Motion planning for quadrupedal locomotion: coupled planning, terrain mapping and whole-body control

Carlos Mastalli[1,2], Ioannis Havoutis[3], Michele Focchi[1], Darwin G. Caldwell[1] and Claudio Semini[1]

***Abstract*—Legged robots promise an advantage over traditional wheeled systems, however, most legged robots are still confined to structured and flat environments. One of the main reasons for this is the difficulty in planning complex whole-body motions while taking into account the terrain conditions. This problem is very high-dimensional as it considers the robot's dynamics together with the terrain model in a suitable problem formulation. In this work, we propose a novel trajectory and foothold optimization method that plans dynamically both foothold locations and motions (coupled planning). It jointly optimizes body motion, step duration and foothold selection, considering the terrain topology. We show that it can be easily generalized to various terrain conditions (i.e. through the terrain costmap), thanks to a parametrized dynamic model, and an online terrain mapping that is used in our real-time whole-body controller. Our whole-body controller tracks compliantly trunk motions while avoiding slippage, as well as kinematic and torque limits. For the sake of analysis, we compare our coupled planner with our previous decoupled planner. With this novel locomotion framework we can cross a wider range of terrain conditions. We report thorough experimental results and comparative evaluations over a set of terrains of progressively increasing difficulty.**

***Index Terms*—trajectory optimization, legged robots, challenging locomotion, whole-body control and terrain mapping**

I. Introduction

Agile locomotion is a key ability for navigating challenging environments. Wheeled or tracked vehicles are efficient in structured environments, for example on flats, roads and paths, but can suffer from limited mobility in many real-world scenarios. Legged locomotion can deliver substantial advantages in real-world environments as it can offer a degree of mobility that is unmatched by the wheeled counterparts. Legged platforms decouple the trajectory of the robot body from its support area. This way, the support areas or points, i.e. the footholds that the legged robot needs to achieve for successfully navigating the terrain, are reduced dramatically and are typically discontinuous.

Due to these characteristics, legged robots offer a clear advantage in unstructured and challenging terrain. Such environments are common in disaster relief, search and rescue, forestry and construction site scenarios. Nonetheless, most legged robots are still confined to structured and flat terrain despite the significant efforts of the research community. The main reason for this is due to the difficulty of generating complex dynamic motions that allow them to cross a vast majority of terrain conditions. Many legged locomotion approaches have focused on the study of reactive behaviors for robot stability without considering the terrain conditions (i.e. "blind" locomotion). A reactive behavior (or motion control) is an instantaneous action that aims to immediately stabilize the robot; it does not consider a horizon of future events. These approaches can only tackle small changes in the terrain topology, and furthermore, they cannot always guarantee the successful accomplishment of the task. Such difficulties have restricted the use of legged systems to controlled environments and research platforms.

Recently, trajectory optimization with contacts gained much attention in the legged robotics community. It aims to overcome the previously mentioned drawbacks of reactive locomotion approaches by considering a horizon of future events (e.g. body movements and foothold locations). For example, it could potentially improve the robot stability along a specific planning horizon given a certain terrain. In spite of the promising benefits of trajectory optimization for rough terrain locomotion, most of the works are focused either on flat conditions or on simulation. Conversely, in rough terrain locomotion, the foothold locations and motions have to be carefully planned.

To ensure locomotion stability, the robot needs to "understand" the environment through a perception system. The terrain modeling serves to quantify the terrain difficulty and uncertainty, so that, the robot can plan foothold locations and movements. The terrain model can be used in two ways: for foothold selection and for foothold interaction[1]. Next, the robot has to evaluate different possible body motions and foothold locations. Robot modeling helps to capture the fundamental dynamics, while reducing an unnecessary set of robot behaviors (i.e. the search space). However, current literature is missing a rigorous study of different motion planning methods for challenging locomotion. For that, we introduced four benchmark terrains to compare various planning methods.

[1]Department of Advanced Robotics, Istituto Italiano di Tecnologia, Genova, Italy. *email*: {carlos.mastalli, michele.focchi, darwin.caldwell, claudio.semini}@iit.it.
[2]CNRS, LAAS, University of Toulouse, France. *email*: carlos.mastalli@laas.fr.
[3]Oxford Robotics Institute, Department of Engineering Science, University of Oxford, United Kingdom. *email*: ihavoutis@robots.ox.ac.uk.

[1]With foothold interaction, we refer to dynamic feasibility, i.e. computation of GRFs and friction cones.

A. Contribution

The main contributions of this work are: first, an exhaustive comparison between the coupled and decoupled planning methods using our test-case planners; second, an extension of our coupled planning framework by exploiting the terrain normals for real-time whole-body control; third, the development of a novel whole-body controller formulation that considers the friction constraints, the robot's kinematic and torque limits. Our new whole-body controller avoids slippage through friction cone constraints that are imposed in real-time using the estimation of terrain normals (i.e. from the terrain mapping). It is designed to track compliantly fast trunk-motions since it incorporates the full dynamics, the kinematic and torque limits of the robot. Furthermore, it combines feedback and feedforward control of swing motions that improves the leg motion tracking performances, important for climbing stairs. On the other hand, for the comparison, we use a set of metrics: the number of footholds, the averaged trunk velocity and the Mechanical Cost of Transport (MCoT). We present an in depth comparison against planners developed in previous work [1, 2]. This article is an extension of our previous works [3] presented at the IEEE International Conference on Robotics and Automation (ICRA) 2017.

The rest of the paper is structured as follows: after discussing previous research in the field of dynamic quadrupedal whole-body locomotion (Section II) we briefly describe our decoupled planner method, which we use for comparison. Later, we explain our terrain mapping algorithm used for planning and control (Section III). Next, we describe our coupled planning method (Section IV). Section V introduces a new controller designed for dynamic motions that considers the friction cone constraints, the robot's full dynamics, kinematic and torque limits. This controller improves the tracking performance and robustness of the locomotion by tracking compliantly desired trunk motions. In Section VI,VII we evaluate the performance of our decoupled and coupled planners on the Hydraulically actuated Quadruped (HyQ), in real-world experimental trials and simulations. Last Section VIII summarizes this work and presents ideas and directions for future work.

II. Related work

One of the earlier approaches to legged locomotion behaviors is statically stable walking. During statically stable walking, locomotion is performed by keeping the robots's Center of Mass (CoM) inside the polygon formed by its supporting feet. It was first identified by [4] and mathematically evaluated by [5]. This work was later extended to facilitate walking over irregular terrain [6].

In environments where smooth, continuous support is available (flats, fields, roads, etc.), exact foot placement is not crucial for the success of the behavior, legged systems can utilize a variety of dynamic gaits; some recent works are, e.g. trotting, galloping [7] and bounding [8, 9]. The work of Marc Raibert crystallized the principles of dynamic locomotion and balancing with legged robots [10]. The *BigDog* and *LS3* quadrupeds are a recent extension of this work. While *BigDog* is able to traverse irregular terrain using a reactive controller, the footholds are not planned in advance. Similar performance can be seen on the HyQ robot, that is able to overcome obstacles with reactive controllers [11, 12] or step reflexes [13, 14].

In contrast, environments with complex geometry, e.g. with obstacles like large gaps, stairs or rubble, such systems quickly reach their limits (i.e. torque limits). Such terrains often afford only a few possible discrete footholds, and there legged robots can employ a range of typically non-gaited locomotion strategies that rely more on accurate foothold planning, and consequentially on features of the terrain. In this case, higher level motion planning is required, that considers the environment geometry and carefully selects appropriate footholds.

The DARPA Learning Locomotion Challenge stimulated the development of footstep planning over rough terrain. It resulted in a number of successful control architectures [15, 16, 17, 18, 19, 20] to plan and execute footsteps to traverse challenging terrain. Rebula et al. [15] avoids global footstep planning by simply choosing the next best reachable foothold. This can cause the robot to locally navigate into an insurmountable obstacle. To avoid this, some methods [21, 19] *globally* plan the complete footsteps from start to goal, though in this case, a time-consuming replanning is necessary when slippage or deviation from the planned path occurs. The approach in [18] stands between the two above mentioned methods and plans a global rough body path to avoid local minima, but the specific footholds are chosen only a few steps in advance. This reduces the necessary time for replanning in case of slippage, while still considering a locally optimal plan. Pongas et al. [16] focused mainly on generating a smooth CoM trajectory, independent of the foothold pattern. Recently, Deits and Tedrake [22] introduced an efficient method, formulated as Mixed-Integer Convex Programming (MICP), to plan a sequence of footholds. This approach has been extended to quadrupedal locomotion on challenging terrain [23, 24]. However, using integer variables requires a convex model of the terrain, which lose validity for significant non-linear curvature of the terrain.

Natural locomotion over challenging terrain requires simultaneous computation of footstep sequences, body movements and gait transitions (coupled planning) [e.g. 25, 26, 27, 28]. One of the main problems with such approaches is that the search space quickly grows and computation time becomes impractical, especially for systems that need solutions in real-time. In contrast, we can split the planning and control problem into a set of sub-problems, following a decoupled planning strategy. For example the body path planner and the footstep planner can be separated, thus reducing the search space for each component [e.g. 17, 21, 29, 2]. This can reduce the computation time at the expense of limiting the planning capabilities of the robot, sometimes required for extreme challenging terrain. There are two main approaches of decoupled planning: *contact-before-motion* [e.g. 30, 31, 21] and *motion-before-contact* [e.g. 32, 17, 22]. These approaches find a solution in motion space, which defines the possible motion of the robot, the former first find the set of footholds to be achieved and then generate the desired motion, while the

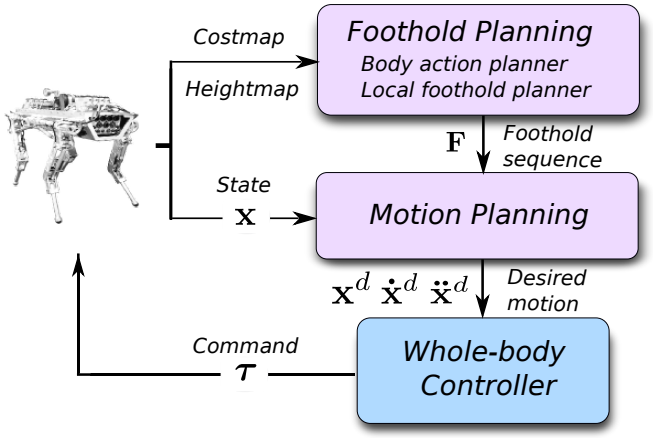


Fig. 1: Overview of our decoupled motion and foothold planning framework [2, 1]. The foothold planner first computes a sequence of body action and then selects the foothold locations $\mathbf{F}$ around an action-specific foothold region. The foothold sequence is planned according the terrain information (i.e. terrain costmap and heightmap). Then the motion planner uses the planned footholds to generate dynamic whole-body motions $(\mathbf{x}^d, \dot{\mathbf{x}}^d, \ddot{\mathbf{x}}^d)$. Finally, the desired motion is compliantly executed by using a combination of feed-forward and feedback terms. (Figure modified from [33])

latter work the opposite way.

A. Decoupled planning

In our previous works [1, 2] we proposed a locomotion framework based on a decoupled planning strategy. First, we planned a sequence of footholds by planning an approximate body path. The approximate body path was computed from a sequence of planned body actions. Then, we chose locally the locations of the footholds. Finally, we generated a body trajectory that ensured dynamic stability and achieved the planned foothold sequence. Fig. 1 shows an overview of our decoupled motion and foothold planning framework for dynamic legged locomotion over challenging terrain.

The overall task was to plan online an appropriate sequence of footholds $\mathbf{F}$ that allows the robot to traverse a challenging terrain toward a body goal state (x, y, θ). To accomplish this, our foothold planner first computed a sequence of body action and then selected the foothold locations around an action-specific foothold region. This generated a bounded sub-optimal body path, through a sequence of body actions, in a growing *body-state graph*. The body-state graph used the explored action to select an appropriate foothold region. We used this region to compute the body cost, or transition cost between two nodes in the graph.

Once a sequence of footsteps was computed, we planned a CoM motion that ensured the dynamic stability for those steps. We used two fifth-order polynomials to describe the horizontal CoM motion. Furthermore we used a cart-table model to estimate the Center of Pressure (CoP) position, and keep it inside the support polygon. For more details the reader can refer to [1, 2].

III. Terrain mapping

This section describes the pipeline of acquisition and evaluation of terrain information. We implemented an onboard terrain information server that holds and continuously updates the state of the environment. We show how this information is processed and transformed to a qualitative metric of the terrain topology.

An occupancy map holds the 3D geometric perception data scanned from vision sensors mounted at the front of the robot (see Fig. 2). For that, we use Octomap[2] as this provides a probabilistic representation that handles sensor noise. Octomap represents both free and occupied spaces and satisfies the required computation time for our application, which is at least 2 Hz with onboard processing (see Section VI for details about the onboard PCs). Octomap uses a hierarchical data structure, for spatial subdivision in 3D, called *octrees*. This octree-based representation is designed to efficiently update and copy the map [for more details see 34]. Moreover, it has a multi-resolution volumetric representation that we use to speed up the computation time of the geometric features such as *slope* and *curvature*.

A. Terrain costmap

The *terrain costmap* quantifies how desirable it is to place a foot at a specific location. The cost value for each voxel in the map is computed using geometric terrain features such as *height deviation*, *slope* and *curvature* [similar to 35]. We compute the slope and curvature through regression in a 6 cm×6 cm window around the cell in question; the features are computed from a voxel model (2 cm voxel-size resolution) of the terrain. For instance, the estimated surface normals and curvatures are computed from a set of neighboring occupied voxels. We estimate the surface normals through an analysis of

[2]The source code is available on https://github.com/OctoMap/octomap.

