Heuristic Planning for Rough Terrain Locomotion in Presence of External Disturbances and Variable Perception Quality

Michele Focchi[1] Romeo Orsolino[1] Marco Camurri[1]
Victor Barasuol[1] Carlos Mastalli[1,2] Darwin G. Caldwell[1] Claudio Semini[1]

***Abstract*— The quality of the visual feedback can vary significantly on a legged robot that is meant to traverse unknown and unstructured terrains. The map of the environment, acquired with online state-of-the-art algorithms, often degrades after a few steps, due to sensing inaccuracies, slippage and unexpected disturbances. When designing locomotion algorithms, this degradation can result in planned trajectories that are not consistent with the reality, if not dealt properly. In this work, we propose a heuristic-based planning approach that enables a quadruped robot to successfully traverse a *significantly* rough terrain (*e.g.,* stones up to 10 cm of diameter), in *absence* of visual feedback. When available, the approach allows also to exploit the visual feedback (*e.g.,* to enhance the stepping strategy) in *multiple ways*, according to the *quality* of the 3D map. The proposed framework also includes reflexes, triggered in specific situations, and the possibility to estimate *online* an unknown time-varying disturbance and compensate for it. We demonstrate the effectiveness of the approach with experiments performed on our quadruped robot *HyQ* (85 kg), traversing different terrains, such as: ramps, rocks, bricks, pallets and stairs. We also demonstrate the capability to estimate and compensate for disturbances, showing the robot walking up a ramp while pulling a cart attached to its back.**

I. Introduction

Legged robots are mainly designed to traverse unstructured environments, which are often demanding in terms of torques and speeds. Trajectory optimization [1]–[5] can be used to generate dynamically stable body motions, taking into account: robot dynamics, kinematic limits, leg reachability for motion generation and foothold selection. In particular, motion planning enables the necessary *anticipative* behaviors to address appropriately the terrain. These include: avoiding an obstacle, selecting the most convenient foothold and contact force to realize a certain body motion, and navigate toward a desired direction (*e.g.,* [3], [6]).

Furthermore, when the robot goes beyond the flat terrain condition, a $3D/2.5D$ map of the environment is required, to appropriately address uneven terrain features, through optimization.

Despite the considerable efforts and recent advances in this field [3], [4], [7], [8], optimal planning that takes into account terrain conditions is still far from being realized in *online* on a real platform, due to the computational complexity involved in the optimization. These optimization problems are strongly nonlinear, prone to local minima, and require a significant amount of computation time to be solved [7]. Some improvements have been recently achieved by computing convex approximations of the terrain [5] or the dynamics [8]. Most of these approaches optimize for a whole time horizon and then execute the motion in "open loop", rather than optimizing *online*. Depending on the complexity of the terrain and on the gait, Aceituno *et al.* managed to reduce the computation time (for a locomotion cycle) in a range between 0.5 and 1.5 s, while the approach by Ponton *et al.* requires between 0.8 and 5 s to optimize for a 10 s horizon. Therefore, it is still not possible to optimize *online* and perform a replanning (*e.g.,* through a Model Predictive Control (MPC) strategy). Indeed, we believe that replanning is a crucial feature to intrinsically cope with the problem of accumulation errors which affect real scenarios. These errors can be caused by delays, inaccuracy of the 3D map, unforeseen events (external pushes, slippages, rock falling), or dynamically changing and deformable terrains (*e.g.,* rolling stones, mud, etc.). The sources of errors in locomotion are many: a premature/delayed touchdown due to a change in the terrain inclination, external disturbances, wrong terrain detection. Other source of errors include: tracking delay in the controller, sensor calibration errors, filtering delays, offsets, structure compliance, unmodeled friction and modeling errors in general. These errors can make the actual robot state diverge from the original plan. Moreover, in the prospect of tele-operated robots, the user might want to modify the robot velocity during the locomotion, and this should be reflected in a prompt change in the motion plan.

As mentioned, a significant source of errors can come from disturbances which are not modeled, such as external pushes. A possible solution to this particular issue is to implement disturbance observers to estimate these external disturbances [9]. Englsberger *et al.* [10] proposed a Momentum Based Disturbance Observer (MBDO) for pure linear force disturbances. In his thesis, Rotella and *et al.* [11], implemented an external wrench estimator for a humanoid robot, based on an Extended Kalman Filter. Besides that, he was also compensating for the estimated wrench in an inverse dynamics controller. However the experimental tests on the real robot were limited to a static load acting on the legs while performing a switch of contact between the two feet and, in a different experiment, to an impulsive disturbance without any contact change. We extend this work by presenting a MBDO, capable of estimating both linear and angular components of external disturbance wrenches of variable amplitude applied on the robot during the execution of a walk.

[1]Department of Advanced Robotics, Istituto Italiano di Tecnologia, Genova, Italy. *email*: michele.focchi, romeo.orsolino, marco.camurri, victor.barasuol, darwin.caldwell, claudio.semini}@iit.it.
[2]LAAS-CNRS, Toulouse, France. *email*: carlos.mastalli@laas.fr

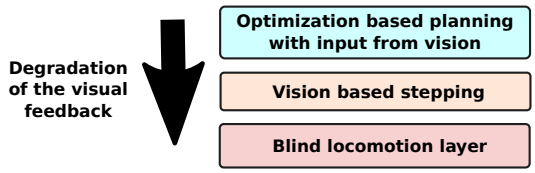

Fig. 1. Locomotion layers for different quality of the visual feedback. The bottom layer is purely reactive (blind) locomotion. The vision-based stepping allows the robot to quickly adapt to local terrain conditions. Then, the motion planning provides the level of anticipation needed in locomotion over challenging terrain.

Despite a disturbance observer is useful to improve the tracking against external disturbances, *online replanning* is still fundamental for rough terrain locomotion, because it represents a mechanism to adapt to the terrain (*terrain adaptation*), quickly recover from planning errors and simultaneously accommodate for the visual input, user setpoint, terrain change and external impacts. In particular, the planning horizon should be large enough to execute the necessary *anticipative* body motions, but at the same time the replanning frequency should be high enough to mitigate the accumulation of errors. Bledt *et al.* [12] implemented online replanning, by introducing a policy-regularized model-predictive control (PR-MPC), for gait generation, where a heuristic policy is meant to provide additional information for the solution of the MPC problem. The authors found that regularizing the optimization with a policy it improves the cost landscapes and decreases the computation time. However, this work has not been demonstrated by hardware experiments. A MPC-based approach has been successfully implemented for a real humanoid in the past. In his seminal work, Wieber [13] addressed the problem of strong perturbations and tracking errors, yet limiting the application to flat terrains. In the quadruped domain, Cheetah 2 has shown running/jumping motions over challenging terrains using a MPC controller [14]. More recently, Bellicoso *et al.* [15], [16] implemented a 1-step replanning strategy, based on Zero Moment Point (ZMP) optimization, on the quadruped robot ANYmal. Their optimization selects the footholds, according to desired user speed and recovery reactions, but it does not incorporate the terrain's map. This limits the applicability to moderately rough terrain.

