historic step out of the science laboratories and into the manufacturing and logistics industries. After being the focus of robotics research for several decades [1], [2], a new generation of high-performance humanoid robots is now being built by industrial and technological companies, like Tesla, Figure AI, 1X, and Unitree [3], joining the highly successful predecessors: Honda's Asimo [4] and Boston Dynamics' Atlas [5].

As humanoid robots develop further, they are moving closer to becoming versatile assistants in daily scenarios, as companions and household helpers. However, everyday tasks that seem straightforward to us and are routinely performed by humans can be extremely complex in the unpredictable and unstructured settings of domestic environments. In particular, generating and executing a coherent motion for a humanoid robot with its high number of degrees of freedom (DoF) constitutes a daunting challenge. The difficulties are further aggravated by the inherently unstable nature of bipedal locomotion, with its high center-of-mass (CoM) and small base of support. These challenges are typically addressed by partitioning the main task into a sequence of motions that fulfill the overall goal (motion planning), and executing this sequence while reacting to unforeseen disturbances (whole-body control). Both topics have a rich history of research and development in the robotics literature.

All authors are with the Institute of Robotics and Mechatronics, German Aerospace Center (DLR), Wessling, Germany. Corresponding author: george.mesesan@dlr.de

Christian Ott is also with the Automation and Control Institute, Faculty of Electrical Engineering and Information Technology (TU Wien), Vienna, Austria

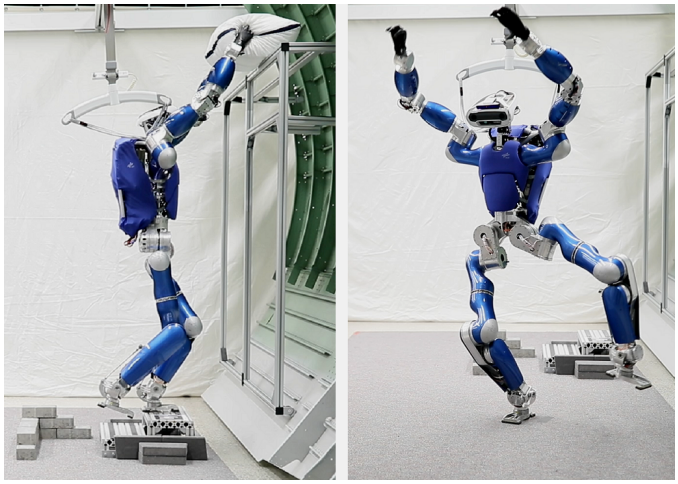

Fig. 1. The humanoid robot TORO, performing two challenging tasks: retrieving a cushion from a high shelf (left), and dynamic balancing during a yoga exercise (right).

Given its critical role in enabling humanoid robots to move safely and effectively, this article particularly emphasizes motion planning. Commonly used planning methods include offline trajectory generation such as the one used by Atlas [5], offline training using Reinforcement Learning (RL) [6], and online optimization-based approaches such as Model Predictive Control (MPC) [7]. Due to their compact formulation and their ability to find useful motions through the extensive search of the solution space, these methods are becoming increasingly popular. However, they are showing significant limitations concerning extensibility and explainability. For example, extending the set of executable skills by adding or modifying existing skills in a RL-based motion planner is a highly challenging problem due to the diffuse encoding of the motion generation in a neural network. Furthermore, as a consequence of employing numerical (data- or optimization-based) methods, the planner can often give no clear explanation of how or why a certain motion was generated.

In this article, we propose an online, extensible, and efficient motion planner, which can serve as a unifying planning interface for a large variety of humanoid and legged robots. The guiding idea that we consistently pursue is *dimensionality reduction*. What can be simpler than issuing straightforward commands like "Bring me a cushion" and "Let's do yoga together" (Fig. 1), or directly controlling a humanoid robot with a joystick on a standard game controller? These command modalities are both intuitive and desirable from the

 DOI: 10.1109/MRA.2025.3527287

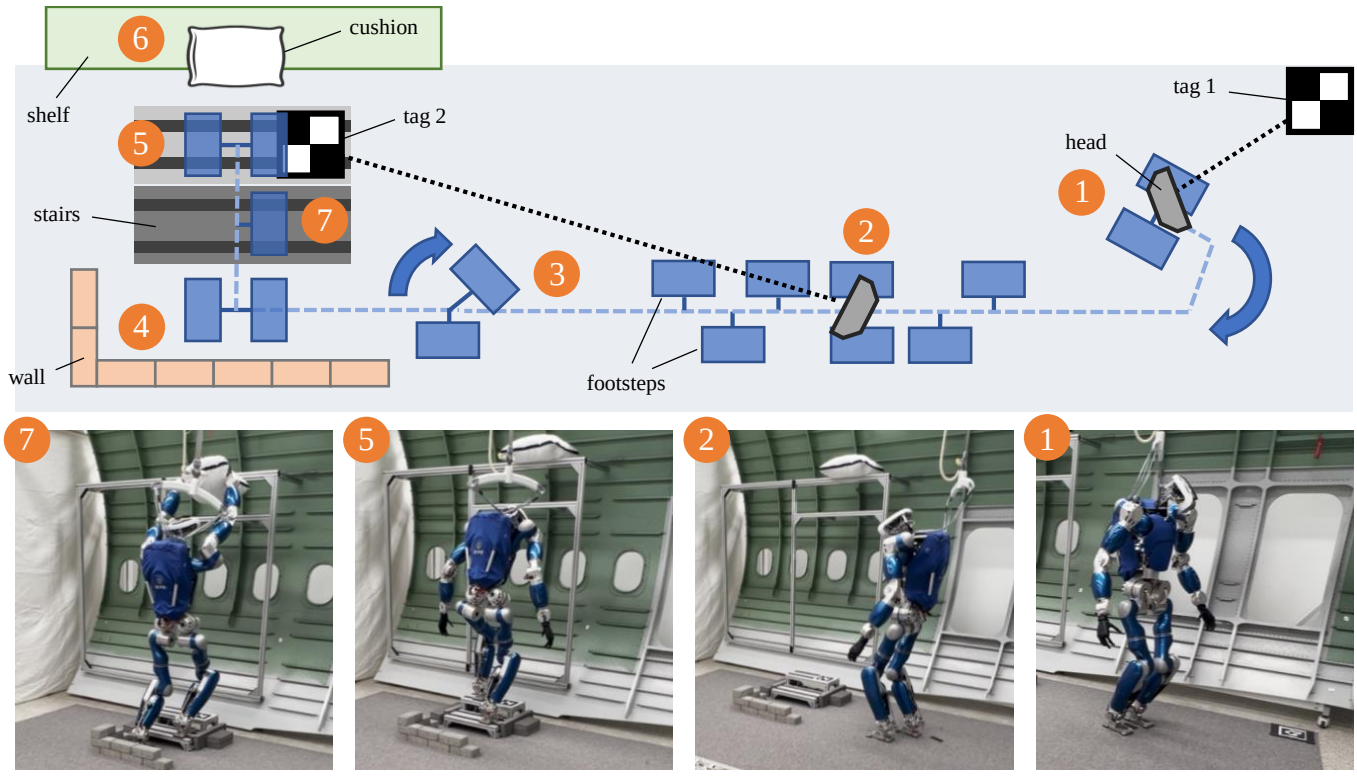

Fig. 2. Demonstration of vision-based locomotion. The robot is asked to autonomously navigate to the shelf and retrieve the cushion. The robot self-localizes in the world map using the AprilTags attached to the environment.

point of view of human-robot interaction. Adopting a top-down perspective, our motion planner automatically expands this high-level command into a sequence of simpler actions (walking, turning, balancing), these in turn into coordinated whole-body motions (taking a step, climbing a stair step, etc.), continuing thus down to the level of executable instructions using mathematical constructs like polynomials and other closed-form expressions. Seen from a bottom-up viewpoint, the motion planner performs a dimensionality reduction at each level of abstraction, building libraries of reusable motion components from which the higher-level elements are being constructed.

A further important feature of our motion planner is that the generated reference trajectories are continuous at the level of accelerations (C^2 continuous). As a general principle, we aim to avoid discontinuities in the reference trajectories, as these lead to two undesirable effects. First, because the robot has a limited control bandwidth, the whole-body controller is unable to follow the reference trajectory in case of a discontinuity, leading to an unnecessary controller tracking error. Second, the sharp controller response caused by the reference discontinuity tends to excite inherent joint and link elasticities of the humanoid robot, thereby further degrading the tracking performance.