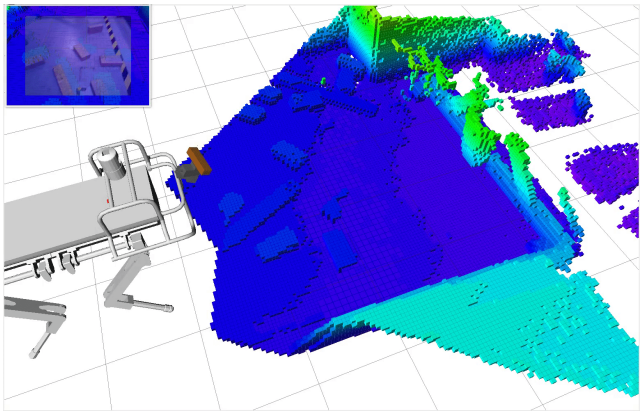


Fig. 2: The HyQ robot mapping the terrain using Octomap [34]. The voxel map is generated from RGBD camera data (Asus Xtion), using the estimated body position. The RGBD sensor is mounted on a Pan and Tilt Unit (PTU) that scans the terrain, with a left/right sweep and up/down frequency of 1 Hz. The occupancy map is built with a 2 cm resolution. To watch the video, click the figure.

the eigenvectors and eigenvalues, a procedure using Principal Component Analysis (PCA), of the set of nearest neighbors (for more information, including the mathematical equations of the least-squares problem, see [36]). Each surface normal is computed from the eigenvector that has the smallest eigenvalue, λ_0 and the surface curvature σ as follows:

$$\sigma = \frac{\lambda_0}{\lambda_0 + \lambda_1 + \lambda_2} \tag{1}$$

Fig. 3 shows a set of estimated surface normals from the occupancy map of a cobblestone terrain. We compute the normals, and other geometric features, with 2 cm of resolution (i.e. costmap resolution). We build the terrain costmap on top of the occupancy map; therefore, the terrain map has its own voxel-size resolution. For all terrain costmaps in this article we use a resolution of 2 cm.

The terrain costmap is incrementally built based on the aforementioned features and updated locally whenever a change in the map is detected. For computing the terrain costmap, we define an area of interest around the robot of 2.5 m×5.5 m. For each pixel of the terrain map, the cost value is computed as a weighted linear combination of the individual features $T(x, y) = \mathbf{w}^T\mathbf{T}(x, y)$, where $\mathbf{w}$ and $\mathbf{T}(x, y)$ are the weights and feature cost values, respectively. The total cost value is normalized, where 0 and 1 are the minimum and maximum cost values, respectively. The weight vector describes the importance of the different features. We will now give a brief overview of the different geometry features used for computing the terrain cost values.

1) Height deviation

The height deviation cost penalizes footholds close to large drop-offs; for instance, this cost is needed for crossing gaps or stepping stones. In fact, staying far away from large drop-offs is beneficial because inaccuracies in the execution of footsteps can cause the robot to step into gaps or banned areas. For example, the HyQ robot foothold inaccuracies are around 4 cm (or 2 voxels) when slippage does not occur; note that slippage events can increase the execution inaccuracies[3]. In addition, the height deviation cost also keeps the feet away from areas in which the shin can collide with the environment, similar as in [2]. This is useful in climbing up/down stairs. The height deviation feature f_h is computed using the standard deviation around a defined neighborhood.

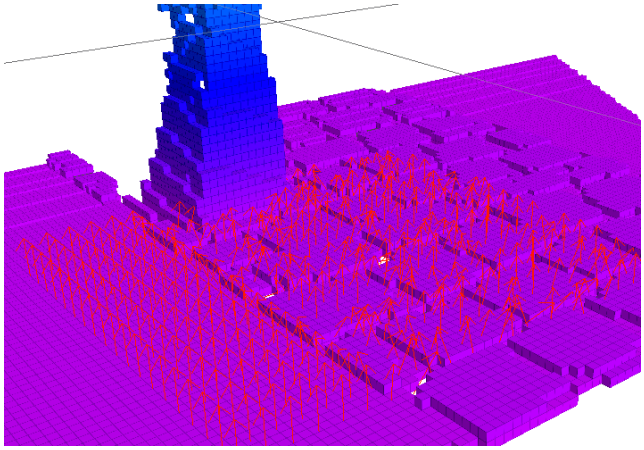


Fig. 3: A set of surface normals are extracted from the RGBD sensor. The surface normals are estimated from the eigenvalues and eigenvectors computed from the nearest neighbors of a query point. To watch the video, click the figure.

TABLE I: Parameter values of the height deviation and slope cost functions. We use the same values for all the experiments presented in this work.

	f_{flat}	$f_{\max}$	$T_{\max}$
height deviation	0.01	0.06	1
slope	$\frac{\pi}{180}$	$\frac{70\pi}{180}$	1

2) Slope

The slope reflects the local surface normal in a neighborhood around the cell, the normals are computed using PCA on the set of nearest neighbors. A high slope value will increase the chance of slipping even in cases where the friction cone is considered, e.g. due to inaccuracies in the friction coefficient or estimated surface normal. Slope cost increases for larger slope values, while small slopes have zero cost as they are approximately flat. We consider the worst possible slope s_{max} occurs when the terrain is very steep (approximately 70°)[4].

We map the height deviation and slope features into cost values through the following piecewise function:

$$T_f(x, y) = \begin{cases} 0 & f \leq f_{flat} \\ -\ln\left(1 - \frac{f(x,y)}{f_{max} - f_{flat}}\right) & f_{flat} < f < f_{max} \\ T_{max} & f \geq f_{max} \end{cases}$$

where f_{flat} is a threshold that defines the flat conditions, f_{max} the maximum allowed feature value, and T_{max} is the maximum cost value. In Table I we report the used values for all the experiments.

3) Curvature

The curvature describes the stability of a given foothold location. For instance, terrain with mild curvature (curvature between $c = 6$ to $c = 9$) is preferable to flat terrain since it reduces the possibility of slipping, as it has a bowl-like structure. Thus, the cost is equal to zero in those conditions. On the other hand, high and low curvature values represent a narrow crack structure in which the foot can get stuck in ($c > 9$) or edge structure in which the foot can slip ($c < -6$), respectively; in those conditions, we have a higher cost value. We use the following piecewise function to compute the cost value from a curvature value c:

$$T_c(x, y) = \begin{cases} 0 & c^+_{mild} > c > c^-_{mild} \\ T_{max} & c \leq c_{crack} \\ & c \geq c^+_{mild} \\ T_{max} - \ln\left(\frac{c(x,y) - c_{low}}{c_{high} - c_{low}}\right) & c_{crak} < c < c^-_{mild} \end{cases}$$

where $c_{crack} = -6$, $c^-_{mild} = 6$, $c^+_{mild} = 9$ and $c_{\max} = 9$.

[3]Slippages produce a drift in the estimated body position, for more details see [37].

[4]We heuristically defined this value based on our experience with the HyQ robot.

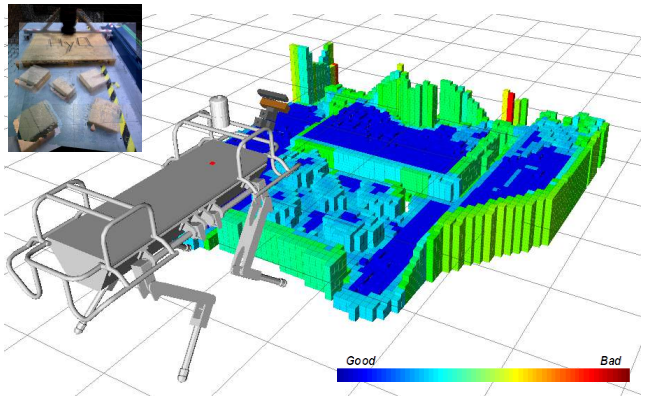


Fig. 4: A costmap generated from RGBD data. The occupancy map is built with a 2 cm resolution. Then the set of features is computed and the total cost value per voxel is calculated. In addition, a heightmap is created with a resolution of 2 cm in z. The cost values are represented using a color scale, where blue is the minimum cost and red is the maximum. (Figure taken from [2])

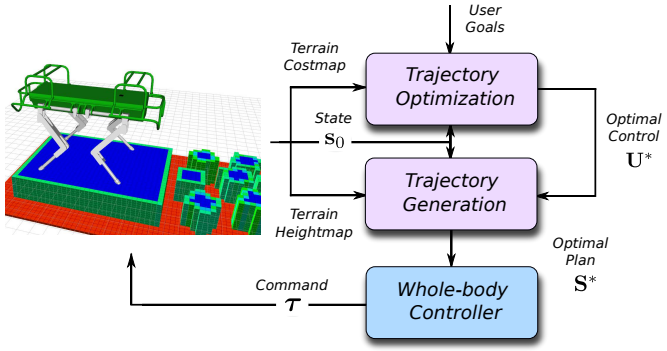


Fig. 5: Overview of our coupled motion and foothold planning framework for locomotion on challenging terrain [3]. We compute offline an optimal sequence of control parameters $\mathbf{U}^*$ given the user's goals, the actual state $\mathbf{s}_0$ and the terrain costmap. Given this optimal control sequence, we generate the optimal plan $\mathbf{S}^*$, that uses trunk attitude planning to adapt to the changes in the terrain elevation. Lastly, the whole-body controller calculates the joint torques $\boldsymbol{\tau}^*$ that satisfy friction-cone constraints. (Figure taken from [3])

Fig. 4 shows the computation of the terrain costmap, using the above-mentioned mapping between geometry features to cost values, from the onboard RGBD sensor. The cost values are represented using a color scale, where blue is the minimum cost and red is the maximum one.

B. Terrain heightmap

The terrain *heightmap* allows the robot to define the vertical component (i.e. z-direction) of the footstep. Indeed the trunk/swing trajectory can be approximately planned to place the foot at the correct height. A higher resolution of the heightmap helps the robot to accurately establish a footstep, which improves the overall execution. The heightmap is computed using the same resolution (2 cm) as the costmap in the (x, y)-plane. Along the z-direction we desire higher accuracy to step safely onto obstacles, so we use a resolution of 1 cm. In essence, the heightmap is a $2^1/_2$-dimensional *projection* of the costmap that can be more efficiently handled by the subsequent steps of the motion and foothold planning. For instance, the terrain costmap associates the cost value to the highest occupied voxel.

IV. Coupled Motion and Foothold Planning

In this section, we address the locomotion as a coupled planning problem of CoM motions and footholds, where the foothold locations are selected using a terrain costmap while the trunk height and attitude are adapted for different terrain elevations (see Fig. 5). First, we jointly generate the CoM trajectory and the swing-leg trajectory using a sequence of parametric preview models and the terrain elevation map (Section IV-A). Then, we optimize a sequence of control parameters (the CoP displacement, the phase duration and the foothold locations) given the terrain costmap (Section IV-B). Compared with our previous work [3], we have reformulated the terrain model in our optimization problem as duality cost-constraint. This terrain model allows us to navigate in various terrain conditions without the needed of re-tuning. Additionally, we have improved our whole-body controller which is able to avoid slippage using an online terrain map (for more details see Section V). Note that coupled planning allows us also to optimize step timing and to use the dynamics for foothold selection which is not possible for decoupled planners, such as our above-mentioned planning method.

A. Trajectory generation

This section describes the low-dimensional trajectory generation for a sequence of control parameters and a given terrain heightmap. We generate the horizontal CoM trajectory[5] and the 2D foothold locations using a sequence of low-dimensional preview models. In order to adapt to changing terrain elevation, we modulate the trunk attitude and height using an estimate of the support plane, and the maximum allowed angular accelerations of the trunk (for more details see Section IV-A1b). We describe the sequence of control parameters w.r.t. the horizontal frame, which allows us to decouple the CoM and trunk attitude planning.

1) Preview model

Preview models are low-dimensional representations that describe and capture different locomotion behaviors, such as walking and trotting, and provide an overview of the motion [38]. With a reduced model we can still generate complex locomotion behaviors and their transitions; furthermore, we can integrate it with reactive control techniques. This is more suitable for challenging terrain as it simplifies the optimization problem landscape. In the literature, different models that capture the legged locomotion dynamics such as point-mass, inverted pendulum, cart-table, or contact wrench have been studied by [39, 40].

[5]The horizontal frame has been using too for reactive motion generation [see 11].

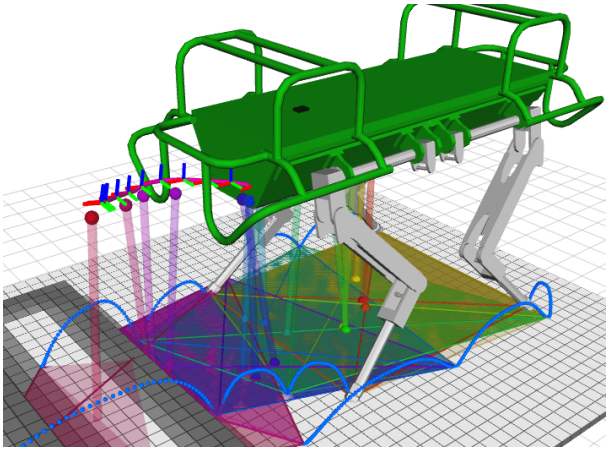


Fig. 6: A trajectory obtained from a low-dimensional model given a sequence of optimized control parameters and the terrain heightmap. The colored spheres represent the CoM and CoP positions of the terminal states of each motion phase. The CoP spheres lie inside the support polygon (same color is used). Note that color indicates the phase (from yellow to red). The trunk adaptation is based on the estimated support planes in each phase. Since the control parameters are expressed in the horizontal frame, the horizontal CoM trajectories and the trunk attitude are decoupled. (Figure taken from [3])

Our preview model decouples the CoM motion from the trunk attitude[6] (Fig. 6). For the CoM motion, we use the cart-table template [41]. The cart-table model encompasses a point mass assumption which has no angular momentum. However, to control the attitude we need to apply moments to the robot's CoM. High centroidal moments (e.g. due to high trunk angular acceleration) can hamper the postural stability condition (e.g. causing shifts on the CoP that can move it out of the support polygon [42], making the robot losing its capability to balance. Consequently, for the attitude planning, we limit the maximum moments applied to the CoM by limiting the maximum angular acceleration and setting a corresponding margin for the CoP on the support polygon.

a) CoM motion:

In our previous work [1], for fixed step durations, we showed that the CoP movement is approximately linear, i.e.:

$$\mathbf{p}^H(t) = \mathbf{p}_0^H + \frac{\delta \mathbf{p}^H}{T} t. \tag{2}$$

Note that $\mathbf{p}^H = (x^H, y^H) \in R^2$ is the horizontal CoP position, $\delta \mathbf{p}^H \in R^2$ the horizontal CoP displacement and T is the phase duration. The $(\cdot)^H$ apex means that the vectors are expressed in the horizontal frame.