Most vision-based approaches [3], [6] require reasonably accurate $3D$ maps of the environment [17]. However, the accuracy of a $3D$ map strongly depends on reliable state estimation [18]–[20], which involves complex sensor fusion algorithms (inertial, odometry, LiDAR, cameras). Typically, these algorithms involve the fusion of high-frequency proprioceptive estimates from inertial and leg odometry [18], with low-frequency pose correction from visual odometry [21]. The proprioceptive estimate (and, indirectly, the pose correction) can be jeopardized by unforeseen events such as: slippage, unstable footholds, terrain deformation and compliance of the mechanical structure. For the above reasons, we envision different locomotion layers, according to the quality of the perception feedback available, as depicted in Fig. 1.

At the bottom layer we have a *blind* reactive strategy that is always active. This strategy does not require any vision feedback. This layer mainly contains basic terrain adaptation mechanisms and reflex strategies. There are motion primitives that are triggered, to override the planner in situations where the robot could be damaged (*e.g.,* stumbling).

In cases when the vision feedback is denied (*e.g.,* foggy, smoky, poorly lit areas), a *reactive strategy* is preferred [22], [23]. On the other hand, when a degraded visual feedback is available, this can still be usefully exploited for 1-step horizon planning (middle block in Fig. 1) [24]. Then, when the vision feedback is reliable, it can be used for more sophisticated terrain assessment [25], [26].

In this work, we present a motion control framework for rough terrain locomotion. The terrain adaptation and the mitigation of tracking errors is achieved through a 1-step *online* replanning strategy, which can work either blindly or with visual feedback. This work builds upon a previously presented statically stable *crawl* gait [27] where the main contribution regarded the whole-body control and where the planning part was only superficially drafted. Hereby, our focus is mainly on the capability to adapt to rough terrains during locomotion.

In this framework is also incorporated a number of features to increase the robustness of the locomotion, such as slip detection and two reflex strategies, previously presented in [28], [29]. The reflexes are automatically enabled to address specific situations such as: loss of mobility (in cases of abrupt terrain changes, see *height reflex* in [28]) and unforeseen frontal impacts (see *step reflex* in [29]).

A. Contribution

The main contributions of this article are *experimental*:

- We show that with the proposed approach the Hydraulically actuated Quadruped (HyQ) robot [30] can successfully negotiate different types of terrain templates (ramps, debris, stairs, steps), some of them with a significant roughness[1] (Fig. 2). The size of the stones are up to 12 cm (diameter) that is about 26% the retractable leg length. We also applied this strategy, with little variations (see Section VII-B), to the task of climbing up and down industrial-size stairs, also performing 90deg turns while climbing the stairs in simulation.
- Another *experimental* contribution comes from the application of our MBDO to quadrupedal locomotion. We are able to close the loop and to compensate *online* for the disturbances during locomotion. Additionally, while planning the trajectories, we also consider the shift of the ZMP due to the estimated external disturbance. We will show HyQ walking up a 22 deg inclined ramp, while pulling a 12 kg wheelbarrow attached to his back with a rope. The wheelbarrow was also impulsively loaded up to 15 additional kg, with the weights being added in different instants during the experiment. The

[1]See accompaning video of rough terrain experiments: `https://youtu.be/EUhw-aAFYrw`

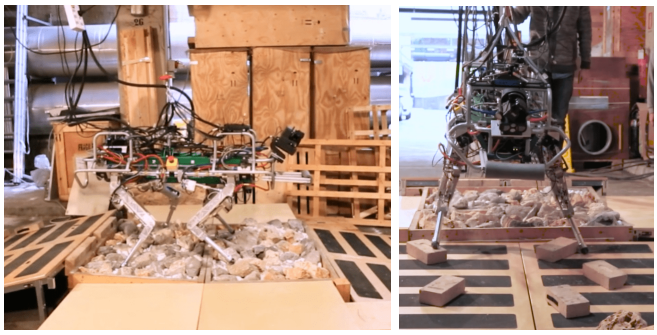

Fig. 2. HyQ crawling on a rough terrain playground: (left) lateral and (right) frontal view. Our locomotion framework can deal with moving rocks and various terrain elevation changes.

robot was also able to crawl robustly on a flat terrain while leaning against a constant horizontal pulling force of 150 N. To the authors knowledge this is the first time such tasks are realized on a real quadruped platform.

- As a marginal contribution, we introduce a *smart terrain estimation* algorithm, that improves a the state of the art terrain estimation and adaptation [15]. This is particularly beneficial in some specific scenarios (*e.g.,* when one stance foot is considerably far from the plane fitted by the other three stance feet).

B. Echord++

This work is part of the Echord++ HyQReal experiment, in which quadrupedal locomotion strategies are developed that will be used for a newly designed hydraulic robot. At the time of writing this manuscript, the new quadruped robot HyQ-REAL was not yet fully assembled. Therefore, the presented framework was demonstrated on its predecessor version HyQ. The presented software framework can easily be run on both platforms, since other layers take care of the differences in robot kinematics and dynamics.

C. Outline

The remainder of this paper is detailed as follows. In Section II we briefly present the locomotion framework; Sections III and IV, detail the body motion and swing motion phases of this gait, respectively. In Section IV, a particular focus is given to the different stepping strategies depending on the presence of a visual feedback; Section V describes the reactive modules to enhance the locomotion on rough terrains and deal with unpredictable events; in Section VI, we present an improved terrain estimator; Section VII shows a strategy for stair climbing based on the proposed framework; in Section VIII, we describe the implementation of our disturbance observer; in section IX we address the conclusions.

II. Locomotion Framework Overview

In this section, we briefly illustrate our *statically* stable crawling framework, (previously presented in [27]), enriched with additional features specific for rough terrain locomotion. Fig. 3 shows the block diagram of the framework. The core module is a state machine (see [27] for details) which orchestrates two temporized/event-driven locomotion phases: the *swing phase*, and the *body motion phase*. In the former, the robot has 3 legs in stance, while in the latter it has 4 legs in stance. During the *body motion phase*, the robot Center of Mass (CoM) is shifted onto the *future* support triangle, (opposite to the *next* swing leg, in accordance to a user-defined foot sequence). As default footstep sequence we use: RH, RF, LH, LF[2]. The CoM trajectory is generated after estimating the terrain inclination (see Section III). At each touchdown, the inclination is updated by fitting a plane through the actual stance feet positions. When the terrain is uneven (and the feet are not coplanar), an *average* plane is found.