From the point of view of the robot operator, our motion planner offers a set of high-level motion skills such as walking, turning, stair climbing, etc., which can be activated individually or concatenated to form complex sequences. All motion skills have default configurations that make them immediately usable, or, alternatively, the robot operator can configure them as desired at various levels of granularity. Once the robot operator issues a command to the robot, the whole motion planning process is performed online and is fully autonomous, requiring no additional human input. At the same time, the motion planner is extensible with new motion elements at all levels of abstraction. The final result is a highly flexible and configurable planning environment that can be used to implement various scenarios in challenging environments.

II. Demonstrations

In this section, we describe three different application scenarios for humanoid robots in homelike environments. A video of the performed experiments can be found as a multimedia attachment.

The demonstrations are performed using DLR's torque-controlled humanoid robot TORO [8], a 27 degrees-of-freedom (DoF) robot, with a height of 1.74 m and a total weight of 79.2 kg. Unlike position-controlled robots, torque-controlled robots [8]–[10] are capable of directly controlling the forces exerted by their joints, enabling both gentle interactions with the environment and precise movements. Moreover, TORO's whole-body control algorithm is completely based on the concept of impedance control, which guarantees passivity and compliance in human-robot and robot-environment interactions. This type of controller allows the robot's feet to reactively adapt to different types of ground surfaces during locomotion, including gravel, grass, sports mattresses, or scattered Lego blocks.

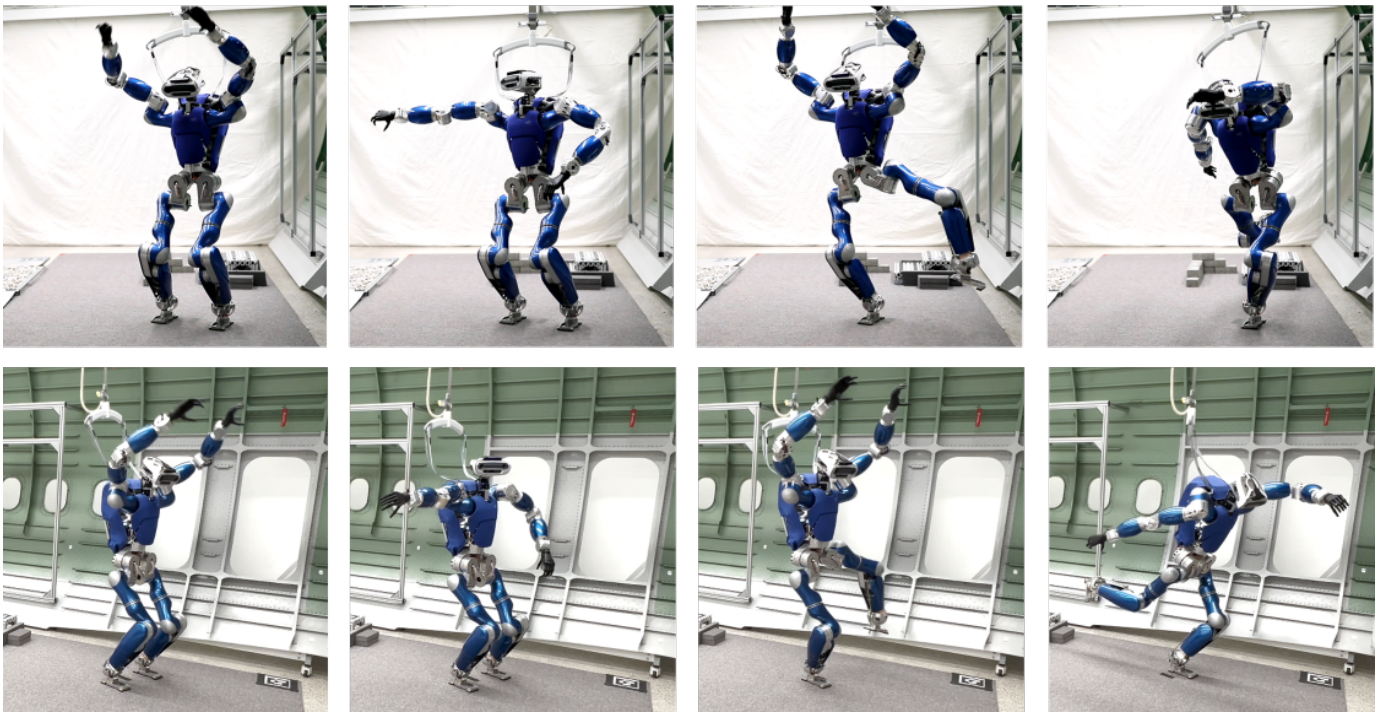

Fig. 3. The humanoid robot TORO performs dynamic balancing tasks inspired by yoga exercises. The sequence of images from the complete gymnastics routine is presented from left to right. The top row depicts the front view, while the bottom row depicts a three-quarter view.

Demonstration A: Vision-based Locomotion

In the first scenario, the robot is asked to autonomously retrieve a cushion stored on a shelf that is accessible only through a narrow entry-point into a confined space and a set of stairs. An overview of the entire motion sequence is provided in Fig. 2. In the following, we describe the individual motion elements of the motion sequence.

Point 1: First, the robot uses an AprilTag [11] attached to the ground (tag 1) to determine its own position and orientation with respect to the workspace objects (self-localization). Once this is known, the motion sequence to turn and walk towards the stairs is generated.

Point 2: The robot stops and localizes the stairs using the AprilTag attached to the stairs (tag 2). While tag localization is also possible during walking, the localization accuracy is higher when the robot is stationary. Here, due to the challenging task of navigating the narrow space in front of the stairs and climbing the stairs, we opted for the high localization accuracy in detriment of the task execution speed.

Point 3: Since the space in front of the stairs is highly confined, TORO has to turn and walk sideways to approach the stairs. Note that, from this moment on, a high positioning accuracy for the step placements is required.

Point 4: The robot leans its upper body forward while maintaining a constant CoM position, moving its hips backwards in the process. This posture significantly reduces the maximum torque required in the knee joints during stair climbing.

Point 5: TORO climbs the stairs dynamically, utilizing only one foot per stair step.

Point 6: The robot grasps the cushion using its inherent compliance provided by the impedance control strategy. This task requires precise torque control and a high positioning accuracy to avoid a collision with the shelf.

Point 7: The robot descends the stairs backwards with one foot per stair step while carrying the cushion.

After descending the stairs, the robot exits the confined space by again walking sideways, then returns to the starting position, and delivers the cushion.

Demonstration B: Dynamic Balancing

In the second scenario, the robot performs several yoga poses (Fig. 3) that can be envisioned as a motivation for the elderly to maintain a healthy body activity level. In addition to the motion planner capabilities, this scenario demonstrates the high performance and robustness of the torque-controlled robot in combination with a passivity-based whole-body controller. Note that all poses are performed in a continuous dynamic motion sequence.

Demonstration C: Shared Autonomy Walking

In the last scenario, we demonstrate how the robot can navigate a homelike environment in a shared autonomy mode (see Fig. 4). The robot localizes itself in a world map via AprilTags attached to the environment. The operator commands the walking direction and velocity via a standard gamepad. At the same time, the robot validates the human input for feasibility and ignores commands if these lead to violations of the defined boundaries. These constraints consist of workspace boundaries such as walls, virtual barriers, or inaccessible ground areas. The operator can observe the robot world representation and the workspace boundaries on a screen where this information is provided as a digital twin environment.