Applying this linear control law in the cart-table model, we derive an analytic solution for the horizontal dynamics [38]:

$$\mathbf{x}^H(t) = \boldsymbol{\beta}_1 e^{\omega t} + \boldsymbol{\beta}_2 e^{-\omega t} + \mathbf{p}_0^H + \frac{\delta \mathbf{p}^H}{T} t, \tag{3}$$

where the model coefficients $\boldsymbol{\beta}_{1,2} \in R^2$ depend on the actual state $\mathbf{s}_0$ (horizontal CoM position $\mathbf{x}_0^H \in R^2$ and velocity $\dot{\mathbf{x}}_0^H \in R^2$, and CoP position), the trunk height h, the phase duration, and the horizontal CoP displacement:

$$\boldsymbol{\beta}_1 = (\mathbf{x}_0^H - \mathbf{p}_0^H)/2 + (\dot{\mathbf{x}}_0^H T - \delta \mathbf{p}^H)/(2\omega T),$$

$$\boldsymbol{\beta}_2 = (\mathbf{x}_0^H - \mathbf{p}_0^H)/2 - (\dot{\mathbf{x}}_0^H T - \delta \mathbf{p}^H)/(2\omega T),$$

where $\omega = \sqrt{g/h}$ and g is the gravity acceleration.

b) Trunk attitude:

A trunk attitude modulation is required when the terrain elevation varies. A simple approach consists of aligning the trunk with respect to the estimated support plane, avoiding that the robot reaches its kinematic limits. On the other hand, adjusting the trunk attitude requires applying a moment at the CoM, and as a consequence, the CoP $\mathbf{p} \in R^3$ will be shifted by a proportional amount $\Delta \mathbf{p}$ (for more details see [42]):

$$\begin{aligned} \Delta p_x &= -\tau_{com_y}/mg, \\ \Delta p_y &= \tau_{com_x}/mg, \end{aligned} \tag{4}$$

where τ_{com_y}, τ_{com_x} are the horizontal components of the moment about the CoM. By exploiting a simplified flywheel model for the inertia of the robot we can link these moments to the CoP displacement $\Delta \mathbf{p}$ (rewritten in vectorial form) and to the angular acceleration $\dot{\boldsymbol{\omega}}$:

$$\boldsymbol{\tau}_{com} = \mathcal{I} \dot{\boldsymbol{\omega}}, \tag{5}$$

$$\Delta \mathbf{p} = \boldsymbol{\tau}_{com} \times m\mathbf{g}. \tag{6}$$

where $\mathcal{I} \in R^{3\times3}$ is the time-invariant inertial tensor approximation of the centroidal inertia matrix of the robot. Therefore, we can guarantee the CoP condition by limiting the angular accelerations $\dot{\boldsymbol{\omega}}_{max}$ (i.e. the allowed applied moments) and setting a corresponding safety margin r on the support polygon in our optimization (Section IV-B3) as:

$$r = \|(\mathcal{I} \dot{\boldsymbol{\omega}}_{max}) \times m\mathbf{g}\|. \tag{7}$$

Therefore, we adapt the trunk attitude in such a way that it does not affect the CoP condition (i.e. by using the maximum allowed angular acceleration $\dot{\boldsymbol{\omega}}_{max} \in R^2$). We compute the maximum angular acceleration in frontal and transverse plane (i.e. $\dot{\omega}_x$ and $\dot{\omega}_y$) give the stability vector $\mathbf{r} \in R^2$. Note that we compute it from the stability margin r, i.e. the support polygon margin.

We employ cubic polynomial splines to describe the trunk attitude motion (frontal and transverse). The attitude adaptation can be done in different phases. For instance, we can compute the required angular displacement given the phase duration and guarantee that it does not exceed the allowed angular accelerations. The trunk height is computed given the estimated support plane and we keep it constant along each phase.

2) Preview schedule

Describing legged locomotion can be achieved through a sequence of different preview models — a preview schedule. Using this, the robot can automatically discover different foothold sequences by enabling or disabling different phases in our optimization process.

[6]In this work, with *trunk attitude* we refer to roll and pitch only.

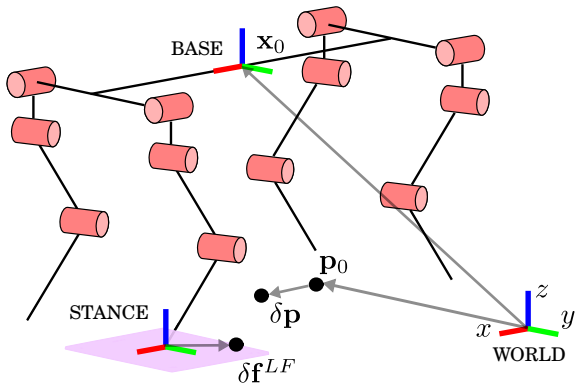


Fig. 7: Sketch of different variables and frames used in our optimization. The footshift $\delta\mathbf{f}^{LF}$ is described w.r.t. the stance frame, its boundary values are defined by the foothold region (the pink rectangle). The stance frame is calculated from the default posture and expressed w.r.t. the base frame. (Figure modified from [3])

In the preview schedule, we build up a sequence of control parameters that describes the locomotion action of the n phases:

$$\mathbf{U} = \begin{bmatrix} \mathbf{u}_1^{st/sw} & \cdots & \mathbf{u}_n^{st/sw} \end{bmatrix}, \tag{8}$$

where $\mathbf{u}_i^{st} = \begin{bmatrix} T & \delta\mathbf{p}^H \end{bmatrix}$ and $\mathbf{u}_i^{sw} = \begin{bmatrix} T & \delta\mathbf{p}^H & \delta\mathbf{f}^l \end{bmatrix}$ are the preview control parameters for the stance and swing phases, respectively. Additionally, the footshift $\delta\mathbf{f}^l$ is described w.r.t. the stance frame (Fig. 7), which is calculated from the default posture of the robot. Note that n is the number of phases, and l is the foot index.

We describe a dynamic walking gait as a combination of 6 different preview phases or timeslots (i.e. $n = 6$) where 4 of them are swing phases. Our combination of phases is stance, Left-Hind (LH) swing phase, Left-Front (LF) swing phase, stance, Right-Hind (RH) swing phase and Right-Front (RF) swing phase[7]. With this fixed preview schedule, we can describe different walking patterns, e.g. by assigning a zero duration to different phases ($T_i = 0$).

B. Trajectory optimization

The trajectory optimization step computes an optimal sequence of control parameters $\mathbf{U}^*$ used for the generation of the low-dimensional trajectories (Section IV-A). We formulate this as a receding horizon trajectory optimization problem, where the current timeslot is optimized while taking future timeslots into account. The horizon is described by a predefined number of preview schedules N with n timeslots or phases (e.g. our locomotion cycle has 6 timeslots). Considering future phases presents several advantages for challenging terrain locomotion. It enables us to generate desired behaviors that anticipate future terrain conditions, and it results in smoother transitions between phases.

In our approach, the optimal solution at the current phase i comprises of a set of control parameters $\mathbf{u}_i^*$ describing the duration of phase T_i^*, the CoP displacement $\delta\mathbf{p}^{H_i^*}$, and the footshift $\delta\mathbf{f}_i^*$ of the corresponding phase. We define the footshift in the nominal stance frame which corresponds to the default posture. Note that there are phases without foot swing. To the best of our knowledge, our approach is the first that jointly optimizes phase duration and foothold selection, while considering terrain conditions. This contribution has been presented in [3].

1) Receding horizon planning

Given an initial state $\mathbf{s}_0$, we optimize a sequence of control parameters inside a predefined horizon, and apply the optimal control of the current phase. We find the sequence of control parameters $\mathbf{U}^*$, through an unconstrained optimization problem, given the desired user commands (trunk velocities):

$$\mathbf{U}^* = \underset{\mathbf{U}}{\operatorname{argmin}} \sum_j \omega_j g_j(\mathbf{S}(\mathbf{U})), \tag{9}$$

where $\mathbf{S} = \begin{bmatrix} \mathbf{s}_1 & \cdots & \mathbf{s}_{Nn} \end{bmatrix}$ is the sequence of preview states. The preview state is defined by the CoM position and velocity $(\mathbf{x}, \dot{\mathbf{x}})$, CoP position $\mathbf{p}$ and the stance support region $\boldsymbol{\sigma}$, i.e. $\mathbf{s} = \begin{bmatrix} \mathbf{x} & \dot{\mathbf{x}} & \mathbf{p} & \boldsymbol{\sigma} \end{bmatrix}$, where $\boldsymbol{\sigma} = \begin{bmatrix} \mathbf{f}_1 & \cdots & \mathbf{f}_j \end{bmatrix}$ is defined by the position of the active feet $\mathbf{f}_j \in R^2$. We solve the trajectory optimization using the Covariance Matrix Adaptation Evolution Strategy (CMA-ES) [43]. CMA-ES is capable of handling optimization problems that have multiple local minima, such as those introduced by the costmap (see Section III-A and Eq. (12)) and the phase duration (see Eq. (3)). In the description of our optimization problem, we use soft-constraints as these provide the required freedom to search in the landscape of our optimization problem. The cost functions or soft-constraints $g_i(\mathbf{S})$ describe: *1*) the user command as desired walking velocity and travel direction, *2*) the CoM energy, *3*) the terrain cost, *4*) a soft-constraint to ensure stability, i.e. the CoP condition, and *5*) the preview model soft-constraint, which is required due to the decoupling of the horizontal and vertical dynamics.

2) Cost functions

We encode the desired body velocity from the user by mapping it into an average walking velocity. Additionally, the CoM trajectory should accelerate as little as possible during the phases. Note that this implicitly reduces the required joint torques. We evaluate the desired velocity command for the entire planning horizon Nn as follows:

$$g_{velocity} = \left(\dot{\mathbf{x}}_{desired}^H - \frac{\mathbf{x}_{Nn}^H - \mathbf{x}_0^H}{\sum_{i=1}^{Nn} t_i} \right)^2, \tag{10}$$

where $\dot{\mathbf{x}}_{desired}^H \in R^2$ is the desired horizontal velocity, $\mathbf{x}_{Nn}^H$ is the terminal CoM position, $\mathbf{x}_{Nn}^H$ is the actual CoM position, and t_i is the durations of i^{th} phase. Note that the terminal state defines the latest state that we consider in the planning horizon.

The Mechanical Cost of Transport (MCoT) quantifies the mechanical energy of transporting the robot from one place to another. Minimizing the MCoT reduces the energy consumption for navigating a determined terrain. Since we do not have access to the joint torques and velocities in our optimization,

[7]The robot is in stance phase when all the feet are on the ground.

we estimate the MCoT from the kinetic energy for a point-mass system (i.e. $\mathcal{K} = \frac{1}{2}m\dot{\mathbf{x}}^2$). Thus, we compute the total cost along the phases by:

$$g_{cot} = \sum_{i=1}^{Nn} COT(\dot{\mathbf{x}}), \tag{11}$$

where the MCoT for point-mass system is defined as $COT(\dot{\mathbf{x}})\frac{\mathcal{K}}{mgd}$ with d equals to the travel distance in the xy plane.

To cope with different terrain difficulties, we compute a costmap from an onboard sensor as described in Section III-A. The costmap quantifies how desirable it is to place a foot at a specific location using geometric features such as *height deviation*, *slope* and *curvature*. This allows the robot to negotiate different terrain conditions (Fig. 8). Thus, given a footshift and CoM position, we compute the foothold location cost as:

$$g_{terrain} = \mathbf{w}^T\mathbf{T}(x, y), \tag{12}$$

where $\mathbf{w}$ and $\mathbf{T}(x, y)$ are the weights and cost values of every feature, respectively. We use a cell grid resolution of 2 cm, a half of the robot's foot size, and the terrain features are computed from a voxel resolution of 2 cm (described in Section III-A). As in [2], we demonstrated that this coarse map is a good trade-off in terms of computation time and information resolution for foothold selection. We cannot guarantee convexity in the terrain costmap, which has to be considered in our optimization process.

3) Soft-constraints

As mentioned in Section IV-A1b, the CoP trajectory must be kept inside the support polygon which is shrunk by a margin r. This margin guarantees dynamic stability when a maximum moment is applied to the CoM (Section IV-A1b). We use a set of nonlinear inequality constraints to describe the support region:

$$\mathbf{L}(\boldsymbol{\sigma})^T \begin{bmatrix} \mathbf{p} \\ 1 \end{bmatrix} > \mathbf{0}, \tag{13}$$

where $\mathbf{L}(\cdot) \in R^{l\times 3}$ are the coefficients of the l lines, $\boldsymbol{\sigma}$ the support region defined from the selected foothold locations, and $\mathbf{p}$ the CoP position. Note that the stability constraints are nonlinear as a consequence of adding the foothold positions as decision variables.

Due to the decoupling of the horizontal and vertical motions, we implement a preview model soft-constraint that ensures the cart-table height is approximately equal to:

$$h = \|\mathbf{x} - \mathbf{p}\| \tag{14}$$

where $\mathbf{x}$ and $\mathbf{p}$ are the CoM and CoP positions, respectively. Note that when the cart-table is falling down, the CoM trajectory increases exponentially as in Eq. (3). This effect arises from the fact that we decouple the horizontal and vertical dynamics, hence adding this soft-constraint guarantees the validity of the model.

To reduce the computation time, we impose both soft-constraints only in the initial and terminal state of each phase as they will be guaranteed in the entire phase. In fact, the linear CoP trajectory will belong to the convex support polygon if the initial and terminal positions are inside this region. Note that the support polygon remains a convex hull as the possible foothold locations cannot cross its geometric center. We ensure this by limiting the foothold search region, i.e. by bounding the footshift (see Fig. 7). These soft-constraints are described as quadratic cost terms.