The *swing phase* is a swing-over motion followed by a linear search motion (see Section IV), which terminates with a *haptic* touchdown. The haptic touchdown ensures that the swing motion does not stop in a prescheduled way; instead, the leg keeps extending until a new touchdown is established. When a contact is detected (either by thresholding the Ground Reaction Force (GRF), or directly via a foot contact sensor [31], [32]), the touchdown is established.
A search motion with *haptic* touchdown is a key ingredient to achieve robust *terrain adaptation*. Indeed, haptic touchdown is important to mitigate the effect of tracking errors, because it allows to trigger the stance only when the contact is truly stable. This is important whenever there is a discrepancy between the plan and the real world. This typically happens when a vision feedback is used to select the foothold (see Section IV-B), because the accuracy of a height map, used for locomotion, is typically in the order of centimeters [26], [33]. The vision feedback can be also used to estimate the direction of the normal of the terrain under the foot, in order to set the searching, or reaching, motion direction. Both the body and the swing trajectories are generated as quintic polynomials. The body trajectory is always expressed in the so-called *terrain frame*, which is aligned to the terrain plane (see Fig. 4 (left)). The terrain frame has the Z-axis normal to the terrain plane, while the X-axis is a projection of the X-axis of the base on the terrain plane, and the Y-axis is to form a right hand side system of coordinates. The swing trajectory is expressed in the *swing frame*, which, depending on the stepping strategy adopted, can be either coincident with the terrain frame (section IV-A) or set independently for each foot (section IV-B). The *Whole Body Control* module (also known as *Trunk Controller* [27]) computes the torques required to control the position of the robot CoM and the orientation of the trunk. At the same time, it redistributes the weight among the stance legs (c_{st}) and ensures that friction constraints are not violated. The Trunk Controller action can be improved by setting the real terrain normal (under each foot) obtained by a 3D map [26], rather than using a single value for all the feet (*e.g.,* the normal to the terrain plane). This becomes more important (to avoid slippage) when the terrain shape differs significantly from a plane.

[2]LH, LF, RH and RF stands for Left-Hind, Left-Front, Right-Hind and Righ-Front legs, respectively.

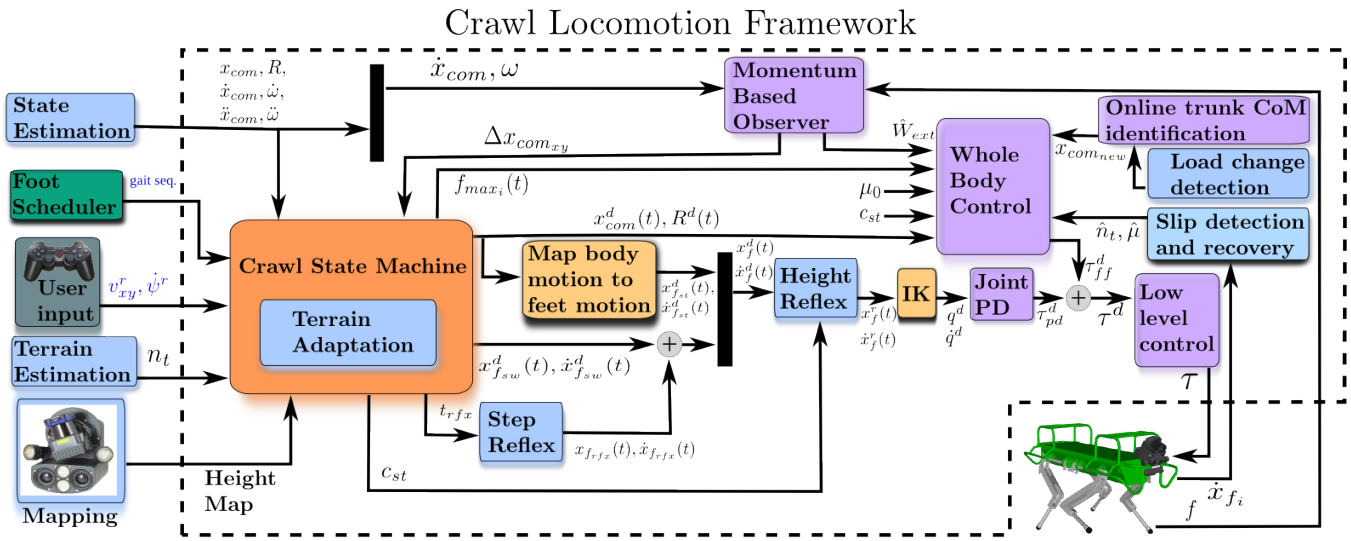

Fig. 3. Block diagram of the crawl locomotion framework. The crawl state machine (orange block) is further detailed in Fig. 5. Please refer to *AppendixA* for a description of the main symbols shown in this figure.