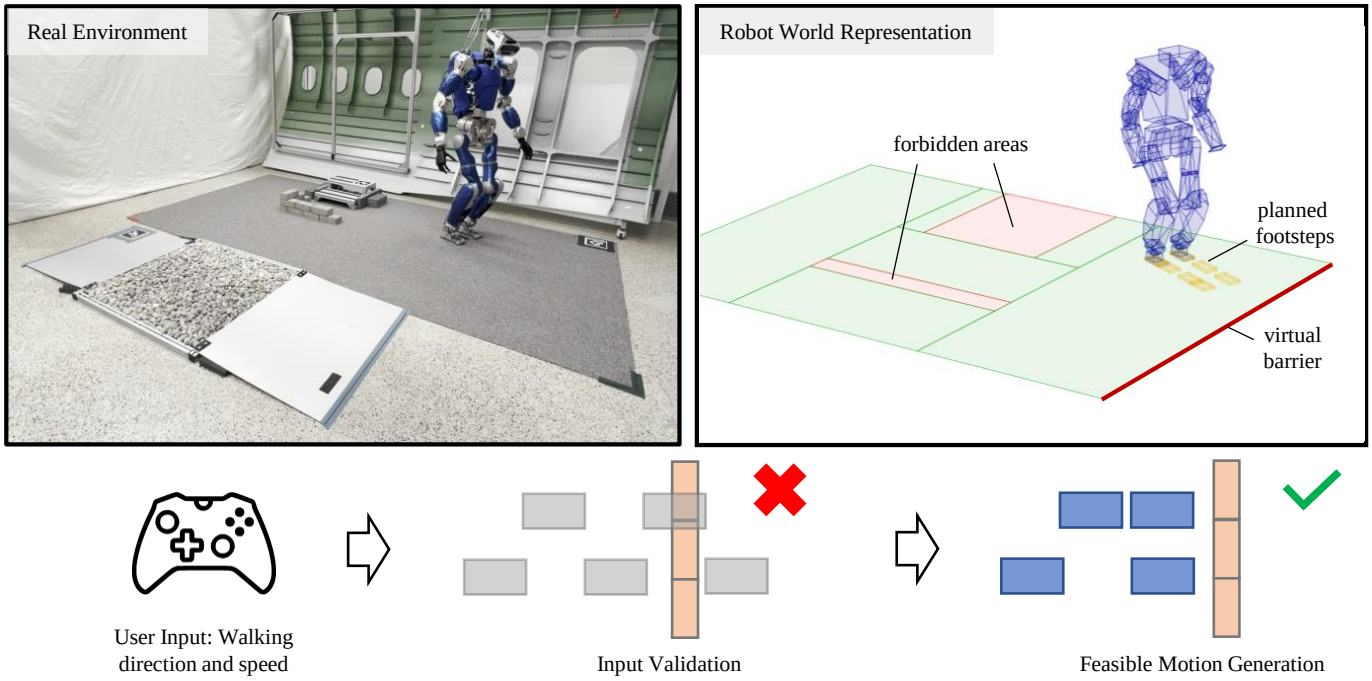

Fig. 4. Demonstration of shared autonomy walking. The operator commands the walking direction and speed while the robot validates the inputs for feasibility based on workspace boundaries, and updates the motion sequence accordingly. The top right image shows the digital twin environment that the robot operator can observe on a screen, depicting the planned footsteps, the forbidden area, and the virtual barrier that the robot is not allowed to cross.

III. Motion Planning

In our presented demonstrations, the human operator initiates the humanoid robot's motion by giving a high-level command through a joystick input device or a computer user interface. The command can be as straightforward as executing a simple action such as walking to a certain point, turning in place, crouching, etc., or more complex such as bringing an object (Demonstration A) or performing a gymnastics routine (Demonstration B). The role of the motion planner is to transform autonomously the high-level command into continuous reference trajectories of the quantities relevant to humanoid locomotion: CoM position, body orientation, angular momentum, feet positions, and upper body configuration. These quantities constitute the whole-body task that the tracking controller is realizing on the humanoid robot through the torque commands.

The motion planner has five main components (see Fig. 5):

- Action sequencer: transforms the high-level command into a sequence of simple actions (walking, turning, balancing) with corresponding intermediate goals.
- Footstep planner: generates successive contacts (footsteps) such that each intermediate goal is fulfilled.
- Plan assembler: creates a motion plan composed of multiple phases for each locomotion subtask quantity (CoM position, base orientation, etc.)
- Reference trajectory generator: produces the instantaneous values for each locomotion subtask and assembles the whole-body task.
- Footstep adjustment module: computes contact adjustments in response to unrecoverable tracking errors reported by the whole-body controller

The actions sequencer, the footstep planner, and the plan assembler are executed asynchronously, i.e., using an event-based execution principle. In contrast, the reference trajectory generator and the footstep adjustment module are part of the realtime process, and are computed synchronously at the execution rate of the whole-body controller (1 kHz). In the following, we describe each module in more detail and discuss their functionality in the context of the demonstrations presented above.

For the sake of clarity, we provide here explicit definitions of common concepts that are used throughout the article. We denote as the robot's *pose* the combined CoM position and body orientation expressed in world coordinates; a sub-quantity in the xy-plane is the $2D$ *pose*, representable as the tuple (x, y, yaw). Combining the robot's pose with the complete joint configuration produces the robot's *posture*. The robot's *stance* is defined as the set of contacts the robot makes with the environment, each *contact* being characterized by the position and orientation of the employed end-effector. Typical stance types are single- and double-support stances, and, in the demonstrations presented here, the end-effectors used for locomotion are always the robot's feet. Therefore, in the rest of the article we use the terms contact and *footstep* interchangeably. Nevertheless, for multi-contact locomotion and balancing, other parts of the body can be used to establish contacts with the environment: hands, elbows, knees, etc., as we have shown in our previous work [12]. Regarding contact transitions, we use the terms *attach* and *detach* to denote making and breaking contacts, respectively.

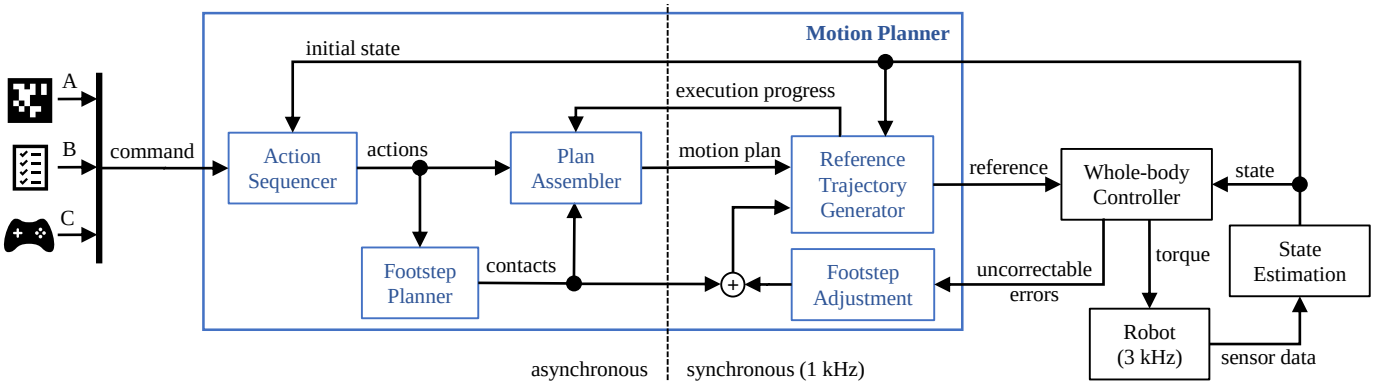

Fig. 5. Overview of the motion planner. The modules composing the motion planner are depicted in blue. The high-level command can be one of the following: adapt an action sequence with information gathered through computer vision (Demonstration A), follow a predefined sequence of actions (Demonstration B), or can be given directly through a joystick interface (Demonstration C).

A. Action sequencer

In response to a high-level command from the human operator, an action sequence is constructed by either adapting a predefined sequence with information gathered through computer vision (Demonstration A), retrieving a stored sequence (Demonstration B), or instantaneously following the commands given through a joystick interface (Demonstration C). Each action is characterized by a specific goal posture and is constructed such that the robot can maintain this posture in static balance indefinitely after finishing the respective motion. For example, a walking action to a certain waypoint is specified at this stage only as the final pose of the robot at the given target location. The number of steps and their exact locations are not yet known, these computations being the responsibility of the footstep planner. In consequence, the total duration of the action is also unknown, as it depends on the number of steps and the time parametrization of the walking gait (single/double support times). These aspects of the motion planning process are covered by the plan assembler.

The action sequencer instantiates actions from a library of configurable action templates that represent basic motions of the humanoid robot. Each action template is a reusable motion element that can be configured with a high level of granularity. We distinguish among four types of executable actions, cataloged here by increasing complexity:

1) Elementary actions affect only one subtask of the whole-body motion, leaving the other subtasks unaffected. Natural instances of elementary actions are: SHIFT-COM, CHANGE-ORIENTATION, MOVE-LIMB, MOVE-JOINT, OPEN-HAND, etc. For example, the CHANGE-ORIENTATION action can be used to change the robot's body orientation relative to the world frame by a configurable amount. We use this action during Demonstration A to command the robot to lean forward (increase the body pitch angle by $18°$) before climbing the stairs.