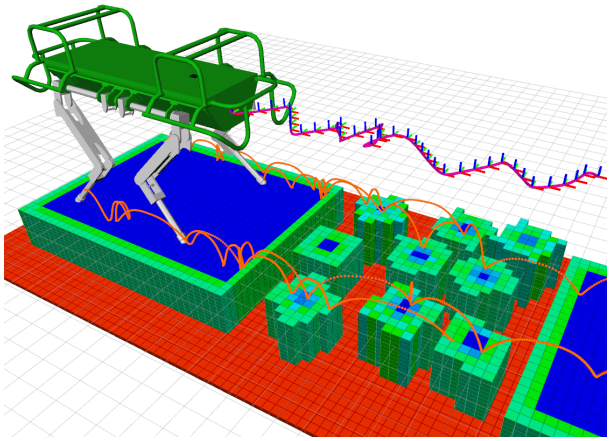


Fig. 8: A costmap allows the robot to negotiate different terrain conditions while following the desired user commands. The costmap is computed from onboard sensors as described in Section III-A. The cost values are continuous and represented in color scale, where blue is the minimum and red is the maximum cost. (Figure taken from [3])

V. Whole-Body Controller

The CoM motion, body attitude and swing motions are controlled by a *trunk controller*. It computes the feed-forward joint torques $\boldsymbol{\tau}^*_{ff}$ necessary to achieve a desired motion without violating friction, torques or kinematic limits. To fulfill these additional constraints we exploit the redundancy in the mapping between the joint space ($\in R^n$) and the body task ($\in R^6$). To address unpredictable events (e.g. limit foot divergence in case of slippage on an unknown surface), an impedance controller computes in parallel the feedback joint torques $\boldsymbol{\tau}_{fb}$ from the desired joint motion $(\mathbf{q}^d_j, \dot{\mathbf{q}}^d_j)$. This controller receives position/velocity set-points that are consistent with the body motion in order to prevent conflicts with the trunk controller. In nominal operations the biggest contribution is generated by the feed-forward torques, i.e. by the trunk controller.

In our previous works [3, 14], the *trunk controller* was designed for quasi-static motions. Indeed we only optimized the ground reaction forces, and then map them into feed-forward joint torques through the centroidal dynamics of the robot. Note that this is a quasi-static mapping where, for instance, the base accelerations and joint motions have no direct influence on the feed-forward torques. Here we improve it by incorporating the full dynamics, the kinematic and torque limits of the robot. This controller allows us to track faster motions than in the previous cases. For instance, for the dynamic motions obtained in Section VI, the legs' joint dynamics starts to play a role, and it becomes necessary

to relax the assumption of quasi-static motions to achieve a good tracking.

As in [44], we consider the full dynamics of the robot and optimize both the generalized accelerations $\mathbf{q}$ and the contact forces $\boldsymbol{\lambda}$. This enables our controller to enforce joint kinematic constraints (Section V-D) and torque limits (Section V-C). Thus, assuming rigid body modeling we can write the equation of motion of the robot as: [8]

$$\underbrace{\begin{bmatrix}\mathbf{M}_b & \mathbf{M}_{bj}\\ \mathbf{M}_{bj}^T & \mathbf{M}_j\end{bmatrix}}_{\mathbf{M}(\mathbf{q})}\underbrace{\begin{bmatrix}\ddot{\mathbf{q}}_b\\ \ddot{\mathbf{q}}_j\end{bmatrix}}_{\ddot{\mathbf{q}}}+\underbrace{\begin{bmatrix}\mathbf{h}_b\\ \mathbf{h}_j\end{bmatrix}}_{\mathbf{h}(\mathbf{q},\dot{\mathbf{q}})}=\underbrace{\begin{bmatrix}\mathbf{0}_{6\times n}\\ \mathbf{I}_{n\times n}\end{bmatrix}}_{\mathbf{S}^T}\boldsymbol{\tau}+\underbrace{\begin{bmatrix}\mathbf{J}_{sb}^T\\ \mathbf{J}_{sj}^T\end{bmatrix}}_{\mathbf{J}_s(\mathbf{q})^T}\boldsymbol{\lambda}, \quad (15)$$

where we split the actuated part from the unactuated part $\mathbf{q} = \begin{bmatrix}\mathbf{q}_b^T & \mathbf{q}_j^T\end{bmatrix}^T \in SE(3)\times R^n$: $\mathbf{q}_b \in SE(3)$, represents the pose of the floating-base part, while $\mathbf{q}_j \in R^n$ are the joint coordinates characterizing the angular position of the n actuated joints. $\ddot{\mathbf{q}}_b = \begin{bmatrix}\ddot{\mathbf{x}}_b^T & \dot{\boldsymbol{\omega}}_b^T\end{bmatrix}^T$ represents the floating-base linear and angular acceleration while $\dot{\mathbf{q}}_b = \begin{bmatrix}\dot{\mathbf{x}}_b^T & \boldsymbol{\omega}_b^T\end{bmatrix}^T \in R^6$ represents the floating-base linear and angular velocity, $\boldsymbol{\tau} \in R^n$ is the vector of actuated joint torques while $\boldsymbol{\lambda} \in R^k$ if the vector of contact forces that are mapped into a force vector through the stack of Jacobians $\mathbf{J}_s \in R^{k\times(6+n)}$. As our robot has nearly point feet, the contact wrench is 3-dimensional and $k = 3c$, where c represents the number of (rigid) contacts with the ground. $\mathbf{M} \in R^{(6+n)\times(6+n)}$ is the joint-space inertia matrix, $\mathbf{h} \in R^{6+n}$ is the force vector that accounts for Coriolis, centrifugal, and gravitational forces, finally $\mathbf{S} \in R^{n\times(6+n)}$ is the matrix that selects the actuated degrees of freedom.

A. Compliance control

To achieve compliantly desired trunk motions (base motions), we compute a reference CoM acceleration ($\ddot{\mathbf{x}}_c^r \in R^3$) and body angular acceleration ($\dot{\boldsymbol{\omega}}_b^r \in R^3$) through a *virtual model*:

$$\begin{aligned}\ddot{\mathbf{x}}_c^r &= \ddot{\mathbf{x}}_c^d + \mathbf{K}_x(\mathbf{x}_c^d - \mathbf{x}_c) + \mathbf{D}_x(\dot{\mathbf{x}}_c^d - \dot{\mathbf{x}}_c),\\ \dot{\boldsymbol{\omega}}_b^r &= \dot{\boldsymbol{\omega}}_b^d + \mathbf{K}_\theta e(\mathbf{R}_b^d\mathbf{R}_b^\top) + \mathbf{D}_\theta(\boldsymbol{\omega}_b^d - \boldsymbol{\omega}_b),\end{aligned} \quad (16)$$

where $(\mathbf{x}_c^d, \dot{\mathbf{x}}_c^d, \ddot{\mathbf{x}}_c^d) \in R^3$ are the desired CoM position, velocity and acceleration respectively, $e(\cdot) : R^{3\times3} \to R^3$ maps the rotation matrix into an associated rotation vector, $\boldsymbol{\omega}_b, \dot{\boldsymbol{\omega}}_b \in R^3$ are the angular velocity and acceleration of the base. $\mathbf{K}_x, \mathbf{D}_x, \mathbf{K}_\theta, \mathbf{D}_\theta \in R^{3\times3}$ are positive-definite diagonal matrices of proportional and derivative gains, respectively.

It is known that the dynamical model in Eq. (15) gets simplified if the CoM velocity is used as a generalized velocity instead of the velocity of the base link [45]. This means replacing the velocity vector $\dot{\mathbf{q}}$ with $\dot{\mathbf{q}}_c$ [9]:

$$\underbrace{\begin{bmatrix}\dot{\mathbf{x}}_b\\ \boldsymbol{\omega}_b\\ \dot{\mathbf{q}}_j\end{bmatrix}}_{\dot{\mathbf{q}}} = \begin{bmatrix}\mathbf{I}_{3\times3} & \left[\mathbf{x}_c(\mathbf{q}_j)\right]_\times & -\mathbf{J}_c(\mathbf{q}_j)\\ \mathbf{0}_{3\times3} & \mathbf{I}_{3\times3} & \mathbf{0}_{3\times n}\\ \mathbf{0}_{n\times3} & \mathbf{0}_{n\times3} & \mathbf{I}_{n\times n}\end{bmatrix}\underbrace{\begin{bmatrix}\dot{\mathbf{x}}_c\\ \boldsymbol{\omega}_b\\ \dot{\mathbf{q}}_j\end{bmatrix}}_{\dot{\mathbf{q}}_c}, \quad (17)$$

where $[.]_\times$ is the skew symmetric operator, $\mathbf{J}_c(\mathbf{q}_j) = \partial\mathbf{x}_c(\mathbf{q}_j)/\partial\mathbf{q}_j$ maps joint velocities into CoM velocities expressed in the robot's base frame. Note that $\mathbf{x}_c(\mathbf{q}_j)$ is the vector that denotes the position of the CoM about the base frame origin, (expressed in the world frame). Using this new coordinates system, the inertia matrix in Eq. (15) becomes block diagonal [45]:

$$\begin{bmatrix}m\mathbf{I}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times n}\\ \mathbf{0}_{3\times3} & \mathbf{I}_{r_c} & \mathbf{0}_{3\times n}\\ \mathbf{0}_{n\times3} & \mathbf{0}_{n\times3} & \bar{\mathbf{M}}_j\end{bmatrix}\begin{bmatrix}\ddot{\mathbf{x}}_c\\ \dot{\boldsymbol{\omega}}_b\\ \ddot{\mathbf{q}}_j\end{bmatrix}+\begin{bmatrix}m\mathbf{g}\\ \mathbf{0}\\ \bar{\mathbf{h}}_j\end{bmatrix}=\begin{bmatrix}\mathbf{0}\\ \mathbf{0}\\ \boldsymbol{\tau}\end{bmatrix}+\bar{\mathbf{J}}_s^T\boldsymbol{\lambda}, \quad (18)$$

where $\mathbf{I}_{r_c} \in R^{3\times3}$ is the rotational inertia of the robot w.r.t the CoM. Note that $\bar{[\cdot]}$ operator denotes that the matrices/vectors recomputed after the coordinate transform. In particular the single contact Jacobian $\bar{\mathbf{J}}_{s_i}$ is $\bar{\mathbf{J}}_{s_i} = \begin{bmatrix}\mathbf{I}_{3\times3} & \left[\mathbf{x}_{bf_i} - \mathbf{x}_c\right]_\times & \mathbf{J}_i(\mathbf{q}_j) - \mathbf{J}_c(\mathbf{q}_j)\end{bmatrix}$, where $\mathbf{x}_{bf_i}$ is the vector from the base origin to the foot contact point i and $\mathbf{J}_i(q)$ maps joint velocities into the Cartesian velocity of foot i. It is easy to see that the first 3 rows of Eq. (18) represent the Newton equation that describes the rate of change of linear momentum of the robot:

$$m(\ddot{\mathbf{x}}_c + \mathbf{g}) = \sum_{i=1}^{c}\boldsymbol{\lambda}_i. \quad (19)$$

B. Optimization

To be able to enforce the aforementioned constraints, we cast the compliance controller as an optimization problem (in our case a Quadratic Programming (QP) problem) in which the target is to minimize the error between the reference and actual accelerations (linear and angular), whereas our decision variables $\mathbf{u} \in R^{6+n+k}$ are the generalized accelerations $\ddot{\mathbf{q}} \in R^{6+n}$ and ground contact forces $\boldsymbol{\lambda} \in R^k$:

$$\begin{aligned}\mathbf{u}^* &= \underset{\mathbf{u}}{\operatorname{argmin}}\, g(\mathbf{u}) + \|\mathbf{u}\|_{\mathbf{W}}\\ \text{s.t.}\quad & \mathbf{A}\mathbf{u} = \mathbf{b}\\ & \underline{\mathbf{d}} < \mathbf{C}\mathbf{u} < \bar{\mathbf{d}}.\end{aligned} \quad (20)$$

The equality constraints $\mathbf{A}\mathbf{u} = \mathbf{b}$ encode dynamic consistency (e.g. physics is not violated), stance constraints and swing tasks, while the inequality constraints $\underline{\mathbf{d}} < \mathbf{C}\mathbf{u} < \bar{\mathbf{d}}$ encode friction constraints, joint kinematic and torque limits. All of them are stacked in the matrix $\mathbf{A}^T = \begin{bmatrix}\mathbf{A}_p^T & \mathbf{A}_{st}^T & \mathbf{A}_{sw}^T\end{bmatrix}$ and $\mathbf{C}^T = \begin{bmatrix}\mathbf{C}_{fr}^T & \mathbf{C}_j^T & \mathbf{C}_\tau^T\end{bmatrix}$ will be detailed in the following sections.

1) Cost

The first term of the cost in Eq. (20) represents the cost relative to the *tracking* error:

$$g(\mathbf{u}) = \left\| \begin{matrix}\ddot{\mathbf{x}}_c - \ddot{\mathbf{x}}_c^r\\ \dot{\boldsymbol{\omega}}_b - \dot{\boldsymbol{\omega}}_b^r\end{matrix} \right\|_{\mathbf{Q}}, \quad (21)$$

while the secondary objective is a regularization factor to keep the solution bounded or to pursue additional criteria, $\mathbf{Q}$ and $\mathbf{W}$ are the respective positive-definite weight matrices. Note that $\mathbf{Q}$ can be chosen to give different importance to the task error direction (e.g. to implement some "graceful degradation" strategy). The sub-block $\mathbf{W}_{kk}$, relative to the contact forces,

[8] Unless differently specified all vectors are represented with respect to the world frame.

[9] Matrix dimensions that are not specified are assumed to be 3×3.

of $\mathbf{W}$ can be adjusted to pursue different optimal criteria (e.g. minimizing joint torques or improving robustness finding contact forces towards the middle of the cone, as explained in [14]). As the CoM acceleration is not a decision variable, we compute it from the contact forces using the linear momentum part of the centroidal dynamic model (Eq. (19)). Thus, we re-write the tracking cost Eq. (21) as $\|\mathbf{G}\mathbf{u} - \mathbf{g}_0\|_{\mathbf{Q}}$ where: [10]

$$\mathbf{G} = \begin{bmatrix} \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times n} & \frac{1}{m}\mathbf{I}^1_{3\times3} \cdots \frac{1}{m}\mathbf{I}^c_{3\times3} \\ \mathbf{0}_{3\times3} & \mathbf{I}_{3\times3} & \mathbf{0}_{3\times n} & \mathbf{0}_{3\times k} \end{bmatrix},$$
$$\mathbf{g}_0 = \begin{bmatrix} \ddot{\mathbf{x}}^r_c + \mathbf{g} \\ \dot{\boldsymbol{\omega}}^r_b \end{bmatrix}, \tag{22}$$

and $\mathbf{I}^1_{3\times3}, \cdots, \mathbf{I}^c_{3\times3}$ are the identity matrices.