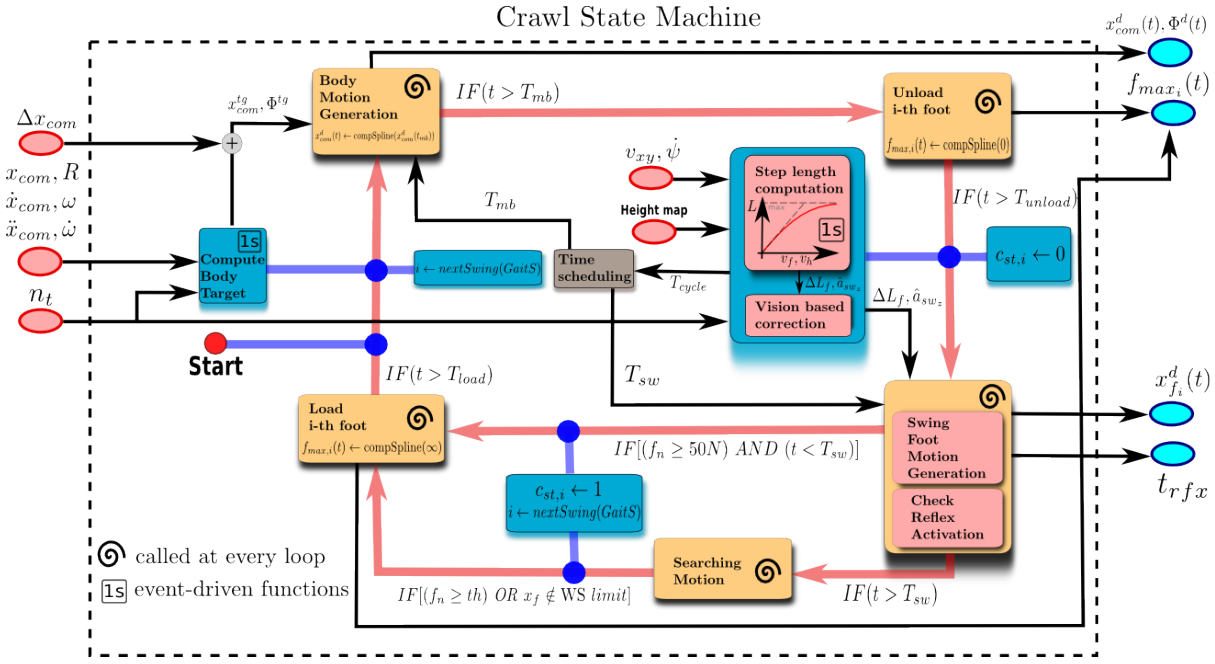

Fig. 5. Logic diagram of the crawl state machine. The yellow rectangles represent the states, the red arrows represent the transitions while the blue boxes represent the actions associated to the transitions. Blocks marked with 1s (1-shot) are event-driven and perform a single computation, while the other blocks (marked with the spiral) are called every loop. Please refer to *AppendixA* for a short description of the main symbols shown in this figure.

To address unpredictable events (*e.g.,* limit slippage on an unknown surface, whenever the optimized force is out of the real (unknown) friction cone) we implemented an impedance controller at the joint level [34][3] in parallel to the whole body controller. The impedance controller computes the feedback joint torques $\tau_{fb} \in \mathbb{R}^n$ (where for HyQ $n = 12$ is the number of active joints) to track reference joint trajectories ($q_j^d,\ \dot{q}_j^d \in \mathbb{R}^n$). The trunk controller output $\tau_{ff}^d \in \mathbb{R}^n$ is added to the contribution of the impedance controller $\tau_{pd}^d \in \mathbb{R}^n$ to form the torque reference $\tau^d \in \mathbb{R}^n$. This is then sent to the low level torque controller. If the model is accurate, the biggest term typically comes from the trunk controller.

Note that the impedance controller should receive position and velocity set-points that are consistent with the body motion, to prevent conflicts with the Trunk Controller. To achieve this, we map the CoM motion into feet motion (see Section III) to provide the correspondent joint references $q_j^d,\ \dot{q}_j^d$.

[3]Without any loss of generality, the same controller can be implemented at the foot level. However, to avoid non collocation problems due to leg compliance, it is safer to close the loop at the joint level.

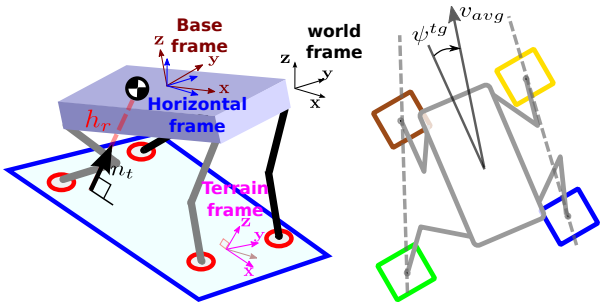

Fig. 4. (left) Definition of the relevant reference frames and of the terrain plane (light blue) used in the locomotion framework and (right) computation of the target ψ^{tg} for the robot yaw.

III. Body Motion Phase

When traversing a rough terrain, unstable footholds (*e.g.,* stones rolling under the feet), tracking errors, slippage and estimation errors can create deviations from the original plan. These deviations require some corrective actions in order to achieve *terrain adaptation*. In our framework, these actions are taken both at the beginning of the swing and of the *body motion phase*. The body motion phase starts with a foot touchdown and ends when the next foot in the sequence is lifted off the ground. After each touchdown event, we compute the target for the CoM position x^{tg}_{com} and the body orientation Φ^{tg} from the *actual* robot state. This feature prevents error accumulation and, together with the haptic touchdown, constitutes the core the so-called *terrain adaptation* feature.

To avoid hitting the kinematic limits, the robot's orientation should be adapted to match the terrain shape. Indeed, some footholds can only be reached by tilting the base, because constraining the base in a given orientation restricts the range of achievable motions.

We chose to parametrize the trajectory for the trunk orientation with Euler angles[4] $\Phi^d(t) = [\phi(t), \theta(t), \psi(t)]$ that represent roll, pitch and yaw, respectively. We chose as trajectories quintic $3D$ polynomials to connect the actual robot orientation (at the beginning of the phase, namely the touchdown) $\Phi^d(0) = [\phi_{td}, \theta_{td}, \psi_{td}]$ to a desired orientation $\Phi^d(T_{mb}) = \Phi^{tg} = [\phi^{tg}, \theta^{tg}, \psi^{tg}]$, where T_{mb} is the duration of the *move body phase*. To match the inclination of the terrain, the target roll and pitch are set equal to the orientation of the terrain plane that was updated at the touchdown ($\phi^{tg} = \phi_t, \theta^{tg} = \theta_t$). The target yaw ψ^{tg} is computed to align the trunk with an average line v_{avg} from the ipsilateral[5] legs (see Fig. 4 (right)). This is somewhat similar to a heading controller that makes sure that the trunk "follows" the motion of the feet (we will see that the feet motion is, on his behalf, driven by the desired velocity set by the user).

Similarly to the angular case, the CoM trajectory is initialized with the *actual* position of the CoM[6], while the target x^{tg}_{com} and can be computed with different stability criterion (*e.g.,* a ZMP-based [1], wrench-based [35], [36]). Hereafter, the vectors are expressed in the world (fixed) frame $\mathcal{W}$, unless otherwise specified. Since the crawl does not involve highly dynamic motions, a heuristic *static* stability criterion can also be used; it consists of a projection of the CoM target $x^{tg}_{com_p}$, inside the future support triangle [27]. The robustness (in terms of stability margin) can be regulated by setting the (projected) target at a distance d (along the support plane) from the middle point of the segment connecting the diagonal feet (see Fig. 6 (right)). We define the *robot height* $h_r \in \mathbb{R}$ as the distance between the CoM and the *terrain plane* (see Fig. 4 (left)). This is computed by averaging the actual positions of the feet ${}_bx_{fi}$ that are in contact with the ground:

$$h_r = e_z^T \frac{1}{c_{st}} \sum_{i=1}^{c_{st}} {}_tR_b({}_bx_{f_i} + {}_bx_{com}), \tag{1}$$

where c_{st} is the number of stance feet, ${}_bx_{com} \in \mathbb{R}^3$ is the CoM offset with respect to the base origin, ${}_tR_b \in SO(3)$ maps vectors from base to the terrain frame and $e_z \in \mathbb{R}^3$ selects the Z component of $3D$ vectors. In general, the robot height should be kept constant (or varying with the cosine of the terrain pitch θ_t on ramps) during locomotion. If the above-mentioned heuristics is used for planning, the CoM target x^{tg}_{com} can be obtained by adding the height vector $h_r n_t$ (in the world frame) to the projection $x^{tg}_{com_p}$ coming from the heuristics (where n_t is the normal to the terrain). Note that, on an inclined terrain, to have the CoM above the desired projection and the distance of the CoM from the terrain plane equal to h_r, the following scaling should be applied (see Fig. 6 (left)):

$$\begin{aligned} cos(\alpha) &= e_z^T n_t, \\ x^{tg}_{com} &= x^{tg}_{com_p} + \frac{h_r}{cos(\alpha)} e_z, \end{aligned} \tag{2}$$

where α is the angle between the vertical e_z and n_t that are unit vectors. Now, similarly to the orientation case, we build a quintic polynomial $x^d_{com}(t)$ to connect the *actual* CoM at the beginning of the motion phase (namely at the

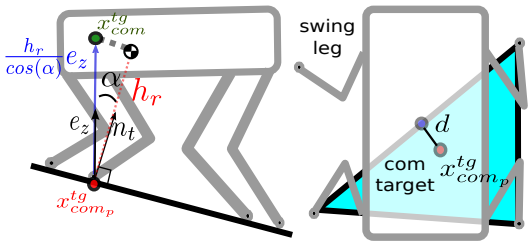

Fig. 6. Heuristic generation of the CoM target. (left) planning on inclined terrain, (right) the future support polygon is depicted in light blue while the projection $x^{tg}_{com_p}$ of the com target on the polygon is a red dot.

[4]This is a reasonable choice because the robot will never hit an orientation singularity (*i.e.,* 90° pitch).

[5]Belonging to the same side of the body.

[6]In experimental trials, we noticed that considering the *desired* feet position instead of the *actual* one for the computation of the CoM target would make the height of the robot decrease gradually.

touchdown) $x_{com,td}$ to the computed target x^{tg}_{com}. To determine the 6 parameters of each quintic, in addition to the initial/final positions we obtained, we enforce to zero the initial/final values for velocity and accelerations both for CoM and Euler angles. This ensures *static stability*[7], and implies that the robot's trunk will not move whenever one leg is lifted from the ground (e.g. during the swing phase). In alternative, if a wrench-based optimization is used (as in [35]), a constraint should be set on the robot height (e.g. to be constant) and the CoM trajectory will result from the optimization. As we mentioned in the previous section, it is convenient to provide a desired joint positions (of the stance legs) that are consistent with the body motion. We first map the body motion into feet motion. This mapping is linear in the velocity domain and can be computed separately for each stance foot i as:

$$\dot{x}^d_{f_i}[k] = -\dot{x}^d_{com}[k] - \omega[k] \times \left(x^d_{f_i}[k-1] - x_{com}([k]\right), \quad (3)$$

where the desired position of the foot $x^d_{f_i}[k-1]$ at the previous loop is used. Then, $x^d_{f_i}[k]$ is obtained by integrating the $\dot{x}^d_{f_i}[k]$ (*e.g.,* with a trapezoidal rule). Subsequently, we compute the corresponding joint desired positions $q_i \in \mathbb{R}^3$ through inverse kinematics: $q_i = IK(x_{f_i})$ and the joint velocities via: $\dot{q}^d_i = J(q_i)^{-1}\dot{x}_{f_i}$, where $J_i \in \mathbb{R}^{3\times3}$ is the Jacobian of foot i[8].

IV. Swing Phase

The swing phase of a leg starts with the foot lift-off and ends with the foot touch-down. The obvious role of a leg's swing phase is to establish a new foothold. Then, an *interaction force* drives the robot's trunk towards the desired direction. The swing phase has two main objectives: 1) attaining enough clearance to overcome potential obstacles and avoid *stumbling* and 2) achieving a stable contact.

A. Heuristic Stepping

In this section, we illustrate the heuristic *stepping strategy* that we use for *blind* locomotion. The goal is to select the footholds in order to realize a *desired* user speed when no map of the surroundings is available. First, we compute the (default) step length (for the swing leg) from the desired linear and heading velocities $v^r_{xy} \in \mathbb{R}^2$, $\dot{\psi}^r \in \mathbb{R}$. For sake of simplicity of notation, *only in this section* all the vectors are expressed in the *horizontal frame* [9] $\mathcal{H}$ (instead of the *world frame* $\mathcal{W}$), unless otherwise specified. Expressing the default step in a *horizontal frame* allows to formulate the swing motion independently from the orientation of both terrain and trunk.

[7]Having a statically stable gait is convenient for locomotion in dangereous environments (e.g. nuclear decommissioning missions) because the motion can be stopped in any moment. However, without any loss of generality the final velocity can be set to any value (e.g. the desired velocity).

[8]Note that we performed a simple inversion since in our robot we have point feet and 3 Degree of Freedom (DoF) per leg, thus the Jacobian matrix is square.

[9]The horizontal frame $\mathcal{H}$ is the reference frame that shares the same origin and yaw value with the base frame but is aligned (in pitch and roll) to the world frame, hence horizontal (see Fig 4 (left)).