2) Balancing actions require the coordination of two or more subtasks while the robot maintains a balancing posture. Typical examples are stance-changing actions like ATTACH-CONTACT and DETACH-CONTACT, where the addition or removal of a contact to the current stance and the corresponding CoM motion need to be coordinated. A more complex example is the REPOSITION-CONTACT action, which encapsulates into a single action a sequence of three simpler actions performed with the same limb: DETACH-CONTACT, MOVE-LIMB, ATTACH-CONTACT. The ATTACH-CONTACT and DETACH-CONTACT actions are used during Demonstration B to switch to the single-support stance and back to double-support for the balancing exercises, with REPOSITION-CONTACT being used to change to a wide stance before performing the arm motion exercises.

3) Locomotion actions consist of successive steps with alternating right and left feet. Using the terminology introduced above, each step can be described as a REPOSITION-CONTACT action of the corresponding leg. Common locomotion actions that are used in our demonstrations are: WALK-TO-WAYPOINT, TURN-IN-PLACE, CLIMB-STAIRS, DESCEND-STAIRS. Additionally, our action library contains further locomotion gaits, such as running, jumping, or skipping, which we described in our previous work [13]. However, the absence of shock-absorbing elements in the real hardware as well as the limitations in joint torques and velocities prevent us from demonstrating these gaits in an experimental setting.

4) Composite actions combine two or more actions into one standalone entity. A composite action extends the basic motion of one of the actions presented so far with additional subtasks, for example, by adding arm and hand motions during walking for a loco-manipulation task. In Demonstration A, a composite action is used for the cushion grasping motion, combining arm motions (MOVE-LIMB actions for the left and right arm, respectively) with a vertical CoM adjustment (SHIFT-COM action). The commanded CoM vertical shift ensures that the robot's hips maintain a relatively constant height above ground during the arms' upward motion.

The complete sequence of actions corresponding to the high-level command performed in Demonstration A is given in Table 1. In general, the action parameters used in this demonstration are fixed (lean forward $18°$, 2 stair steps), fitting our robot's capabilities and the predefined environment. Ideally, these parameters could be captured or computed according to an advanced perception algorithm that interprets the environment to acquire the corresponding parameters for each action template. Note that even though the sequence of

Action	Parameter / Comment
WALK-TO-WAYPOINT	waypoint 2 (relative to tag 1)
WALK-TO-WAYPOINT	waypoint 3 (relative to tag 2)
TURN-IN-PLACE	align yaw angle with tag 2
WALK-TO-WAYPOINT	waypoint 4 (lateral walking)
CHANGE-ORIENTATION	lean forward 18°
CLIMB-STAIRS	2 stair steps
CHANGE-ORIENTATION	stand upright
GRASP-CUSHION	composite action
DESCEND-STAIRS	2 stair steps
LOWER-ARMS	composite action
WALK-TO-WAYPOINT	waypoint 3 (lateral walking)
WALK-TO-WAYPOINT	waypoint 1 (start pose)
RELEASE-CUSHION	composite action

Table 1. Action sequence for cushion retrieval with vision-based locomotion (Demonstration A).

actions is predefined, the locations of the waypoints 2, 3, and 4 are specified relative to the tag poses, which are unknown at the beginning of the demonstration. Therefore, the concrete actions composing the action sequence can only be instantiated when the tag locations are determined. During the localization procedure, while the vision module reorients the camera by moving the robot's head, and the tag detection algorithm is executed, the action sequencer is in idle mode, waiting for the localization result. This situation occurs at waypoint 1 (start pose) for tag 1 localization, and at waypoint 2 for tag 2, after executing the first WALK-TO-WAYPOINT action. Even though no explicit action is generated, this situation is interpreted by the plan assembler and the reference trajectory generator as an instruction to maintain the current reference whole-body task while balancing. The same behaviour is implemented for Demonstration C (see Fig. 4), whenever there is no operator input given through the joystick controller. We discuss these cases in more detail below.

B. Footstep planner

The footstep planning process is initiated by the action sequencer whenever a locomotion action is generated. Given the robot's 2D pose at the end of the previous action and the goal pose of the current locomotion action, the footstep planner generates a sequence of contacts with alternating left and right footsteps connecting the two poses (see Fig. 6b). If the previous action is also a locomotion action, the first footstep is chosen such that it naturally continues the existing sequence; i.e., if the previous sequence ends with the left foot, then the first footstep of the current sequence is taken with the right foot, and vice versa. For other types of preceding actions or in the initial pose, the first footstep is determined based on the motion type (see Fig. 6a): forward and backward walking start with the right foot, lateral walking and turning start with the foot corresponding to the direction of motion (right foot if walking to the right, left foot for left turns, etc.). A similar algorithm is employed when the actions are generated online via a joystick interface: during walking, the footstep planner generates alternating footsteps, when no commands are given, the robot stops and maintains balance in a standing pose, and, finally, when commands are resumed, the initial footstep is determined by the direction of motion.

The parameters guiding the footstep planning process are the maximum step length on the x- and y-axes, and the maximum turning angle during one step. In order to prevent collisions between the feet, the footstep planner avoids footstep placements that would cause the swing foot to cross the body's sagittal plane. Furthermore, potential knee collisions are averted by prohibiting the inward turning of the footsteps. The effect of this strategy can be seen in Fig. 6c, where the walking trajectory curves to the right, as seen in the local coordinates. Here, only the right footsteps contribute to the turning motion, as they correspond to outward turns of the foot; in contrast, the left footsteps maintain the same orientation as the preceding right footsteps.

The result of the footstep planning process is a chain of contacts, where each contact is linked to the preceding one by a constraint formulated as a footstep placement bounding area (see Fig. 6c). This information is used by the footstep adjustment module, which can make changes to the contact positions in response to unrecoverable tracking errors reported by the whole-body controller. For example, when adjusting the constrained contact labeled i, the adjustment offset is limited by the requirement that the footstep remains inside the bounding area associated with the reference contact $i-1$, marked in gray in Fig. 6c. Moreover, as the contact i is shifted, its corresponding bounding area (marked in green in Fig. 6c), being rigidly attached, is moved accordingly. If, as a result, the subsequent contact $i+1$ lies outside the bounding area, it is also automatically shifted until it is again placed fully inside the bounding area. This operation is performed for all remaining contacts in the contact chain, and their respective constraints are enforced (contact $i+2$ must remain within the bounding area associated with contact $i+1$, etc.).

The footstep planner also plays an important role in the shared autonomy scenario (Demonstration C). Using a world map consisting of free and forbidden areas, the footstep planner ensures that all footsteps are placed within the free areas. If, as a result of a human operator's command, the generated footstep would be placed outside the workspace boundaries or would intersect one of the forbidden areas, the footstep planner automatically shifts the footstep such that it is placed fully inside the permissible area.

C. Plan assembler

The plan assembler transforms the action sequence into an executable motion plan, which consists of explicitly timed instructions for each subtask of the whole-body motion (CoM position, body orientation, foot pose, etc.). The motion plan can be regarded as a two-dimensional structure, with time as the horizontal axis, and the subtasks forming the vertical axis (see Fig. 7). The basic building block of the motion plan is the motion *phase*, each phase having a configurable duration and encapsulating a single subtask instruction to the reference trajectory generator: maintain a constant value (Hold), interpolate between start and end values using linear or higher order polynomials (Move, Shift), etc. The motion phases can