2) Physical consistency

To enforce consistency between the contact forces and the floating-base accelerations, we impose the unactuated part (the first 6 rows) of the robot's full dynamics equation Eq. (15) as an equality constraint:

$$\mathbf{A}_p = \begin{bmatrix} \mathbf{M}_b & \mathbf{M}_{bj} & -\mathbf{J}^T_{sb} \end{bmatrix},$$
$$\mathbf{b}_p = -\mathbf{h}_b. \tag{23}$$

3) Stance condition

Avoiding slipping on stance conditions can be described as $\ddot{\mathbf{x}}_f(\mathbf{q}) = \mathbf{0} \in R^{3s}$, where s represents the number of stance feet. To encode these constraints, we re-write them at the acceleration level in order to have a direct dependency from the decision variables ($\mathbf{J}_f\ddot{\mathbf{q}} + \dot{\mathbf{J}}_f\dot{\mathbf{q}} = \mathbf{0}$):

$$\mathbf{A}_{st} = \begin{bmatrix} \mathbf{J}_f & \mathbf{0}_{k\times k} \end{bmatrix},$$
$$\mathbf{b}_{st} = -\dot{\mathbf{J}}_f\dot{\mathbf{q}}, \tag{24}$$

where $\mathbf{J}_f \in R^{k\times(6+n)}$ is the stack of Jacobian of the stance feet, and $\dot{\mathbf{J}}_f$ is its time derivative. For numerical precision, we compute the product $\dot{\mathbf{J}}_f\dot{\mathbf{q}}$ using spatial algebra.

4) Swing task

In our quadruped robot we have two tasks: a *body motion task* and a *swing task* (one or more swinging legs). Due to the particular kinematic structure we never have conflict between these tasks nor we have an indetermination (i.e. there is no nullspace in the joint space). Namely, we always have a task dimension of the same size of the floating-base space. Therefore, there is no need to implement hierarchies or postural tasks as in [44]. In fact we can encode the swing task directly as an *equality constraint*, i.e. by enforcing our swing feet to follow a reference swing acceleration $\ddot{\mathbf{x}}_{f_{sw}}(\mathbf{q}) = \ddot{\mathbf{x}}^r_{f_{sw}} \in R^{3sw}$ where $sw = 4 - c$ is the number of swing legs. As in Section V-B3, we can differentiate twice the constraint to have it at the acceleration level, obtaining

$$\mathbf{J}_{sw}\ddot{\mathbf{q}} + \dot{\mathbf{J}}_{sw}\dot{\mathbf{q}} = \ddot{\mathbf{x}}^r_{f_{sw}}, \tag{25}$$

that in matrix form become[11]:

$$\mathbf{A}_{sw} = \begin{bmatrix} \mathbf{J}_{sw} & \mathbf{0}_{3(4-c)\times k} \end{bmatrix},$$
$$\mathbf{b}_{sw} = \ddot{\mathbf{x}}^r_{f_{sw}} - \dot{\mathbf{J}}_{sw}\dot{\mathbf{q}}, \tag{26}$$

where $\mathbf{J}_{sw} \in R^{3sw\times(6+n)}$ is the stack of swing Jacobians in which there is not contribution from the floating-base part. In fact, we compute the reference swing acceleration w.r.t. the base frame, and using an attractor equation similarly to Eq. (16).

5) Friction cone constraints

The goal of the friction constraints is to avoid slippage and realize a smooth loading/unloading of the legs. For that, we ensure that the contact forces lie inside the friction cones and their normal components stay within some user-defined values (i.e. maximum and minimum force magnitudes). We approximate the friction cones with square pyramids to express them with linear constraints. The fact that the ground contacts are unilateral constraints (i.e. the legs can only push and not pull the ground) can be naturally encoded by setting a zero lower bound of the normal component, while the upper bound allows us to regulate the amount of "loading" for each leg. We define the friction constraints as part of the inequality constraints of our QP as:

$$\underline{\mathbf{d}}_{fr} < \mathbf{C}_{fr}\mathbf{u} < \bar{\mathbf{d}}_{fr}, \quad \mathbf{C}_{fr} = \begin{bmatrix} \mathbf{0}_{p\times(6+n)} & \mathbf{F} \end{bmatrix}$$

with:

$$\mathbf{F} = \begin{bmatrix} \mathbf{F}_0 & \dots & \mathbf{0} \\ \vdots & \ddots & \vdots \\ \mathbf{0} & \dots & \mathbf{F}_c \end{bmatrix}, \quad \underline{\mathbf{d}}_{fr} = \begin{bmatrix} \underline{\mathbf{f}}_0 \\ \vdots \\ \underline{\mathbf{f}}_c \end{bmatrix}, \quad \bar{\mathbf{d}}_{fr} = \begin{bmatrix} \bar{\mathbf{f}}_0 \\ \vdots \\ \bar{\mathbf{f}}_c \end{bmatrix}, \tag{27}$$

where $\mathbf{F} \in R^{p\times k}$ is a block diagonal matrix encoding the cones boundaries for each stance leg and $\underline{\mathbf{d}}_{fr}, \bar{\mathbf{d}}_{fr} \in R^p$ are the lower/upper bound respectively. Note that $p = 5c$ is the number of inequality constraints and $k = 3c$ is the dimension of the force vector. For a single i^{th} stance leg, we have 4 inequalities for the pyramid bounds and 1 for the normal force:

$$\mathbf{F}_i = \begin{bmatrix} (-\mu_i\mathbf{n}_i + \mathbf{t}_{1_i})^\top \\ (-\mu_i\mathbf{n}_i + \mathbf{t}_{2_i})^\top \\ (\mu_i\mathbf{n}_i + \mathbf{t}_{2_i})^\top \\ (\mu_i\mathbf{n}_i + \mathbf{t}_{1_i})^\top \\ \mathbf{n}_i^\top \end{bmatrix}, \underline{\mathbf{f}}_i = \begin{bmatrix} -\infty \\ -\infty \\ 0 \\ 0 \\ \lambda_{\min_i} \end{bmatrix}, \bar{\mathbf{f}}_i = \begin{bmatrix} 0 \\ 0 \\ \infty \\ \infty \\ \lambda_{\max_i} \end{bmatrix}, \tag{28}$$

where $\mathbf{n}_i \in R^3$ is the direction normal to the surface, $\mathbf{t}_{1_i}$, $\mathbf{t}_{2_i} \in R^3$ are the tangential directions, $\mu_i \in R$ is the coefficient of friction, and $\lambda_{\min_i}, \lambda_{\max_i} \in R$ are the minimum and maximum values allowed for the normal forces of the i^{th} foot, respectively. Note that the direction of the surface is estimated from the terrain mapping.

C. Torque limits

We deliberately chose to do not incorporate the torques among the decision variables since they are redundant. They can be expressed as bi-linear function of accelerations and

[10] In order to be able to set operational space gains that have physical meaning (e.g. spring/dampers), without any loss of generality, it is possible to pre-multiply the linear/angular accelerations (and the cost G) by the mass and the rotational inertia, respectively.

[11] Alternatively, it is possible to write the swing task in the joint space rather than in the operational space by changing the matrix A_{sw}, b_{sw}.

contact forces. Therefore we can enforce constraints on the joint torques (e.g. the actuation limits $\boldsymbol{\tau}_{\min} < \boldsymbol{\tau} < \boldsymbol{\tau}_{\max}$) by rephrasing them as constraints on both acceleration and contact forces. For that we exploit the actuated part of the full dynamics Eq. (15):

$$\underline{\mathbf{d}}_\tau < \mathbf{C}_\tau \mathbf{u} < \bar{\mathbf{d}}_\tau, \tag{29}$$

$$\begin{aligned} \mathbf{C}_\tau &= \begin{bmatrix} \mathbf{M}_{bj}^T & \mathbf{M}_j & -\mathbf{J}_{cj}^T \end{bmatrix}, \\ \underline{\mathbf{d}}_{\boldsymbol{\tau}} &= -\mathbf{h}_j + \boldsymbol{\tau}_{\min}(\mathbf{q}_j), \\ \bar{\mathbf{d}}_{\boldsymbol{\tau}} &= -\mathbf{h}_j + \boldsymbol{\tau}_{\max}(\mathbf{q}_j), \end{aligned} \tag{30}$$

where $\boldsymbol{\tau}_{\min}(\mathbf{q}_j), \boldsymbol{\tau}_{\max}(\mathbf{q}_j) \in R^n$ are the lower/upper bounds on the torques. These bounds must be recomputed at each control loop because they are dependent on the joint positions. This is due to the presence of linkages on the sagittal joints (e.g. to translate the motion from linear to rotational). They set a joint dependent profile on the maximum torque that is non-linear in the joint range.

D. Joint kinematic limits

We enforce joint kinematic constraints as function of the joint accelerations (i.e. $\ddot{\mathbf{q}}_{j_{\min}} < \ddot{\mathbf{q}}_j < \ddot{\mathbf{q}}_{j_{\max}}$). We select them via the matrix $\mathbf{C}_j$:

$$\underline{\mathbf{d}}_j < \mathbf{C}_j \mathbf{u} < \bar{\mathbf{d}}_j, \tag{31}$$

$$\begin{aligned} \mathbf{C}_j &= \begin{bmatrix} \mathbf{0}_{n\times 6} & \mathbf{1}_{n\times n} & \mathbf{0}_{n\times k} \end{bmatrix}, \\ \underline{\mathbf{d}}_j &= \ddot{\mathbf{q}}_{j_{\min}}(\mathbf{q}_j) \qquad \bar{\mathbf{d}}_j = \ddot{\mathbf{q}}_{j_{\max}}(\mathbf{q}_j), \end{aligned}$$

also the upper/lower bounds on accelerations $\ddot{\mathbf{q}}_{j_{\min,\max}}(\mathbf{q}_j)$ should be recomputed at each loop. They are set in order to make the joint to a achieve the end-stop (zero velocity) in a time interval $\Delta t = 10dt$, where dt is the loop duration. For instance if the joint are at a distance $\mathbf{q}_j - \mathbf{q}_{j_{\max}}$ from the end-stops with a velocity $\dot{\mathbf{q}}_j$, the deceleration to cover this distance in a Δt time interval and approach the end-stop with zero velocity will be:

$$\ddot{\mathbf{q}}_{j_{\min,\max}} = -\frac{2}{\Delta t^2}(\mathbf{q}_{j_{\min,\max}} - \mathbf{q}_j - \Delta t\, \dot{\mathbf{q}}_j). \tag{32}$$

E. Torque computation

We map the optimal solution $\mathbf{u}^* = \begin{bmatrix} \ddot{\mathbf{x}}^* & \boldsymbol{\lambda}^* \end{bmatrix}$ into desired feed-forward joint torques $\boldsymbol{\tau}_{ff}^* \in R^n$ using the actuated part of the full dynamics equation of the robot as:

$$\boldsymbol{\tau}_{ff}^* = \begin{bmatrix} \mathbf{M}_{bj}^T & \mathbf{M}_j \end{bmatrix} \ddot{\mathbf{q}}^* + \mathbf{h}_j - \mathbf{J}_{cj}^T \boldsymbol{\lambda}^* \tag{33}$$

Finally, the trunk controller torques $\boldsymbol{\tau}^*$ are summed with the joint PD torques (i.e. feedback torques $\boldsymbol{\tau}_{fb}$) to form the desired torque command $\boldsymbol{\tau}^d$ that is sent to the low-level joint-torque controllers [46]:

$$\boldsymbol{\tau}^d = \boldsymbol{\tau}_{ff}^* + PD(\mathbf{q}_j^d, \dot{\mathbf{q}}_j^d), \tag{34}$$

where $\mathbf{q}_j^d \in R^n, \dot{\mathbf{q}}_j^d \in R^n$ are the desired joint positions and velocities[12].

[12]Note that $\mathbf{q}_j^d, \dot{\mathbf{q}}_j^d$ should be consistent with the body task and the joint accelerations computed from the trunk controller $\ddot{\mathbf{q}}_j^*$.

VI. Experimental Results

To understand the benefit of the coupled motion planning method, we first compare the decoupled and coupled approaches in various challenging terrains (e.g. stepping stones, pallet, stairs and gap). We use as test-cases the decoupled and coupled planners presented in [1] and [3], respectively. After that, we demonstrate how trunk attitude modulation can heuristically generate angular motions that ensure the CoP condition. Subsequently, we analyze the effect of the terrain costmap in our coupled planner. We show how different weighting choices result in various behaviors without affecting significantly the robot stability and the reduction of the MCoT. Finally we demonstrate the capabilities of our locomotion framework (i.e. coupled planner, whole-body controller, terrain mapping and state estimation) by crossing terrains with various slopes and obstacles. For all the experimental results, please see the accompanying video in Extension 1 or in Youtube[13].

All the experiments are conducted with HyQ, a 85 kg hydraulically actuated quadruped robot [47]. The HyQ robot is fully-torque controlled and equipped with precision joint encoders, a depth camera (Asus Xtion), a MultiSense SL sensor and an Inertial Measurement Unit (MicroStrain). HyQ roughly has the dimensions of a goat, i.e. 1.0 m×0.5 m×0.98 m (length × width × height). The leg length ranges from 0.339-0.789 m and the hip-to-hip distance is 0.75 m (in the sagittal plane). HyQ has two onboard computers: a Pentium i5 with Real Time (RT) Linux (Xenomai) patch, and a Pentium i5 with Linux. The Xenomai PC handles the low-level control (hydraulic-actuator control) at 1 KHz and communicates with the proprioceptive sensors through EtherCAT boards. Additionally, this PC runs the high-level controller (whole-body controller) at 250 Hz. Both RT threads (i.e. low- and high-level controllers) communicate through shared memory. On the other hand, the non-RT PC processes the exteroceptive sensors for generating the terrain map and then computing the plans. These motion plans are sent to the whole-body controller (i.e. the RT PC) through a RT-friendly communication.

A. Decoupled and coupled planning

In this section we show how our decoupled and coupled planning approaches perform on various challenging terrains, including terrains with height variations such as the gap and stepping stones cases. For all these scenarios, we computed the costmap using the *standard deviation of the height values*, which is estimated through a regression in a 2 cm×2 cm window around the cell of interest. The costmap is built using a resolution of (2 cm×2 cm×1 cm) in (x, y, z), respectively (Section III). The higher resolution value in z reduces the discrepancy between expected and detected footholds, that sometimes lead to inconsistencies, that in turn generate tacking errors.