The swing trajectory consists in a parametric curve whose four main parameters are the linear $\Delta L_{x0}, \Delta L_{y0}$ and angular ΔH_0 displacements of the foot during one step (see Fig. 7) and the default swing duration T_{sw}. These quantities can be obtained through the nonlinear mapping $F(\cdot)$:

$$\begin{bmatrix}\Delta L_{x0} & \Delta L_{y0} & \Delta H_0 & T_{sw}\end{bmatrix} = F(v^r_{xy}, \dot{\psi}^r). \quad (4)$$

$F(\cdot)$ makes sure that, at low speeds, to a velocity increment it corresponds linearly another increment of the step length (see *compute step length* block in Fig. 5). When the step length approaches the maximum value, the cycle time T_{cycle} (sum of swing and stance time of each leg) is decreased and the stepping frequency $f_s = 1/T_{cycle}$ is increased accordingly to avoid hitting the kinematic limits. For instance, for the X component, we can use the following equation:

$$A = \frac{2\Delta L^{max}_{x0}}{\pi}, \quad (5)$$

$$G = \frac{\Delta L^{tr}_{x0}}{\Delta L^{max}_{x0} v_{tr}}, \quad (6)$$

$$\Delta L_{x0} = A \cdot \arctan(G v^r_x), \quad (7)$$

where ΔL^{max}_{x0} is the maximum allowable step length in X direction. According to Eq. (7), ΔL_{x0} linearly increases with velocity up to the transition point ΔL^{tr}_{x0}, which corresponds to the user-defined value v_{tr}. Then, the step length is increased less than linearly with velocity (because the stepping frequency is also increased), up to the saturation point ΔL^{max}_{x0}. After this point, only the frequency increases. Similar computations are performed for ΔL_{y0} and ΔH_0. Since it is possible to set different values for heading and linear speed, the cycle time (and so the stepping frequency) is adjusted to the minimum coming from the 3 velocity components:

$$T_{cycle} = min\left(\frac{\Delta L_{x0}}{v^r_x}, \quad \frac{\Delta L_{y0}}{v^r_y}, \quad \frac{\Delta H_0}{\dot{\psi}^r}\right). \quad (8)$$

When a new value of T_{cycle} is computed, the durations of the body and swing trajectories are recomputed as $T_{mb} = (T_{cycle} - T_{lu})D$ and $T_{sw} = (T_{cycle} - T_{lu})(1-D)$ respectively, according to the *duty factor* D [37]. The variable T_{lu} represents the cumulative duration of the load/unload phases. As explained in [27], we remark that a load/unload phase at the touchdown/lift-off moments is important to avoid torque discontinuities. In particular, a *load phase* at the touchdown allows the trunk controller to redistribute smoothly the load on all the legs. At the same time, it ensures that the GRF always stay inside the friction cones, thus reducing the possibility of slippage.

Note that a heading displacement ΔH_0 can be converted into a X,Y displacement of the foot, as shown in Fig. 7, by:

$$\Delta L_{h0} = E_{xy}\begin{bmatrix}0 & 0 & \Delta H_0\end{bmatrix}^T \times x_{\text{hip}} \quad (9)$$

where $x_{\text{hip}} \in \mathbb{R}^3$ is the vector from the origin of the base frame to the hip of the swinging leg, and $E_{xy} \in \mathbb{R}^{2\times3}$ selects the X,Y components. Then, the default step becomes:

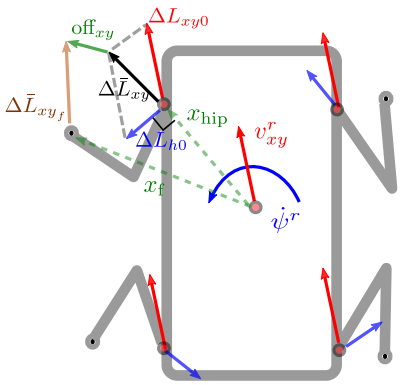

Fig. 7. Vector definitions for the heuristic stepping strategy. Red arrows represent the desired linear velocity v^r_{xy} and the linear component ΔL_{xy0} of the default step. In blue is the desired heading velocity ψ^r and the angular component ΔL_{h0} of the default step $\Delta \bar{L}_{xy0}$ that is in black. The offset to increase/decrease the stance size is in green, The step about the foot ΔL_{xy} is depicted in brown.

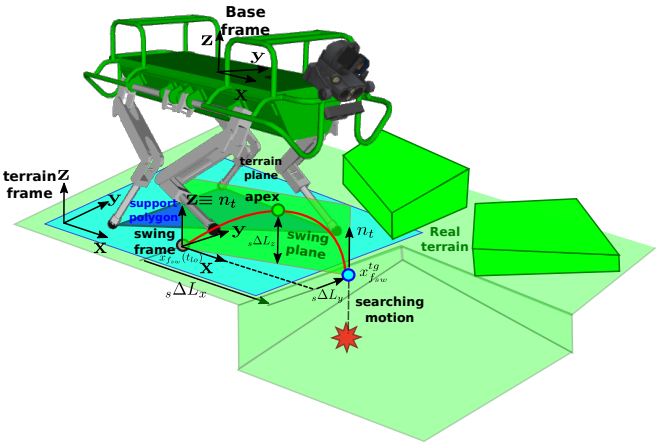

Fig. 8. Swing frame and terrain frame definition for the heuristic-based stepping strategy used in blind locomotion. The terrain presents a decrease in elevation, haptically handled by the searching motion.

$$\Delta \bar{L}_{xy0} = \Delta L_{xy0} + \Delta L_{h0}. \tag{10}$$

Note from the figure that we defined the *default* step about the hip rather than the foot position. This is crucial to avoid inconvenient kinematic configurations while walking (*e.g.,* stretched or "crouched" configuration), thus degenerating the support polygon. Indeed, if we refer each step to the previous foot position, an anticipated/delayed touchdown would produce unexpected step lengths. This would make the stance feet get closer/farther with time[10]. From $\Delta \bar{L}_{xy0}$, since the swing polynomial is defined at the foot level, we have to determine the step $\Delta L_{xy} \in \mathbb{R}^2$ to take about the actual foot position,

$$\Delta L_{xy} = \Delta \bar{L}_{xy0} + E_{xy}(\text{off}_{xy} + x_{hip} - x_f), \tag{11}$$

where off_{xy} is an offset which allows the user to adjust the average size of the stance polygon. In delicate situations, it might be useful to walk with the feet more outward to increase locomotion stability at the price of a bigger demanded torque at the Hip Abduction/Adduction (HAA) joint.

As a final step, it is convenient to express the swing motion in the *swing plane* (see Fig. 8)[11]. This means we need to express the above quantities in the swing frame $\mathcal{S}$[12]. Therefore, we set quintic $3D$ polynomials $p(t) \in \mathbb{R}^3$ of duration T_{sw}, where the X, Y components go from $(0,0)$ to ${}_s\Delta L_{xy}$, while for the Z component we set two polynomials such that the swing trajectory passes by an intermediate *apex* point at a time $T_{sw}\chi$. At the apex, the Z component is equal to the step height ${}_s\Delta L_z$) and $\chi \in [0,1]$ represents the apex ratio that can be adjusted to change the apex location (see Fig. 8).

Hence, the swing foot reference trajectory is computed from the *actual* foot position at the instant of lift-off to the desired foothold as:

$$x^d_{f_{sw}}(\bar{t}) = x_{f_{sw},lo} + p(\bar{t}), \tag{12}$$

where $\bar{t} \in \begin{bmatrix} 0 & T_{sw} \end{bmatrix}$ and T_{sw} is the swing duration, and the final target is defined as $x^{tg}_{f_{sw}} = x_{f_{sw},lo} + p(T_{sw})$.