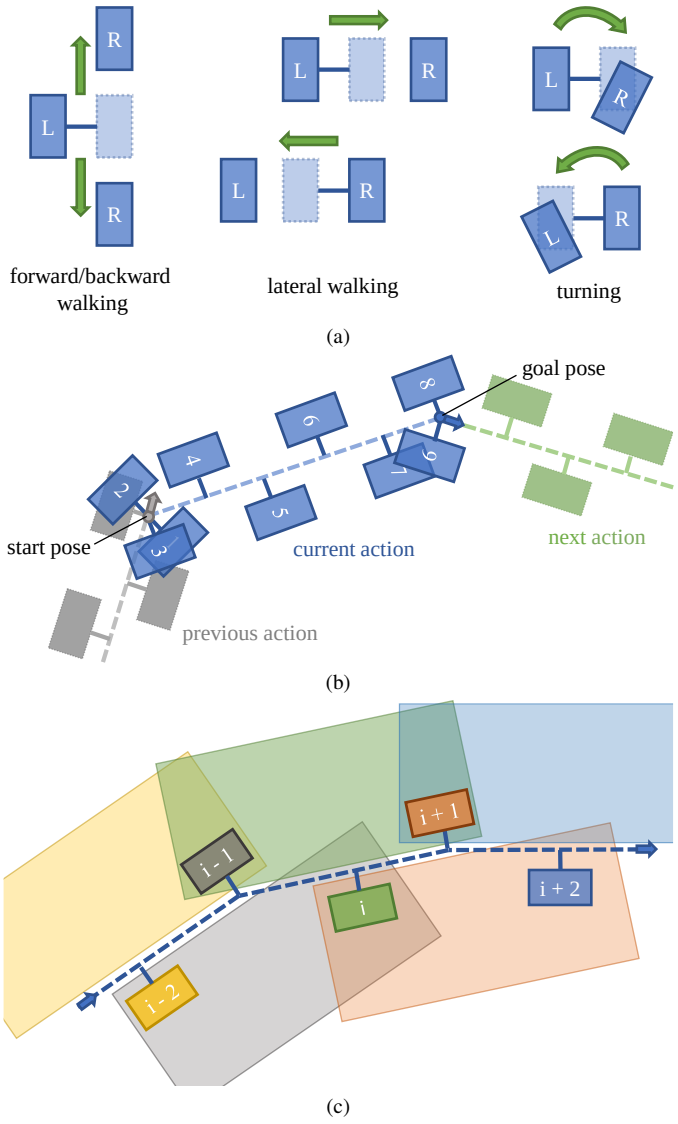

Fig. 6. Footstep planner details: (a) choosing the first footstep depends on the motion type; (b) example of generated footsteps connecting start and goal poses as part of an action sequence; (c) contact sequence with corresponding footstep placement bounds. Each footstep must be placed fully inside the bounding area associated with its preceding footstep. The correspondence between a footstep and its associated bounding area is depicted with matching colors.

be configured with a high degree of granularity, and are linked together such that the final state of one phase becomes the initial state of the subsequent phase. This approach, combined with the usage of C^2 continuous interpolations within each phase, ensures that no discontinuities are created in the reference trajectories. Note that the actual implementation of a particular phase type depends on the subtask on which it is applied. For example, the motion phase labeled "Hold" is used on all subtask plans with the same semantics of maintaining the previous state unchanged for the complete phase duration. However, the specific state values being held constant depend on the actual subtask on which the phase is used.

In most cases, the motions performed by the humanoid robot require a high degree of coordination among the individual subtasks. The classic example in this regard is walking, where foot movements and contact transitions are carefully synchronized with the CoM motion. With these considerations in mind, we introduce a new element called a *move*, which contains the detailed, complete motion plan for a basic unit of motion such as taking a step, climbing a stair step, leaning forward, etc. (see Fig. 8). Moves are designed as a bridging layer between actions and motion phases: each action is implemented by one or multiple moves, and each move contains one or several phases for each subtask. This intermediate layer aspect becomes apparent also in the typical execution times of the various motion elements. While action execution times can stretch up to 10 seconds or more (e.g., WALK-TO-WAYPOINT, CLIMB-STAIRS), and motion phase durations are generally short (e.g., a contact attach phase lasts 50 ms, the double support phase during walking, 300 ms), moves are designed as reusable motion elements with durations around 1 or 2 seconds (e.g. a walking step takes 1.2 s, a climbing step 1.7 s). Of course, the durations given here are meant only for exemplification. Faster or, if needed, slower walking can be easily implemented through the appropriate parametrization of the corresponding moves. In fact, our method of creating CoM trajectories admits arbitrary motion durations with no upper limits, and lower limits only given by the physical constraints of the robot [14].

In an additional role, moves serve as units of communication between the plan assembler and the reference trajectory generator. The motion planner employs a rolling window approach to the plan execution, selecting a fixed number of moves from the total plan to be sent to the reference trajectory generator (Fig. 7). Whenever a move execution is finished, the plan assembler is notified and shifts the rolling window by one, removing the executed move and appending the next move to the executing plan. Consequently, we need to consider the case where the communication between the asynchronous and the synchronous parts of the motion planner is interrupted, and the final move within the rolling window would leave the robot in an undesirable state. One such example is during walking, where the "Walk Step" move ends in a single support stance with the swing foot at the next footstep location without having established contact (see Fig. 8a). To avoid these situations, the plan assembler creates a safe motion plan by appending a stopping step that leaves the robot in a statically stable configuration with both feet parallel to each other (see Fig. 8b and rightmost robot image in Fig. 7). The "Stop Step" implementation and its time parametrization depend on the last move in the plan, with different variants for each walking mode: walking on flat terrain, stair climbing, etc. Note that this stopping step is not inserted into the original plan. Instead, it is appended to a copy of the plan selected by the rolling window method before being sent for execution. Therefore, in the nominal case, the original plan is executed normally, using the rolling window approach presented above.

A special case that we mentioned above is when the action sequencer is waiting for the tag localization procedure to complete, and no actions are being generated. The plan assembler recognizes this state and automatically creates waiting moves, consisting of Hold phases for all subtasks, and appends them to the motion plan. This instructs the robot to maintain its

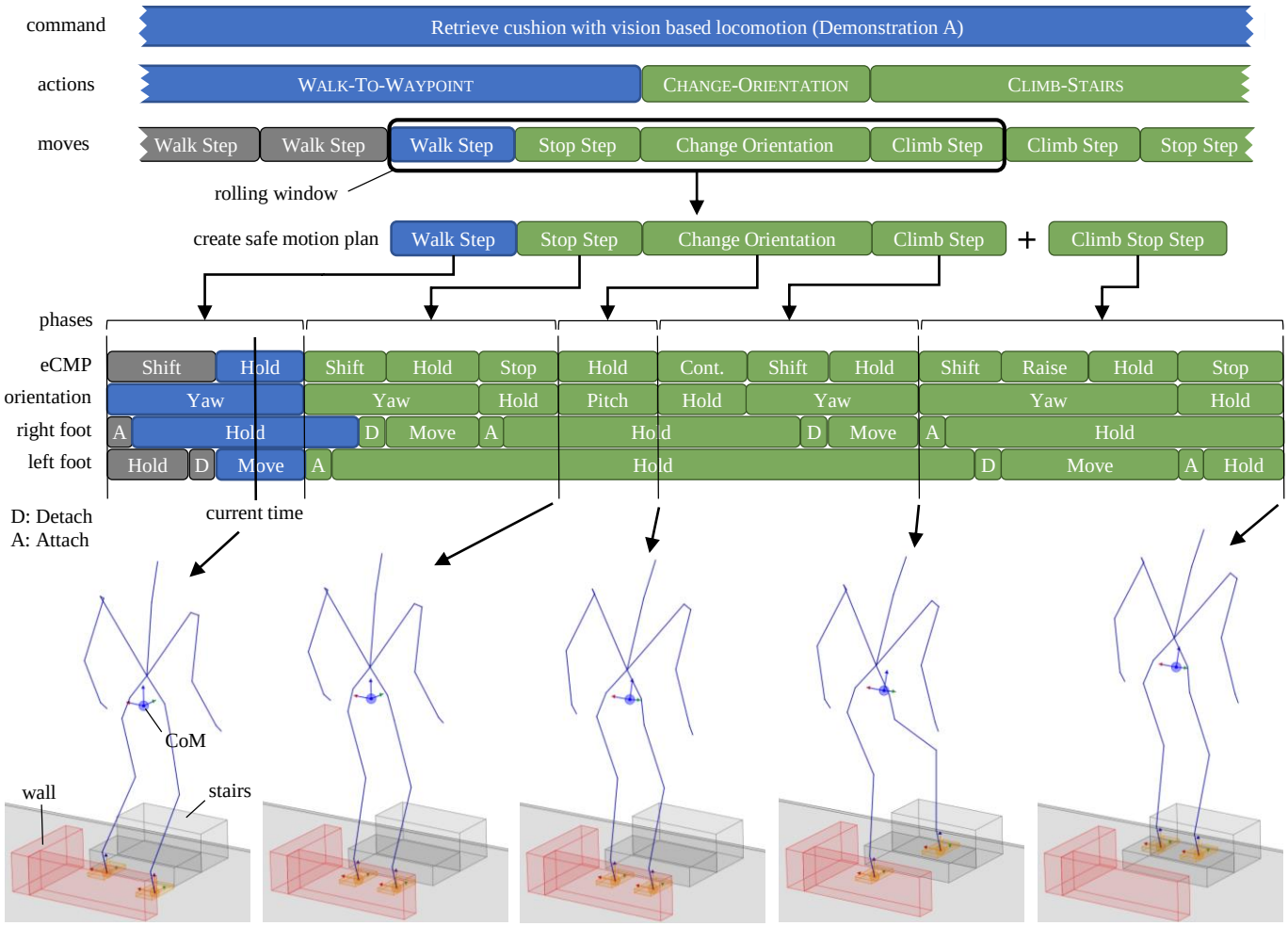

Fig. 7. Motion plan example. Motion elements (actions, moves, phases) that have been executed are depicted in gray, current elements are blue, and future ones are green. Vertically, the figure shows the plan detailing process, starting from the high-level command down to the level of motion phases, which encapsulate individual subtask instructions. The bottom images show schematically reference robot postures at the current time (leftmost image), and at future moments in time (middle three images), with the final image depicting the safe resting posture of the robot in case the plan execution is stopped prematurely.

balance in the current contact configuration. When the tag localization is complete, the plan assembly process resumes normally for the newly generated actions.