For the decoupled planner, the swing and stance durations are pre-configured since they cannot be optimized over, as in a decoupled planner. The footstep planner explores partially a set of candidate footholds using the terrain-aware heuristic

[13]https://youtu.be/ywkiCu3ZAyE

TABLE II: Number of footholds, averaged walking speed and normalized MCoT for various challenging terrains for our coupled (Coup.) and decoupled (Dec.) planners. We normalize the MCoT with respect to the walking velocity. All the results are computed from simulations.

	# of Footholds			Avg. Speed [cm/s]			MCoT / speed [s/cm]		
Terrain	Coup.	Dec.	Ratio	Coup.	Dec.	Ratio	Coup.	Dec.	Ratio
S. Stones	**31**	38	0.82	**11.16**	6.29	1.77	**13.20**	11.43	1.15
Pallet	**35**	36	0.97	**9.23**	6.92	1.33	**13.21**	11.70	1.13
Stairs	**21**	23	0.91	**12.79**	11.26	1.14	**10.22**	6.24	1.63
Gap	**18**	24	0.75	**12.76**	9.00	1.42	**9.05**	6.84	1.32

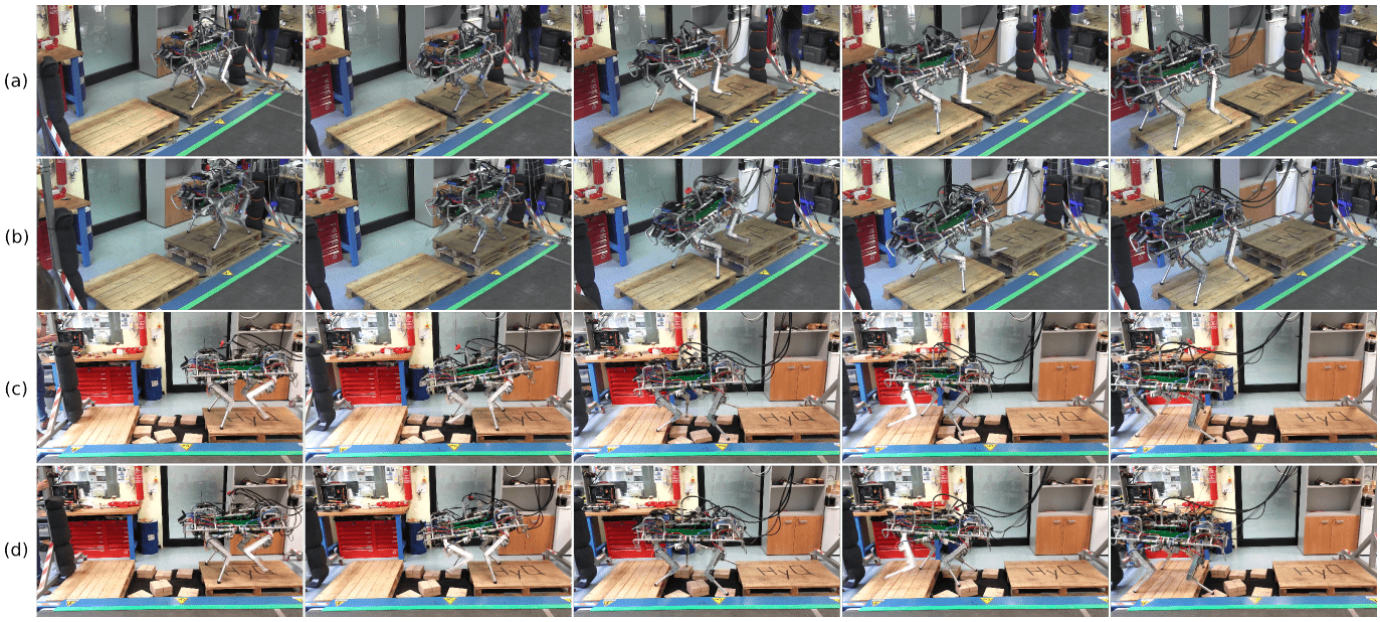


Fig. 9: Snapshots of experimental trials used to evaluate the performance of the coupled planner. (a) crossing a gap of 25 cm length while climbing up 6 cm. (b) crossing a gap of 25 cm length while climbing down 12 cm. (c) crossing a set of 7 stepping stones. (d) crossing a sparse set of stepping stones with different stone elevations (6 cm). To watch the video, click the figure.

function [see 2]. It might not be possible to compute a set of polynomial's coefficients, given a predefined swing and stance duration, that satisfy the dynamic stability for a determined footstep sequence choice. We tuned those durations per every terrain, they range from 0.5 to 0.7 sec and from 0.05 to 1.4 sec for the swing and stance[14] phases, respectively.

In the case of the coupled planner, we used the *same* weight values (see Section IV-B2-IV-B3) for all the results presented in Table II (i.e. 300, 30 and 10 for the human velocity commands, terrain, and CoM energy weight values, respectively). We do not re-tune these gains for navigating all these terrains, as is sometimes necessary with the decoupled planner; in this respect the coupled planner shows a higher level of generality compared with our decoupled planner. We impose a soft-constraint boundary in the terrain cost, when the terrain cost is higher than 80% of its maximum value. All soft-constraints have higher weights and a high offset cost[15], which allows the CMA-ES solver to ensure the constraints are satisfied, given enough exploration steps [48]. We hand-tuned the terrain soft-constraint parameters (weight and offset) in such a way that the dynamic stability and preview model soft-constraints are not violated. For all the cases presented in this paper, the defined mapping from geometry feature to terrain cost values (see Section III-A) is suitable. However, with machine learning, we can infer a mapping function that increases the generality of the terrain costmap model as explained in [29]. Another important point is that our coupled planner does not depend on having a good warm-start, which might be difficult to define for all possible terrain topologies. We used the same stability margin and allowed angular acceleration (as in Section VI-B) for the trunk attitude planner, and our horizon is $N = 1$, i.e. 1 cycle of locomotion or 4 steps.

Compared to the decoupled planner, we managed to increase the walking velocity at least 14%, while also modulating the trunk attitude. The foothold error is on average around 2 cm, half compared with the decoupled planner; we get these results with the state estimation algorithm proposed in [49]. This dramatically increases the success rate of the stepping stones trials to 90%; an increment of 30% with respect to the decoupled

[14]In this work, with *stance* phase, we refer to all the feet on ground.

[15]This method allows us to handling non-linear constraints, and it often works better than resampling. See [48] for a general overview on boundary and constraints handling.

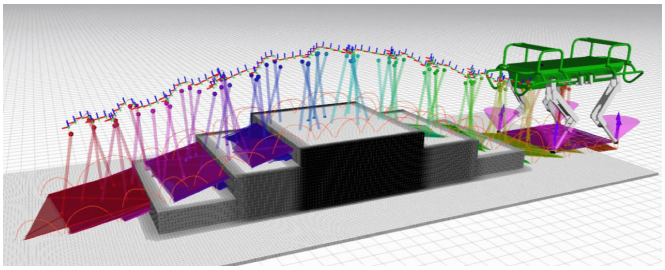


Fig. 10: An optimized sequence of control parameters for the stair climbing case. As in previous experiments, we use the same optimization weight values for the entire course of the motion. To watch the video, click the figure.

planner, see the reported success rate in [1]. We define as *success* when the robot crosses the various obstacles of the terrain, e.g. it does not make a step in the gap, reaches torque limits, etc. In Table II we report the number of footholds, the averaged trunk speed, and the MCoT our coupled and decoupled planners for various challenging terrains. Jointly optimizing the motion and footholds reduces the number of required footholds for crossing a terrains because it considers the robot dynamics for the foothold selection. It also increases the trunk speed and success rate even with terrain elevation changes (e.g. gap and stepping stones). The MCoT is higher for our coupled planner; however, this is an effect of higher walking velocities and of the tuning of the cost function. This is expected even if we normalized the MCoT with respect to the walking velocity. Note that the velocity increases quadratically the kinetic energy, and as a consequence the MCoT. We also found that the tuning of the MCoT cost does not affect the stability and the foothold selection. An important drawback of optimizing foothold location and step timing giving a terrain costmap is that increases substantially the computation time. In fact for our planners, it is increased from 2-3 sec to 10-15 min, for more details about the computation time of the decoupled planner see [2]. The main reason is that we search for global minimum by estimating the gradient in the latter case [see 43]. Instead, for the former one, we use a tree-search algorithm (i.e. Anytime Repairing A* (ARA*)) with heuristic function that guides the solution towards a shortest path, not the safest one.

Trunk attitude adaptation tends to overextend the legs, especially in challenging terrains, as larger motions are required. To avoid kinematic limits, we defined a foot search region. This ensures kinematic feasibility up to 12 cm of terrain height difference (coupled planner), as is illustrated in Fig. 9a,b. Note that we had to define a more conservative foot search region in the decoupled one, making very challenging to cross gaps or stepping stones with height variations. We could also generate trajectories with two stepping stones 6 cm higher than the other ones. These terrain irregularities produce a trunk modulation in roll and pitch as can be observed in Fig. 9d; the terrain height is used for the trajectory generation not for the optimization (as explained in Fig. 5). The execution performance on stepping stones without changes in terrain elevation is shown in Fig. 9c. Crossing the terrain in Fig. 9a-d is only possible with the coupled planner since we managed to increase the foothold region from (20 cm×23.5 cm) to (34 cm×28 cm). For all our optimizations, we define a stability margin of $r = 0.1$ m which is good trade-off between modeling error and allowed trunk attitude adjustment on the HyQ robot. Note that the origin of this region is defined by the stance frame (see Fig. 7), and that increasing this region enables broader foothold options. In fact, the coupled planning considers the robot's dynamics as it jointly optimizes the CoM motions and foothold locations, while the decoupled planning can only consider the robot's kinematics for the foothold planning. Additionally, we show in simulation that our planner can climb stairs (see Fig. 10). The robot computes a footstep sequence for climbing up and down using the terrain costmap and the same optimization weights as before. This supports that previously tuned weights generalize well in new terrains.

B. Approximating the angular momentum effects

Both motion planning methods use the *cart-table* model which reduces the dimensionality of the problem but it neglects the angular momentum of the motion. However, the robot needs to modulate its attitude (i.e. change the angular momentum) for navigating terrains with various heights. Therefore, we have proposed (Section IV-A1b) a trunk attitude method that ensures the CoP condition. We showcase the automatic trunk attitude modulation, during a dynamic walk, as illustrated in Fig. 11a. To validate the attitude modulation method, we plan a *fast* (compared to the common walking-gait velocities of HyQ) dynamic walk with a trunk velocity of 18 cm/s, with initial trunk attitude of 0.17 and 0.22 radians in roll and pitch, respectively. We do not use the terrain costmap for generating the corresponding footholds, thus the resulting feet locations come from the dynamics of walking itself, while maximizing the stability of the gait. We compute the maximum allowed angular acceleration given the trunk inertia matrix of HyQ, from Eq. (7), which results in 0.11 rad/s^2 as the maximum diagonal element. The trunk attitude planner uses this maximum allowed acceleration to align the trunk and support plane through cubic polynomial splines (as explained in Section IV-A1b).

The resulting behavior shows the HyQ robot successfully walking while changing its trunk roll and pitch angles. The trunk attitude planner adjusts the roll and pitch angles given the estimated support region at each phase. Fig. 11b shows the CoM tracking performance for initial trunk attitude of 0.17 rad and 0.22 rad in roll and pitch, respectively. Fig. 11c shows that the entire attitude modulation is accomplished in the first 6 phases (i.e. one cycle of locomotion or four steps). Because our attitude planner ensures dynamic stability, the HyQ robot crosses successfully terrains with various elevations as shown in Fig. 9a-d. Note that the stability margin is the same for all the experiments in this paper ($r = 0.1$ m).

C. The effect of the terrain costmap

The terrain costmap plays an important role for the foothold selection. Different weighting choices on the terrain costmap produce various behaviors, affected by Eq. (12). In Fig. 12

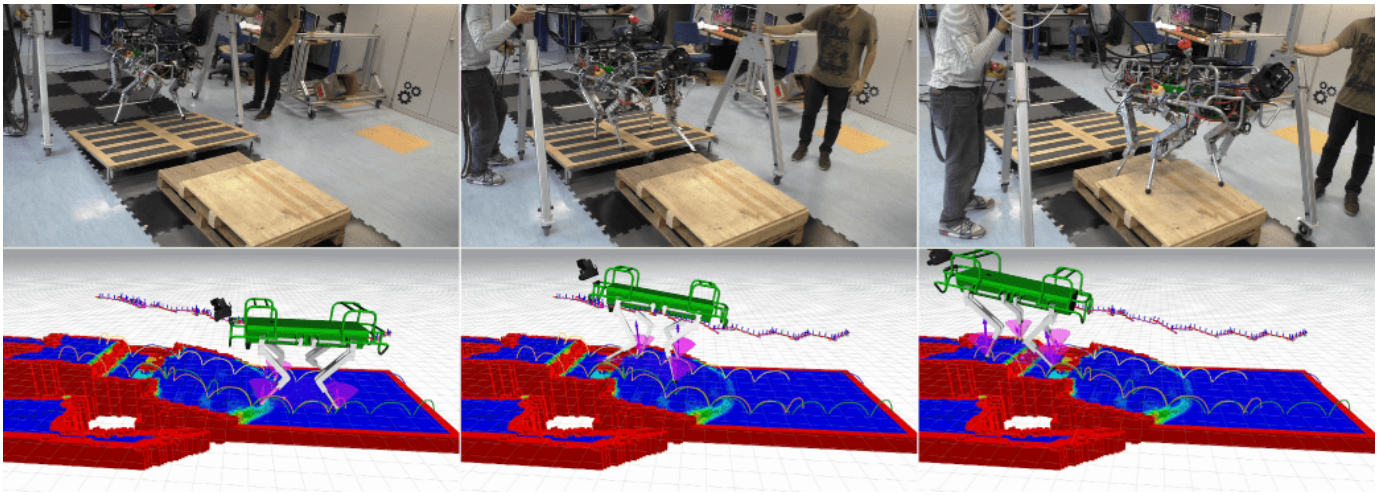


Fig. 13: Crossing a terrain that combines elements of the previous cases; first a ramp of 10 degree, then a gap of 15 cm and finally a step with 15 cm height. Execution of the planned motion with the HyQ robot (top). Visualization of the terrain costmap, friction cone and Ground Reaction Forces (GRFs) (bottom). The color for the friction cone and GRFs are magenta and purple, respectively. To watch the video, click the figure.

we show two different behaviors obtained with two different weights values of the terrain cost function. For simplicity, we analyzed the effect of these weights for gap crossing. In this study case we compute the terrain costmap using only the height deviation feature, since the geometry of the terrain is simple. The cost values are represented using gray scale, where white and black are the minimum and maximum cost values, respectively. A higher value in the terrain weight describes a higher risk for foothold locations near the borders of the gap. Strongly penalizing the terrain costmap results in the robot not being able to cross the gap due to its kinematic limits (Fig. 12(*bottom*)). By reducing the terrain weight up to an appropriate value, the coupled planner decides to select footholds closer to the gap borders, which allows the robot to cross the gap (Fig. 12(*top*)). The terrain weight mainly influences the foothold selection, and does not influence the stability or the MCoT.