Remark: the *swing frame* (unless specified) is always aligned with the *terrain frame*. Consequently, we have an initial *retraction* along the normal to the *terrain plane* (thus avoiding possible trapping or stumbling of the foot), while the step is performed *along* the terrain plane. Having the swing motion expressed this way allows also to simply adjust the step clearance with the step height ${}_s\Delta L_z$. This parameter regulates the maximum retraction from the terrain (apex point). In general, the apex is located in the middle of the swing, but it can be parametrized to shape differently the swing trajectory.

1) Searching motion: Haptic triggering of the touchdown allows to accommodate the shape of the terrain by stopping the swing motion either before of after the foot reaches the target $x^{tg}_{f_{sw}}$. If the touchdown is deemed to happen after the target is reached, the trajectory is continued with a *searching* motion: the swing foot keeps moving (linearly) along the direction of the terrain plane normal n_t (see Fig. 8) until it touches the ground or eventually reaches the workspace (WS) limit. In this way, the stance is triggered in any case. To avoid mobility loss, a height reflex can be enabled to aid the *searching* motion with the other stance legs (see section V-B).

B. Vision Based Stepping

The *terrain plane* is a very coarse approximation of the terrain. If a vision feedback is available, it is advisable to

[10]As a matter of fact, if the support polygon shrinks, the robustness decreases. In particular, CoM tracking errors and external pushes can make the ZMP get very close to the support polygon boundary. This would result in a situation where not all the contact forces are "pushing" on the ground (*e.g.,* the robot starts tipping about the line connecting two feet).

[11]We call the *swing plane* the plane passing though the X-Z axes of the swing frame.

[12]The *swing frame*, in general, is aligned with the *terrain frame* unless a vision based stepping strategy is used. In this case the swing frame is *independently* computed for each foot (as explained in Section IV-B) from the visual input.

exploit this information, because it enables several improvements:

1) to correct the desired foothold by computing target foothold on the *real* terrain rather than on its planar approximation (see Fig. IV-B). This allows to increase the overall swing *clearance* (compare blind stepping in Fig. 8 with vision based stepping in Fig. IV-B).
2) instead of having the *swing frame* coincident to the *terrain frame*, it is possible to set a different swing frame for each foot. This enables to perform the swing on different planes for each leg (this is mandatory for walking though peculiar features shapes like two V-shaped walls [27]).

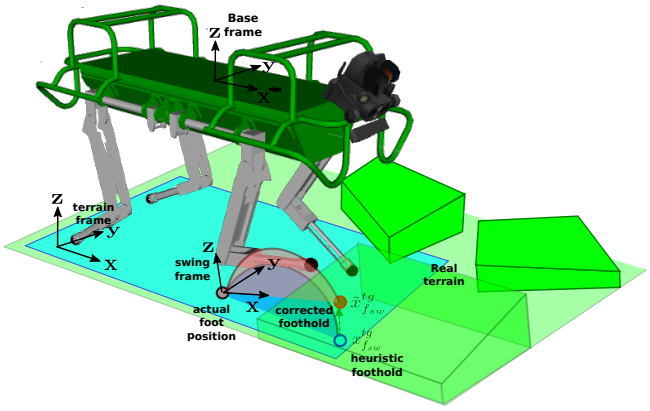

Fig. 9. Swing frame definition for the vision-based stepping strategy. Blue and red shaded area represent the swing in the heuristic and the vision-based cases, respectively. In this case the terrain elevation is higher than the foot location at the lift-off and the target foot-hold (blue dot) computed with heuristics, is corrected by vision (red dot) to lay on the real terrain. The swing frame is also adjusted accordingly.

To achieve the first point, we first compute a step (along the terrain plane) using the heuristic-based stepping strategy. This is meant to realize the user velocity. Then the height map of the terrain is queried at the target location and the Z component of the target is corrected to have it on the real terrain (X, Y components remain unchanged):

$$\tilde{x}^{tg}_{z} = H\left(E_{xy} x^{tg}_{f_{sw}}\right), \tag{13}$$

where $E_{xy} \in \mathbb{R}^{2\times 3}$ selects the X,Y components, and $H(.)$ is a function that queries height of the terrain for a certain (X,Y) location in the X-Y plane[13].

As a final step, by simple geometric calculations, we compute the Z axis $\hat{a}_{sw_z}$ of the swing frame such that its X axis $\hat{a}_{sw_x}$ is aligned with the segment connecting the actual foot position to the corrected foothold $\tilde{x}^{tg}_{f_{sw}}$ (see Fig 10).

[13]Due to the map is expressed in a (fixed) world frame, it is necessary to apply appropriate kinematic conversions to this frame before evaluating the map.

$$\begin{aligned} a_{sw_z} &= (\tilde{x}^{tg}_{f_{sw}} - x_{f_{sw}}) \times ({}_wR_b e_y), \\ \hat{a}_{sw_z} &= \frac{a_{sw_z}}{\|a_{sw_z}\|}, \\ \hat{a}_{sw_y} &= \hat{a}_{sw_z} \times e_x, \\ \hat{a}_{sw_x} &= \hat{a}_{sw_y} \times \hat{a}_{sw_z}, \end{aligned} \tag{14}$$

where the $\hat{(.)}$ represents unit vectors and e_x, e_y are the base frame axes (expressed in the world frame). This correction it allows to achieve more clearance about the real terrain during the swing motion and reduces the chances of stumbling. The first consequence is that the swing frame is no longer aligned with the terrain frame, and the swing trajectory, will now happen in the plane $\hat{a}_{sw_x}/\hat{a}_{sw_z}$. Additionally, if the normal n_r of the *real* terrain is available from the visual feedback [26], we can exploit this to further correct the swing plane, in order to have the swing-down trajectory approaching the real terrain *along* its normal. This is particularly useful when the robot has to step on a laterally slanted surface (e.g. like in [27]).

However, in our experience we noticed that, along the sagittal direction, it is more important to maximize the clearance. Therefore we remove the component of n_r along the X axis computed in (14) ($\hat{a}^{v}_{sw_z} = n_r - \hat{a}^{T}_{sw_x} n_r) \hat{a}_{sw_x}$) and use as the new swing Z axis. The new vision-corrected Z-axis of the *swing frame* it becomes n_{rp} (see Fig. 10), while the other axes should be recomputed accordingly as in (14).