In the following, we discuss the subtask plans in more detail and, using Fig. 7 and Fig. 8 as case studies, we explain the effects of each individual phase type. As a general principle, whenever contacts are involved in the motion phase execution, the start and end values are defined relative to the contact poses. For example, the phase "Move $c_1 \rightarrow c_3$" from Fig. 8a instructs the robot to move its left foot from contact c_1 to contact c_3. The actual poses of these two contacts are determined only at execution time, and they can differ from the planned contact poses after being modified by the footstep adjustment module.

1) *CoM subtask*. For the CoM trajectory generation, we use the three-dimensional Divergent Component of Motion (3D-DCM) framework [15], which is a reformulation, without loss of generality, of Newton's second law of motion

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

where $\ddot{\boldsymbol{c}}$ denotes the CoM acceleration, m the robot's total mass, $\boldsymbol{f}_{\text{ext}} \in \mathbb{R}^3$ is the sum of all external forces acting on the robot, and $\boldsymbol{g} = (0\ 0\ -g)^T$, the gravitational acceleration vector. The 3D-DCM framework proposes the substitution of the external force $\boldsymbol{f}_{\text{ext}}$ in equation (1) with a point, called the Enhanced Centroidal Moment Pivot (eCMP). The eCMP $\boldsymbol{e} \in \mathbb{R}^3$ encodes the direction and magnitude of the external force via

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

where b is a time constant defined as $b := \sqrt{\frac{\Delta z}{g}}$, with Δz denoting, for walking, the average CoM height above ground. After introducing two additional points, the Virtual Repellent Point (VRP) as $\boldsymbol{v} := \boldsymbol{e} + (0\ 0\ \Delta z)^T$, and the 3D Divergent Component of Motion (DCM) as $\boldsymbol{\xi} := \boldsymbol{c} + b\,\dot{\boldsymbol{c}}$, the second-order CoM dynamics (1) can be written equivalently as the two first-order dynamics

$$\begin{cases} \dot{\boldsymbol{c}} = -\frac{1}{b}(\boldsymbol{c} - \boldsymbol{\xi}), \\ \dot{\boldsymbol{\xi}} = \frac{1}{b}(\boldsymbol{\xi} - \boldsymbol{v}). \end{cases} \tag{3}$$

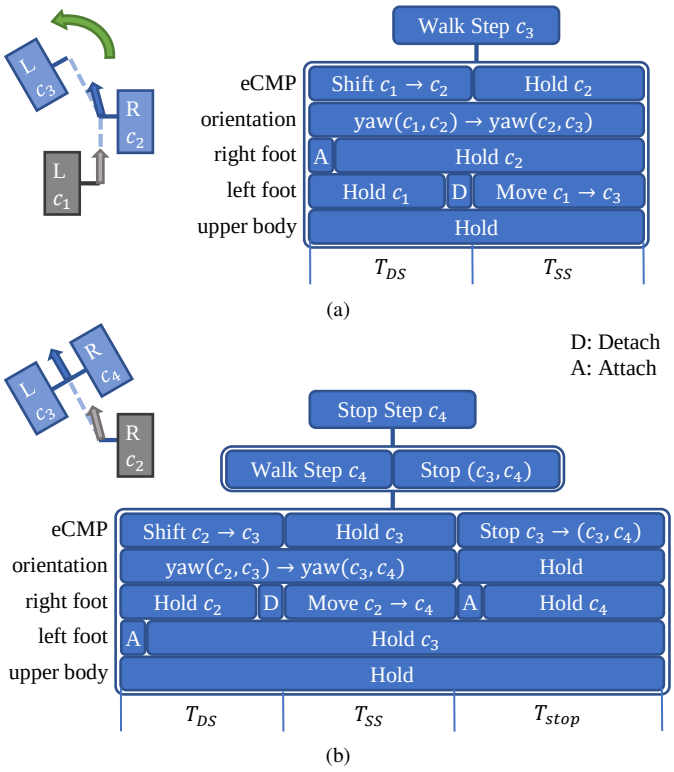

Fig. 8. Two examples of reusable moves, showing the planned footsteps on the left side and the detailed motion plans on the right. (a) A walking step taken with the left leg, using the right leg for support. The double and single support phases are indicated by their durations, T_{DS} and T_{SS}, respectively. (b) The subsequent stopping step taken with the right leg. This step type extends the functionality of the walking step with an additional stopping phase with duration T_{stop}. At the end of the motion, the robot is standing, with its CoM horizontal position located in the middle of the support area formed by the c_3 and c_4 footsteps.

The planning process using the 3D-DCM framework consists of placing a sequence of eCMP waypoints, computing the associated VRPs with the vertical offset Δz given as a parameter, and integrating in closed-form (3) for a piecewise interpolation of the VRP trajectory through the placed waypoints. From a planning point of view, there are two major advantages of using the 3D-DCM framework. First, the planning problem is simplified by directly placing eCMP waypoints onto arbitrary footstep sequences, compared to finding feasible force profiles in the vector space for the same problem. Second, the split into two first-order dynamics enables the combination of reverse-time integration of the DCM dynamics and forward-time integration of the CoM dynamics, leading to bounded CoM trajectories for arbitrary motion durations [14].

The CoM subtask motion phases encapsulate instructions for the eCMP waypoint placement and its corresponding motion during the phase execution. In the motion plan example from Fig. 7, we use the following phase types, presented in the order in which they appear in the plan:

- Shift: moves the eCMP from one footstep to the next using linear interpolation. This phase is usually described as the double support phase of the walking gait, however, the term weight-shift is a more accurate depiction of the performed motion.
- Hold: maintains a constant eCMP position. This phase is typically used for walking during the single support phase, or while balancing and performing other subtasks such as changing the body orientation.
- Stop: moves the eCMP from the last support footstep to a point situated in the middle of the standing support area. At the end of this phase, the DCM and VRP positions coincide, which corresponds to the resting state of the 3D-DCM framework (zero DCM velocity).
- Continuity: computes an intermediate eCMP waypoint that eliminates a discontinuity in the resulting DCM trajectory caused by the reverse-time integration combined with the stationary initial DCM position (for more details, see [13]). This phase is used in the initial state or after a Stop phase has been executed, i.e., whenever the robot starts moving from a standing position.
- Raise: moves the eCMP vertically while maintaining a fixed horizontal position. This motion is typically used during stair climbing for raising the CoM position to the configured height above each stair step.

2) *Body orientation subtask*. The robot's body orientation at the end of each motion phase is given as Euler angles, more specifically using the Tait-Bryan formalization (also called roll-pitch-yaw angles)[1]. Here, we prefer this formulation because its singularity, a pitch angle of $90°$, lies far outside the range of motions commonly performed by the humanoid robot. Moreover, each motion phase can change the angle of a single rotation axis while leaving the others unaffected. For example, crouched locomotion in vertically confined spaces [12] is implemented by increasing the pitch angle once, at the start of the motion, and only adjusting the yaw angle during walking. We use the same method here, in Demonstration A, for climbing the stairs. During walking, the yaw angle is parametrized with the yaw orientations of the right and left footsteps: in Fig. 8a, the term yaw(c_2, c_3) denotes the average yaw angle of footsteps c_2 and c_3. In the motion plan example from Fig. 7, we use the following phase types:

- Yaw: aligns the yaw angle with the current footsteps, as discussed above, using a fifth-order polynomial interpolation from the previous value.
- Hold: maintains a constant orientation.
- Pitch: changes the pitch angle to lean forward or stand upright, using a fifth-order polynomial interpolation.