D. Crossing terrain with various slopes

Our coupled planner does not consider the non-coplanar contact condition and friction cone (since the used cart-table model neglects them). However, the HyQ robot can still cross successfully a wide range of terrains, as demonstrated in Fig. 13. The robot can successfully cross in simulation ramps up to 20 degrees in similar friction conditions to real experiments ($\mu = 0.7$). For the non-coplanar condition problem we use the cart-table model to plan horizontal CoM motions; then we ensure dynamic stability even with trunk attitude adjustments (i.e. applying a bounded CoM torque). Additionally, our whole-body controller achieves the planned motion without violating friction, torques or kinematics constraints (see Fig. 14). For that, it considers the full robot dynamics and optimizes both CoM accelerations and contact forces; for instance, the GRFs have to lie inside the friction cone constraints Fig. 13(*bottom*). The terrain surface normals are computed online from vision (see Section III). The coefficient of friction used in this trials (i.e. simulation and experiments) is 0.7, which is a conservative estimation of the real contact conditions.

VII. Discussion

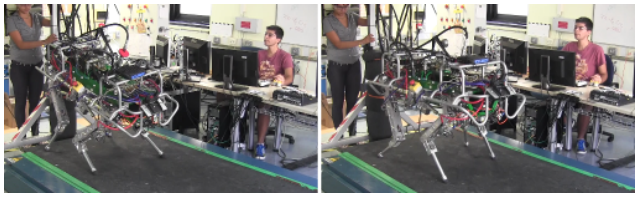
In Section VI, we performed a substantial number of trials with the HyQ robot. To compare decoupled and coupled planning approaches, we used the similar terrain environments for the two groups of experiments. Hereafter, we describe the factors that improve the overall performance of the tasks.

A. Decoupled and coupled planning

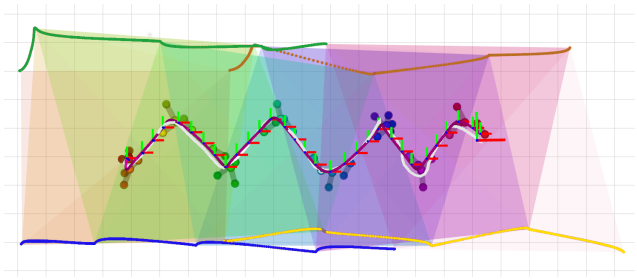
Coupled motion and foothold planning allows us to consider the dynamics for the foothold selection. Considering the dynamics is important to increase the range of potential foothold locations and to adjust the step duration; both allow the robot to cross a broader range of terrains. We noticed that the coupled planner handles various terrain elevations more easily because of the joint optimization process. Crossing gaps with various elevations exposed the limitation of decoupled methods, since the required motions (steps) were larger (see Fig. 9a). However, an important drawback of coupled foothold and motion planning is the increment of the computation time compared with decoupled planning. It is possible to reduce the computation time by describing the foothold through integer variables [e.g. 22, 23], but this would not allow us to model non-linear curvature of the terrain. Instead the presented coupled planning uses a terrain model that considers a broader range of challenging environments. In any case, the computation time remains longer for coupled planning as we presented in [24].

B. Considering angular momentum effects

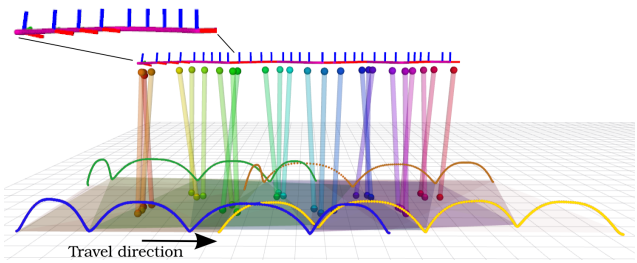
The *cart-table model* estimates the CoP position, yet it neglects the angular components of the body motion, which



(a) Dynamic walking and trunk modulation



(b) CoM tracking performance



(c) Trunk attitude modulation

Fig. 11: (a) Dynamic attitude modulation by approximating the angular momentum effects. The initial trunk attitude is 0.17 and 0.22 radians in roll and pitch, respectively. (b) Body tracking when walking and dynamically modulating the trunk attitude. The planned CoM (magenta) and the executed trajectory (white) are shown together with the sequence of support polygons, CoP and CoM positions. Note that each phase is identified with a specific color. (c) A lateral view of the same motion shows the attitude correction (sequence of frames), and the cart-table displacement. Note that we use the RGB color convention for drawing the different frames. In (b)-(c) the brown, yellow, green and blue trajectories represent the LF, RF, LH and RH foot trajectories, respectively.

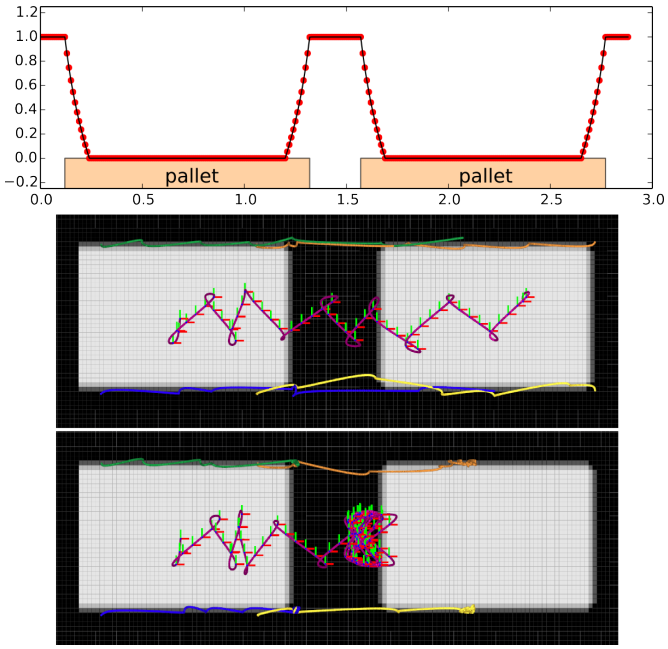


Fig. 12: The effect of changing terrain weight values when crossing a gap of 25 cm. The costmap is computed only using the height deviation feature (*top*); the red points represent the discretization of the continuous cost function (1 cm). If we choose an appropriate terrain weight value the robot crosses the gap (*middle*). In contrast, an increment of 200% in the weight penalizes excessively footholds close to the gap and as result the robot cannot cross the gap as kinematic limits are exceeded (*bottom*).

can lead to inaccurate estimation within the support polygon. This can affect the stability when going up or down gaps or stairs, crossing uneven stepping stones with various elevations, etc. To systematically address these effects without affecting the stability, we found a relationship between the applied torques to the CoM and the displacement of the CoP. Later, we connected it with the stability margin by assuming a time-invariant inertial tensor approximation of the inertia matrix. Experimental results with the HyQ robot validated this method for flat and challenging terrain locomotion. Our proposed method can be applied to other legged systems, such as humanoids.

C. The effect of the terrain costmap

Considering the terrain topology increases the complexity of the trajectory optimization problem. Moreover, optimizing the step duration introduces many local minima in the problem landscape. For solving these issues, we propose a low-dimensional parametrized model which allows us to solve the optimization problem with stochastic-based exploration. Even thought our problem is non-convex, we reduced the number of required footholds by an average of 13.75% compared to our convex decoupled planner (Table II).

D. Considering terrain with slopes

Higher walking speed increases the probability of foot-slippage. When one or some of the feet slip backwards, or when a foot is only slightly loaded, in the subsequent base motion phase th “pushing” backwards can result into foot slippage. Both events are more likely to happen in a terrain with different elevations due to errors in the state estimation or noise in the perception sensors. Including friction-cone constraints in the inverse dynamics torque calculation step has shown to generate movements without foot slippage. We demonstrated experimentally that is possible to navigate a wide range of terrain slopes without considering the friction cone stability in the planning level.

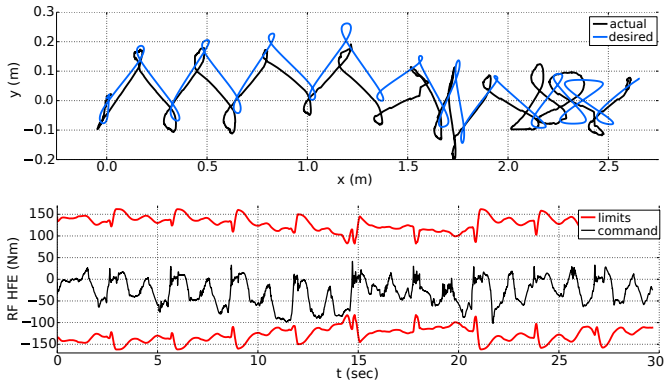


Fig. 14: The execution performance of the HyQ robot crossing a terrain that combines elements of all previous cases. (*Top*): CoM tracking performance, desired (blue) and executed (black) motions. The tracking error is mainly due to low-frequency correction of the pose estimate drift from exteroceptive sources. (*Bottom*): applied torque command along the course of the motion. At $t = 14$ sec, the planned motion produced a movement that reached the torque limits; however, the controller applies a torque command inside the robot's limits. In fact, the tracking error increases at approximately $x = 1.25m$, and is reduced in the next steps.

E. Terrain mapping and state estimation

Estimating the state of the robot with a level of accuracy suitable for planned motions has been proven to be a challenging task. Reliable state estimation is crucial for planned walks, as accurate foot placement directly depends on the robot's base pose estimate. The body pose estimate is also used to compute the feed-forward torque commands through a virtual model. The major sources of error for inertial-legged state estimation are Inertial Measurement Unit (IMU) gyro bias and foot slippage. These produce a pose estimate drift, which can be reduced by improving the contact state estimate [37], but it cannot be completely eliminated, since the pose is not observable from proprioceptive sources. The pose drift particularly affects the feed-forward torques, which are computed from the trunk controller, see Eq. (33). To eliminate it, we fused high frequency (1 kHz) proprioceptive sources (inertial and Leg odometry) with low frequency exteroceptive updates (0.5 Hz for LiDAR scan matching, 10 Hz for visual odometry) in a combined Extended Kalman Filter [49]. However, we noticed that the drift accumulated in between the high frequency proprioceptive updates and the low frequency exteroceptive updates affected the overall execution during our experimental trials. In practice, to cope with this problem we reduced the compliance of our whole-body controller, by increasing the proportional control gains, since the controller can quickly track the pose estimate corrections from exteroceptive updates.

VIII. Conclusion

In this paper, we presented our framework for dynamic whole-body locomotion on challenging terrain. We presented our coupled planning approach that exploits terrain normals and torque limits for real-time whole-body control. We compared with prior work on motion planning methods and highlighted the advantages and disadvantages of coupled and decoupled motion and foothold planning. In our test-case planners, we built a unified method for quantifying the terrain difficulty (i.e. terrain costmap). We showed that our terrain model is suitable for decoupled and coupled planning. We showed that reduced models for motion planning (such as cart-table) are still suitable for a wide range of challenging scenarios. In fact, we used full dynamic models in our real-time whole-body controller in order to avoid slippage, torque and kinematic limits. Furthermore, these models allow us to better formulate the trajectory optimization while also considering the terrain topology. We demonstrated that coupled planners increase the locomotion capabilities, at the price of higher computation time and problem complexity.

A. Future works

The decoupled and coupled motion planners are able to generate specific behaviors such as the walking gait. An important limitation of coupled planning is the high computation time, which is required for replanning. Learning a control policy from a data base of control parameters could potentially tackle the limitation regarding the computation time. On the other hand, some terrain conditions cannot be successfully crossed with a pre-specified behavior/gait. Many cases may require more general behaviors, where we need to consider the contact forces, discontinuities and the hybrid nature of the dynamics of a legged robot. Including the contact forces in the problem formulation and optimization might improve the motion generality, for example as shown in [26, 27, 50].

Finding useful model representations for legged locomotion has been explored earlier [e.g. 51], where Central Pattern Generators (CPGs) are used as an efficient representation to integrate sensory information into trajectory generation. However, it is not clear how to use such methods for motion planning because they do not allow us to easily predict the system's stability in a determined horizon. Instead, we believe that our representation (using control parameters) allows us to evaluate and predict the system's stability in a more intuitive and computationally efficient manner. In fact, we could potentially integrate reactive strategies such as *step reflexes* for negotiation of unexpected obstacles [13] and *slip recovery* for uncertainties over the terrain normal and friction coefficient [52] along a planned motion. We believe that a combination of both approaches will increase the required robustness for real-world applications.

Acknowledgment

This work was supported by the Istituto Italiano di Tecnologia (IIT), with additional funding from the European Union's Seventh Framework Programme for research, technological development and demonstration under grant agreement no 601116 as part of the ECHORD++ (The European Coordination Hub for Open Robotics Development) project under the experiment called *HyQ-REAL*. The authors would like to thank also the other members of the IIT's Dynamic Legged

Systems Lab who contributed to the success of this project: Andreea Radulescu, Marco Camurri, Romeo Orsolino and former members Felipe Polido, Jose Colmenares and Stephane Bazeille.

REFERENCES

[1] A. Winkler, C. Mastalli, I. Havoutis, M. Focchi, D. G. Caldwell, and C. Semini, "Planning and Execution of Dynamic Whole-Body Locomotion for a Hydraulic Quadruped on Challenging Terrain," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2015.

[2] C. Mastalli, A. Winkler, I. Havoutis, D. G. Caldwell, and C. Semini, "On-line and On-board Planning and Perception for Quadrupedal Locomotion," in *IEEE International Conference on Technologies for Practical Robot Applications (TEPRA)*, 2015.

[3] C. Mastalli, M. Focchi, I. Havoutis, A. Radulescu, S. Calinon, J. Buchli, D. G. Caldwell, and C. Semini, "Trajectory and Foothold Optimization using Low-Dimensional Models for Rough Terrain Locomotion," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2017.

[4] E. Muybridge, *Animals in motion*. Courier Dover Publications, 1957.

[5] R. B. McGhee and A. A. Frank, "On the stability properties of quadruped creeping gaits," *Mathematical Biosciences*, vol. 3, no. 0, pp. 331–351, 1968.

[6] J. Estremera and P. G. De Santos, "Free gaits for quadruped robots over irregular terrain," *The International Journal of Robotics Research (IJRR)*, vol. 21, no. 2, pp. 115–130, 2002.

[7] H.-W. Park and S. Kim, "Quadrupedal galloping control for a wide range of speed via vertical impulse scaling," *Bioinspiration & Biomimetics*, vol. 10, no. 2, 2015.

[8] H. W. Park, M. Y. Chuah, and S. Kim, "Quadruped bounding control with variable duty cycle via vertical impulse scaling," in *IEEE International Conference on Intelligent Robots and Systems (IROS)*, 2014.

[9] R. Orsolino, M. Focchi, C. Mastalli, H. Dai, C. Darwin, and C. Semini, "A New Feasibility Metric for Trajectory Optimisation of Legged Robots using Wrench Polytopes," 2017.

[10] M. H. Raibert, *Legged robots that balance*. MIT press Cambridge, MA, 1986, vol. 3.

[11] V. Barasuol, J. Buchli, C. Semini, M. Frigerio, E. R. De Pieri, and D. G. Caldwell, "A Reactive Controller Framework for Quadrupedal Locomotion on Challenging Terrain," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2013.

[12] I. Havoutis, J. Ortiz, S. Bazeille, V. Barasuol, C. Semini, and D. G. Caldwell, "Onboard Perception-Based Trotting and Crawling with the Hydraulic Quadruped Robot (HyQ)," in *IEEE International Conference on Intelligent Robots and Systems (IROS)*, 2013.

[13] M. Focchi, V. Barasuol, I. Havoutis, C. Semini, D. G. Caldwell, V. Barasuol, and J. Buchli, "Local Reflex Generation for Obstacle Negotiation in Quadrupedal Locomotion," *International Conference on Climbing and Walking Robots and the Support Technologies for Mobile Machines (CLAWAR)*, 2013.

[14] M. Focchi, A. del Prete, I. Havoutis, R. Featherstone, D. G. Caldwell, and C. Semini, "High-slope terrain locomotion for torque-controlled quadruped robots," *Autonomous Robots*, vol. 41, no. 1, pp. 259–272, 2017.

[15] J. R. Rebula, P. D. Neuhaus, B. V. Bonnlander, M. J. Johnson, and J. E. Pratt, "A controller for the littledog quadruped walking on rough terrain," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2007.

[16] D. Pongas, M. Mistry, and S. Schaal, "A robust quadruped walking gait for traversing rough terrain," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2007.

[17] J. Z. Kolter, M. P. Rodgers, and A. Y. Ng, "A control architecture for quadruped locomotion over rough terrain," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2008.

[18] M. Kalakrishnan, J. Buchli, P. Pastor, M. Mistry, and S. Schaal, "Learning, planning, and control for quadruped locomotion over challenging terrain," *The International Journal of Robotics Research (IJRR)*, vol. 30, no. 2, pp. 236–258, 2010.

[19] M. Zucker, N. Ratliff, M. Stolle, J. Chestnutt, J. A. Bagnell, C. G. Atkeson, and J. Kuffner, "Optimization and learning for rough terrain legged locomotion," *The International Journal of Robotics Research (IJRR)*, vol. 30, no. 2, pp. 175–191, 2011.

[20] A. Shkolnik, M. Levashov, I. R. Manchester, and R. Tedrake, "Bounding on rough terrain with the LittleDog robot," *The International Journal of Robotics Research (IJRR)*, vol. 30, no. 2, pp. 192–215, 2011.

[21] P. Vernaza, M. Likhachev, S. Bhattacharya, A. Kushleyev, and D. D. Lee, "Search-based planning for a legged robot over rough terrain," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2009.

[22] R. Deits and R. Tedrake, "Footstep Planning on Uneven Terrain with Mixed-Integer Convex Optimization," in *IEEE International Conference on Humanoid Robots*, 2014.

[23] B. Aceituno-Cabezas, H. Dai, J. Cappelletto, J. C. Grieco, and G. Fernandez-Lopez, "A Mixed-Integer Convex Optimization Framework for Robust Multilegged Robot Locomotion Planning over Challenging Terrain," in *IEEE International Conference on Intelligent Robots and Systems (IROS)*, 2017.

[24] B. Aceituno-Cabezas, C. Mastalli, H. Dai, M. Focchi, A. Radulescu, D. G. Caldwell, J. Cappelletto, J. C. Grieco, G. Fernandez-Lopez, and C. Semini, "Simultaneous Contact, Gait and Motion Planning for Robust Multi-Legged Locomotion via Mixed-Integer Convex Optimization," *IEEE Robotics and Automation Letters (RAL)*, 2017.

[25] Y. Tassa and E. Todorov, "Stochastic Complementarity for Local Control of Discontinuous Dynamics," in

Robotics: Science and Systems (RSS), 2010.

[26] I. Mordatch, E. Todorov, and Z. Popović, "Discovery of complex behaviors through contact-invariant optimization," *ACM Transactions on Graphics*, vol. 31, no. 4, pp. 1–8, 2012.

[27] M. Posa, C. Cantu, and R. Tedrake, "A direct method for trajectory optimization of rigid bodies through contact," *The International Journal of Robotics Research (IJRR)*, 2013.

[28] H. Dai, A. Valenzuela, and R. Tedrake, "Whole-body Motion Planning with Simple Dynamics and Full Kinematics," in *IEEE International Conference on Humanoid Robots*, 2014.

[29] M. Kalakrishnan, J. Buchli, P. Pastor, and S. Schaal, "Learning locomotion over rough terrain using terrain templates," in *IEEE International Conference on Intelligent Robots and Systems (IROS)*, 2009.

[30] A. Escande, A. Kheddar, and S. Miossec, "Planning support contact-points for humanoid robots and experiments on HRP-2," in *IEEE International Conference on Intelligent Robots and Systems*, 2006.

[31] K. Hauser and J. C. Latombe, "Multi-modal Motion Planning in Non-expansive Spaces," *The International Journal of Robotics Research (IJRR)*, vol. 29, no. 7, pp. 897–915, 2009.

[32] M. Zucker, J. A. Bagnell, C. Atkeson, and J. Kuffner, "An Optimization Approach to Rough Terrain Locomotion," in *IEEE International Conference on Automation and Robotics (ICRA)*, 2010.

[33] C. Mastalli, "Planning and Execution of Dynamic Whole-Body Locomotion on Challenging Terrain," Ph.D. dissertation, Istituto Italiano di Tecnologia (IIT) and University of Genova, 2017.

[34] A. Hornung, K. M. Wurm, M. Bennewitz, C. Stachniss, and W. Burgard, "OctoMap: An efficient probabilistic 3D mapping framework based on octrees," *Autonomous Robots*, 2013. [Online]. Available: http://octomap.github.com

[35] J. Z. Kolter, Y. Kim, and A. Y. Ng, "Stereo vision and terrain modeling for quadruped robots," in *IEEE International conference on Robotics and Automation (ICRA)*, 2009.

[36] R. B. Rusu, "Semantic 3D Object Maps for Everyday Manipulation in Human Living Environments," Ph.D. dissertation, Computer Science department, Technische Universitaet Muenchen, Germany, 2009.

[37] M. Camurri, M. Fallon, S. Bazeille, A. Radulescu, V. Barasuol, D. G. Caldwell, and C. Semini, "Probabilistic Contact Estimation and Impact Detection for State Estimation of Quadruped Robots," *IEEE Robotics and Automation Letters (RAL)*, 2017.

[38] I. Mordatch, M. de Lasa, and A. Hertzmann, "Robust physics-based locomotion using low-dimensional planning," *ACM Transactions on Graphics*, vol. 29, no. 4, p. 1, 2010.

[39] R. Full and D. Koditschek, "Templates and anchors: neuromechanical hypotheses of legged locomotion on land," *Journal of Experimental Biology*, vol. 202, no. 23, pp. 3325–3332, 1999.

[40] D. E. Orin, A. Goswami, and S. H. Lee, "Centroidal dynamics of a humanoid robot," *Autonomous Robots*, vol. 35, pp. 161–176, 2013.

[41] 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 (ICRA)*, 2003.

[42] M. B. Popovic, A. Goswami, and H. Herr, "Ground Reference Points in Legged Locomotion: Definitions, Biological Trajectories and Control Implications," *The International Journal of Robotic Research (IJRR)*, vol. 24, pp. 1013–1032, 2005.

[43] N. Hansen, "CMA-ES: A Function Value Free Second Order Optimization Method," in *PGMO COPI 2014*, Paris, France, 2014.

[44] A. Herzog, N. Rotella, S. Mason, F. Grimminger, S. Schaal, and L. Righetti, "Momentum Control with Hierarchical Inverse Dynamics on a Torque-Controlled Humanoid," *Autonomous Robots*, vol. 40, no. 3, pp. 473–491, Mar. 2016.

[45] C. Ott, M. a. Roa, and G. Hirzinger, "Posture and balance control for biped robots based on contact force optimization," in *IEEE International Conference on Humanoid Robots*, 2011, pp. 26–33.

[46] T. Boaventura, J. Buchli, C. Semini, and D. G. Caldwell, "Model-based hydraulic impedance control for dynamic robots," *IEEE Transaction on Robotics (TRO)*, vol. 31, no. 6, pp. 1324–1336, 2015.

[47] C. Semini, N. G. Tsagarakis, E. Guglielmino, M. Focchi, F. Cannella, and D. G. Caldwell, "Design of HyQ – a Hydraulically and Electrically Actuated Quadruped Robot," *IMechE Part I: Journal of Systems and Control Engineering*, vol. 225, no. 6, pp. 831–849, 2011.

[48] N. Hansen, "The CMA Evolution Strategy: A Tutorial," *ArXiv preprint arXiv:1604.00772*, 2016. [Online]. Available: https://arxiv.org/abs/1604.00772

[49] S. Nobili, M. Camurri, V. Barasuol, M. Focchi, D. G. Caldwell, C. Semini, and M. Fallon, "Heterogeneous Sensor Fusion for Accurate State Estimation of Dynamic Legged Robots," in *Robotics: Science and Systems (RSS)*, 2017.

[50] C. Mastalli, I. Havoutis, M. Focchi, D. G. Caldwell, and C. Semini, "Hierarchical Planning of Dynamic Movements without Scheduled Contact Sequences," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2016.

[51] L. Righetti and A. J. Ijspeert, "Programmable central pattern generators: an application to biped locomotion control," in *IEEE International Conference on Robotics and Automation*, 2006.

[52] M. Focchi, V. Barasuol, M. Frigerio, D. G. Caldwell, and C. Semini, "Slip Detection and Recovery for Quadruped Robots," in *International Symposium on Robotics Research (ISRR)*, 2015.

Carlos Mastalli is currently a Robotics Researcher in the Gepetto Team at LAAS-CNRS working on the topic of multi-contact planning and control in legged robots. He received his M.Sc. in Mechatronics, from Simon Bolivar University in 2013, working on a machine learning method for autonomous backhoe machines. After that, he completed his Ph.D. on "Planning and Execution of Dynamic Whole-Body Locomotion on Challenging Terrain" in April 2017 at Istituto Italiano di Tecnologia. He is also improved significantly the locomotion framework of the HyQ robot. His research interests include optimal control, motion planning, whole-body control and machine learning for legged locomotion.

Ioannis Havoutis is a Lecturer in Robotics at the University of Oxford. He is part of the Oxford Robotics Institute and a co-lead of the Dynamic Robot Systems group. His focus is on approaches for dynamic whole-body motion planning and control for legged robots in challenging domains. From 2015 to 2017, he was a postdoc at the Robot Learning Interaction Group, at the Idiap Research Institute. Previously, from 2011 to 2015, he was a senior postdoc at the Dynamic Legged System lab the Istituto Italiano di Tecnologia. He holds a Ph.D. and M.Sc. from the University of Edinburgh.

Michele Focchi is currently a Researcher at the Advanced Robotics department, Istituto Italiano di Tecnologia. He received both the B.Sc. and the M.Sc. in Control System Engineering from Politecnico di Milano in 2004 and 2007, respectively. In 2009 he started to develop a novel concept of air-pressure driven micro-turbine for power generation in which he obtained an international patent. In 2013, he got a Ph.D. degree in Robotics, where he developed low-level controllers for the Hydraulically Actuated Quadruped (HyQ) robot. Currently his research interests are dynamic planning, optimization, locomotion in unstructured/cluttered environments, stair climbing, model identification and whole-body control.

Darwin G. Caldwell is a founding Director at the Istituto Italiano di Tecnologia in Genoa, Italy, and a Honorary Professor at the Universities of Sheffield, Manchester, Bangor, Kings College, London and Tianjin University China. His research interests include innovative actuators, humanoid and quadrupedal robotics and locomotion (iCub, HyQ and COMAN), haptic feedback, force augmentation exoskeletons, dexterous manipulators, biomimetic systems, rehabilitation and surgical robotics, telepresence and teleoperation procedures. He is the author or co-author of over 450 academic papers, and 17 patents and has received awards and nominations from several international journals and conferences.

Claudio Semini received a M.Sc. degree in Electrical Engineering and Information Technology from ETH Zurich, Switzerland, in 2005. He is the Head of the Dynamic Legged Systems (DLS) laboratory at Istituto Italiano di Tecnologia (IIT). From 2004 to 2006, he first visited the Hirose Laboratory at Tokyo Tech, and later the Toshiba R&D Center, Japan. During his doctorate from 2007 to 2010 at the IIT, he developed the hydraulic quadruped robot HyQ and worked on its control. After a postdoc with the same department, in 2012, he became the Head of the DLS lab. His research interests include the construction and control of versatile, hydraulic legged robots for real-world environments.