3) *Limb subtask*. Each limb (arm or leg) has two main modes of operation: "in-contact" and "free". While in-contact, the limb is used to push against the environment, and thus generates an external force that moves the CoM along its reference trajectory. The direction and magnitude of the contact force are computed in realtime by the whole-body controller and depend on the required CoM force, the number of limbs in contact with the environment, and their respective contact constraints (friction cone, center of pressure bounds, etc.). Consequently, we can say that, in this operation mode, the limb's subtask is completely subordinated to the CoM subtask. In contrast,

[1]The Tait-Bryan angles are applied in ZYX order, i.e. z-y'-x"(intrinsic rotations) or x-y-z (extrinsic rotations). The intrinsic rotations are known as yaw, pitch, and roll.

when the limb is free, the position and orientation of the end-effector (foot or hand) can be given as Cartesian quantities, or, alternatively, its configuration can be specified in joint space. Due to the widely different nature of the two operation modes, switching from one to the other requires transition phases that ensure the continuity of the commanded forces [16]. Note that in our presented demonstrations, the legs are always controlled in Cartesian space, while the arms are controlled in joint space.

In the motion plan example from Fig. 7, we use the following limb phase types:

- Attach: establishes a rigid contact with the environment. The limb is in-contact already at the start of this transition phase, however the maximum generable contact force is limited throughout the phase, gradually increasing to its nominal value.
- Hold: maintains the current operation mode. If the limb is free, maintain a constant Cartesian pose or joint configuration.
- Detach: gradually reduces the contact force to zero. The limb is still in-contact throughout the phase, only at the very end of this transition phase, the limb is free.
- Move: moves the foot from one contact pose to the next on a collision-free trajectory. There are different variants of this phase for each walking mode: on flat terrain, a simple ground-clearing motion with a configurable step height is sufficient; for stair climbing and descending, the foot trajectory is designed such that the toe edge does not collide with the stair step.

D. Reference trajectory generator

The motion planner modules presented until this point - the action sequencer, the footstep planner, and the plan assembler - work asynchronously, using an event-based execution principle. Each module reacts to its external input (high-level command or action sequence), computes a result (action sequence, footstep sequence, or motion plan), and then waits for the next event. In contrast, the reference trajectory generator works synchronously, in realtime, at the same execution rate as the whole-body controller. As a consequence, the reference generator has to produce a whole-body task for the controller even in the absence of a motion plan. We identify two cases where this situation can occur.

First, when the reference trajectory generator is started for the first time, it samples the current robot posture (CoM position, body orientation, and joint configuration) and holds this sampled state as a constant whole-body reference. This initial posture is also sent to the action sequencer in preparation for receiving the first command from the human operator. The only assumption made by the motion planner in the initial state is that both feet are in contact with the ground and the initial CoM is positioned such that the robot can maintain its balance indefinitely, i.e. the robot is standing. This default assumption can still be adapted by the human operator to specific scenarios before starting the motion planner and the whole-body controller.

Second, when the motion plan ends, the reference trajectory generator holds the last posture as a constant whole-body reference. As stated before, the final posture for each action is designed such that the robot can maintain it indefinitely (stable balance). As an additional safety feature, the motion plan sent for execution is amended to leave the robot in a safe resting posture in case the plan execution is stopped prematurely due to a communication breakdown.

Finally, we consider the nominal case, where the reference task is computed from a received motion plan. For each subtask, the trajectory generator keeps track of the current move and the local execution time, notifying the plan assembler of the execution progress whenever a move is completed.

Each motion phase contains a single instruction to be performed by the trajectory generator, together with the necessary parameters: the phase duration, the interpolation function, and the subtask-specific goal values or contact references, where applicable. In general, the information contained in the current motion phase is sufficient to compute the instantaneous quantities. However, for the CoM subtask, the trajectory generation uses eCMP waypoints and the reverse-time integration of the DCM trajectory, starting from a DCM resting position at the end of the motion plan. This approach requires the reference generator to determine all DCM waypoints, and, in case of locomotion gaits that contain flight phases, also all CoM waypoints. To address this requirement, we propose in our recent work [13] a highly efficient, realtime capable algorithm for computing the DCM and CoM waypoints for an arbitrary sequence of motion phases. As an example of the execution performance, the algorithm calculates all waypoints for a sequence of 26 CoM motion phases within 25 μs on TORO's computing hardware [8].

A further noteworthy aspect pertains to the implementation of the transitions between the two limb operation modes, in-contact and free. While in-contact, only the contact force is relevant, with the foot reference pose being left unspecified. As a result, the robot's actual foot pose naturally adopts the height (z coordinate) and slope (roll and pitch angles) of the ground under the foot, this contact adaptability being one of the main advantages of the torque-controlled locomotion method. However, once the limb is free, the interpolation algorithm requires an initial reference pose from which to compute the instantaneous reference values. Therefore, at the start of the Detach phase, the reference generator samples and retains the actual foot pose for use as the initial reference value in the subsequent phase, which can be either a Move or Hold phase.

E. Footstep adjustment

The addition of the footstep adjustment module transforms the purely feed-forward motion planner presented so far into a reactive motion planner. In most cases, it is the responsibility of the whole-body controller to react to external disturbances and to correct deviations from the reference trajectory. However, due to actuation limitations as well as the requirement for the robot to react compliantly in human-robot and robot-environment interactions, some tracking errors will remain uncorrected. Combined with the unstable nature of bipedal walking, which is mathematically apparent in the unstable DCM dynamics (3), these tracking errors could lead to the robot falling with potentially dangerous consequences.

Legged locomotion requires the careful synchronization of the CoM trajectory with the feet movement and their respective contact transitions. We have already achieved this synchronization within the plan assembler by referencing the contact poses in all motion phases of the relevant locomotion subtasks (Fig. 8). As a consequence, any runtime changes to the contact poses will continue to produce consistent whole-body trajectories. This approach greatly simplifies the implementation of the reactive module, as it can just modify the planned contacts in response to the unrecoverable tracking errors reported by the controller. We distinguish between two types of correcting reactions performed by the motion planner, with both being introduced in our previous work [17].

First, swing foot tracking errors are interpreted probabilistically to determine the contact placement accuracy. At the start of the swing phase, the likelihood that the foot tracking error will be corrected until contact acquisition is high, but this probability decreases as the phase progresses. Therefore, the planned footstep is gradually adjusted to reflect the likely foot placement, with the final footstep location corresponding to the actual foot pose at the end of the swing phase. We refer to this reaction as the *stumble recovery*, because it adjusts the CoM trajectory in response to swing foot tracking errors.

The second reaction type corresponds to the stepping strategy employed as part of the *push recovery*. In this case, the footstep adjustment is computed from the DCM tracking error that cannot be corrected by the eCMP modulation within the boundaries of the current support area (ankle strategy). In our CoM trajectory generation algorithm, the relation between the instantaneous DCM position and each footstep location is linear, and can be computed efficiently for all stages of the walking gait [17].

Finally, note that one footstep adjustment can lead to additional footsteps being adjusted in order to keep them within their respective placement bounding areas. This aspect was discussed in more detail in the footstep planner section.

IV. Motion Control and Hardware

For the motion execution, we use the passivity-based whole-body torque controller, introduced for balancing tasks in [18], and extended to dynamic contact transitions such as walking, running, or general multi-contact locomotion in [16]. The whole-body controller was described in detail in the referenced publications; here, we present briefly its main components.

State estimation. The tracking performance of the whole-body controller is directly linked to the quality of the state estimation. Our implementation uses a direct, lag-free method of computing the robot's base frame and its velocity using contact information, the robot's kinematic model, as well as measurements of the joint positions, force/torque sensor, and inertial measurement unit (IMU).

Angular momentum (online motion optimization). The motion optimizer uses a subset of the DoF of the whole-body reference trajectories to induce an angular momentum objective resulting from the motion planner [19]. For walking scenarios, the DoF of the upper body are used to regulate the yaw component of the centroidal angular momentum to zero, leading to an arm-swinging motion.

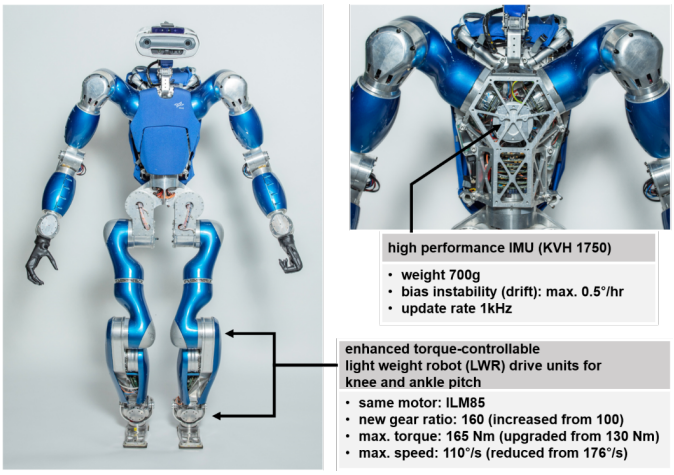

Fig. 9. The humanoid robot TORO, highlighting the hardware improvements made from the previous version presented in [8].

Whole-body controller. The passivity-based whole-body controller [16], [18] implements the whole-body task as a non-strict task hierarchy, creating passive and compliant behavior for each locomotion subtask (CoM location, body orientation, etc.) via virtual spring-damper constructions. The algorithm computes contact forces as the solution of an optimization problem, distributing the CoM wrench to the end-effectors, while taking into account the contact constraints (unilaterality, friction, center of pressure). Furthermore, the compliant feedback controller is combined with a feedforward control element, thereby improving the overall tracking performance.

Low-level torque controller. The joints are modeled as linear mass-spring-damper systems due to their comparably low joint stiffness induced by the strain wave gears. A PD-type torque feedback method is formulated in [20], using the link-side torque measurement to track the commanded joint torques. The joint torque feedback can be interpreted as a shaping of the motor inertia, while the allowable shaping ratio depends mainly on the noise level of the torque sensor. Here, a ratio between 4 and 6 could be achieved, based on our empirical findings. The low-level torque controller at the joints runs at a rate of 3 kHz.

Hardware. Since its debut, DLR's torque-controlled humanoid robot underwent several significant updates, starting with the DLR Biped in 2010, and presenting the full humanoid robot TORO in 2014 [8]. This article presents the latest hardware upgrades (Fig. 9), which focused on increasing the motor torques in the ankle and knee joints, and incorporating an advanced IMU (KVH 1750), leading to a significant enhancement in the quality of the state estimation.

V. Conclusion

In this article, we have examined in detail the inner workings of a practical motion planner, and demonstrated its use in three different challenging scenarios. Note that, while the sequence of actions for Demonstration A has been hand-crafted to achieve the desired task, an automated knowledge-based task planner could use the library of actions provided by the motion planner to infer the task order autonomously.

This extension of the motion planner, as well as increased perception capabilities to navigate the environment without requiring tags, but using for instance a pure Simultaneous Localisation and Mapping (SLAM) approach, are part of our future research efforts.

While the presented motion planner is by no means complete, its extensibility and configurability can be seen as the necessary prerequisites towards the development of a full motion planning framework. We have already taken the first steps in this direction, by extending the motion planning to generate and control more dynamic bipedal locomotion gaits like running and jumping [13], or enhance the walking gait with an angular momentum compensation strategy [19]. Moreover, the planning framework is generic enough to be extendable to quadrupedal locomotion by including additional gait types like trotting, pacing, etc.

Our motion planning framework borrows concepts from the field of software engineering, including abstractions, modularity, and reusability. Its major advantages over holistic methods like Model Predictive Control (MPC) or Reinforcement Learning (RL) lie in the resulting explainability, ease of tuning and adaptation, and in safety and stability guarantees. The latter are based on the availability of analytical solutions for the trajectory generation part, and the logical and consistent structure of the actions, moves, and motion phases. Currently, many of the motion parameters used in our planning framework are empirically tuned to the capabilities of the robot. Therefore, a combination of our motion planner with RL or other optimization techniques carries great potential. This way, motion phases and control parameters, or even complete trajectories or moves, could be optimized and augmented to improve the performance of our planning and control framework even further, while the mentioned safety and stability guarantees can still be upheld.

The motion planning framework proposed here alleviates the load for roboticists to program in detail complex task sequences. Such a framework is required to develop, exploit, and transfer the motion skills among the numerous humanoid robots in development nowadays. In response to the worldwide interest in humanoid robots, we hope that our motion planner can contribute to the rapid advancement of this technology, and can open the door towards new and exciting applications in industrial settings and home environments.

ACKNOWLEDGMENTS

This work was partly supported by ITECH R&D programs of MOTIE/KEIT under Grant 20026194.

REFERENCES

[1] 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 *IEEE International Conference on Robotics and Automation*, 2003, pp. 1620–1626.

[2] O. Stasse, T. Flayols, R. Budhiraja, K. Giraud-Esclasse, J. Carpentier, J. Mirabel, A. Del Prete, P. Souères, N. Mansard, F. Lamiraux *et al.*, "Talos: A new humanoid research platform targeted for industrial applications," in *IEEE-RAS International Conference on Humanoid Robotics*, 2017, pp. 689–695.

[3] J. Lee, J. Kim, A. Alspach, K. Yamane, and C. G. Atkeson, "Can we build Baymax? Part VIII: Let's Talk about Safe, Commercially Viable Humanoids!" https://baymax.org/workshop/2023/, Accessed: 2024-05-06.

[4] Y. Sakagami, R. Watanabe, C. Aoyama, S. Matsunaga, N. Higaki, and K. Fujimura, "The intelligent Asimo: System overview and integration," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, vol. 3, 2002, pp. 2478–2483.

[5] Boston Dynamics, "Atlas," https://www.bostondynamics.com/atlas, Accessed: 2024-05-06.

[6] V. Tsounis, M. Alge, J. Lee, F. Farshidian, and M. Hutter, "Deepgait: Planning and control of quadrupedal gaits using deep reinforcement learning," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 3699–3706, 2020.

[7] J.-P. Sleiman, F. Farshidian, M. V. Minniti, and M. Hutter, "A unified MPC framework for whole-body dynamic locomotion and manipulation," *IEEE Robotics and Automation Letters*, vol. 6, no. 3, pp. 4688–4695, 2021.

[8] J. Englsberger, A. Werner, C. Ott, B. Henze, M. A. Roa, G. Garofalo, R. Burger, A. Beyer, O. Eiberger, K. Schmid, and A. Albu-Schäffer, "Overview of the torque-controlled humanoid robot TORO," in *IEEE-RAS International Conference on Humanoid Robots*, 2014, pp. 916–923.

[9] G. Cheng, S.-H. Hyon, J. Morimoto, A. Ude, J. G. Hale, G. Colvin, W. Scroggin, and S. C. Jacobsen, "CB: A humanoid research platform for exploring neuroscience," *Advanced Robotics*, vol. 21, no. 10, pp. 1097–1114, 2007.

[10] N. G. Tsagarakis, S. Morfey, G. M. Cerda, L. Zhibin, and D. G. Caldwell, "Compliant humanoid coman: Optimal joint stiffness tuning for modal frequency control," in *IEEE International Conference on Robotics and Automation*, 2013, pp. 673–678.

[11] E. Olson, "AprilTag: A robust and flexible visual fiducial system," in *IEEE International Conference on Robotics and Automation*, 2011, pp. 3400–3407.

[12] A. Kheddar, S. Caron, P. Gergondet, A. Comport, A. Tanguy, C. Ott, B. Henze, G. Mesesan, J. Englsberger, M. A. Roa *et al.*, "Humanoid robots in aircraft manufacturing: The Airbus use cases," *IEEE Robotics & Automation Magazine*, vol. 26, no. 4, pp. 30–45, 2019.

[13] 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," *IEEE Transactions on Robotics*, vol. 39, no. 6, pp. 4443–4463, 2023.

[14] G. Mesesan, J. Englsberger, C. Ott, and A. Albu-Schäffer, "Convex properties of center-of-mass trajectories for locomotion based on divergent component of motion," *IEEE Robotics and Automation Letters*, vol. 3, no. 4, pp. 3449–3456, 2018.

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

[16] 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 *IEEE-RAS International Conference on Humanoid Robots*, 2019, pp. 25–32.

[17] G. Mesesan, J. Englsberger, and C. Ott, "Online DCM trajectory adaptation for push and stumble recovery during humanoid locomotion," in *IEEE International Conference on Robotics and Automation*, 2021, pp. 12 780–12 786.

[18] B. Henze, M. A. Roa, and C. Ott, "Passivity-based whole-body balancing for torque-controlled humanoid robots in multi-contact scenarios," *The International Journal of Robotics Research*, vol. 35, no. 12, pp. 1522–1543, 2016.

[19] R. Schuller, G. Mesesan, J. Englsberger, J. Lee, and C. Ott, "Online centroidal angular momentum reference generation and motion optimization for humanoid push recovery," *IEEE Robotics and Automation Letters*, vol. 6, no. 3, pp. 5689–5696, 2021.

[20] A. Albu-Schäffer, S. Haddadin, C. Ott, A. Stemmer, T. Wimböck, and G. Hirzinger, "The DLR lightweight robot: design and control concepts for robots in human environments," *Industrial Robot: an international journal*, vol. 34, no. 5, pp. 376–385, 2007.