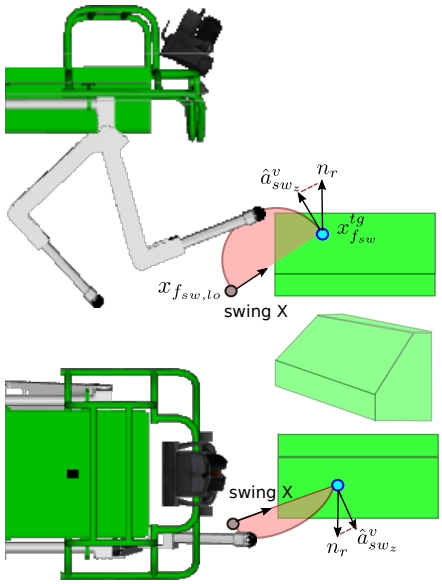

Fig. 10. Vision based adjustment of the target foothold Z coordinate. The swing plane on slanted terrain (top and lateral views).

C. Clearance Optimization

If the quality of the map is high enough, it is possible to adjust the *apex* (point of maximum clearance) and the

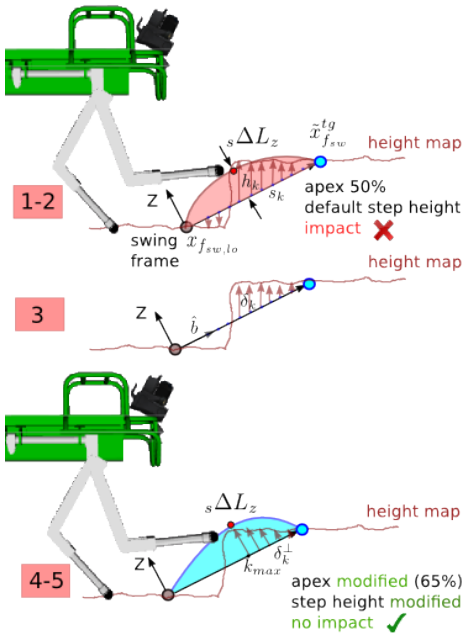

Fig. 11. Clearance optimization. The grey dot represents the actual foot location at lift-off, the blue dot the foot target while the red dot the apex location. The numbers in the red rectangles are related to the steps in (15).

step height ${}_s\Delta L_z$ in accordance to the point of maximum asperity of the terrain, in order to optimize for the clearance along the swing. This is a very simple heuristic, that, in some extent, adapts the swing to the terrain shape. First we take a slice of the map and evaluate the height of the terrain on a line segment (direction $\hat{b}$) connecting the actual foot location with the target $\tilde{x}^{tg}_{f_{sw}}$ (see Fig. 11). Using a discretization of N points we follow these steps:

$$
\begin{aligned}
&1)\ s_k = \tilde{x}^{tg}_{f_{sw}} \frac{k}{N} + \tilde{x}_{f_i}\left(1 - \frac{k}{N}\right), \quad k = 1...N,\\
&2)\ h_k = H(E_{xy} s_k),\\
&3)\ \delta_k = \mathcal{P}_+(h_k - e_z^T s_k),\\
&4)\ \delta_k^{\perp} = \left\| \begin{bmatrix} 0 & 0 & \delta_k \end{bmatrix}^T - \left(\begin{bmatrix} 0 & 0 & \delta_k \end{bmatrix} \hat{b} \right) \hat{b} \right\|,\\
&5)\ \begin{bmatrix} {}_s\Delta L_z & k_{max} \end{bmatrix} = \max_k \delta_k^{\perp},
\end{aligned}
\tag{15}
$$

where 1) is a discretization of the line segment, 2) is the evaluation of the height map on the discretized points, 3) $\mathcal{P}_+(.) : \mathbb{R} \to \mathbb{R}$ is a function made such that it reflects the input if this is positive and outputs zero if this is negative. In 4) we project the δ_k onto the swing Z axis, getting $\delta_k^{\perp}$. In 5) we set the step height ${}_s\Delta L_{f_{sw}}$ as the maximum value among the $\delta_k^{\perp}$ where k_{max} is its index. Finally, the apex can be set to k_{max}/N. This reshapes the swing in a conservative way, by adjusting the apex according to the terrain feature.

D. Time Rescheduling

In our state-machine-based framework, varying the locomotion speed, it means changing the duration of the swing/body phases. However, to have a change in speed promptly reflected in the robot motion (without waiting until the end of the current phase) it is necessary to reschedule the polynomials of the active phase (swing or body motion). As a consequence, we obtain a new duration $T_f^{'}$ (where we call T_f the previous duration). Setting a new duration for a (previously designed) polynomial it is equivalent to find the *initial point* of a new polynomial that: 1) has the new duration $T_f^{'}$, 2) is passing through the same point at the moment of the rescheduling. We know that a quintic polynomial can be expressed as:

$$
p(t) = a^T \mu(t), \quad \dot{p}(t) = a^T \dot{\mu}(t), \quad \ddot{p}(t) = a^T, \ddot{\mu}(t) \tag{16}
$$

where:

$$
\begin{aligned}
a^T &= \begin{bmatrix} a_5 & a_4 & a_3 & a_2 & a_1 & a_0 \end{bmatrix},\\
\mu(t) &= \begin{bmatrix} t^5 & t^4 & t^3 & t^2 & t & 1 \end{bmatrix},\\
\dot{\mu}(t) &= \begin{bmatrix} 5t^4 & 4t^3 & 3t^2 & 2t & 1 & 0 \end{bmatrix},\\
\ddot{\mu}(t) &= \begin{bmatrix} 20t^3 & 12t^2 & 6t & 2 & 0 & 0 \end{bmatrix}.
\end{aligned}
\tag{17}
$$

Setting the initial/final boundary conditions for position, velocity and acceleration: $p_0 = a^T \mu(0)$, $\dot{p}_0 = a^T \dot{\mu}(0)$, $\ddot{p}_0 = a^T \ddot{\mu}(0)$, $p_f = a^T \mu(T_f)$, $\dot{p}_f = a^T \dot{\mu}(T_f)$, $\ddot{p}_f = a^T \ddot{\mu}(T_f)$ it is equivalent to solve a linear system of 6 equations that allows us to find the a_i parameters:

$$
\begin{aligned}
a_0 &= p_0,\\
a_1 &= \dot{p}_0,\\
a_2 &= 0.5\ddot{p}_0,\\
a_3 &= \frac{1}{2T_f^3}\left[20p_f - 20p_0 + T_f(-12\dot{p}_0 - 8\dot{p}_f) + T_f^2(-3\ddot{p}_0 + \ddot{p}_f)\right],\\
a_4 &= \frac{1}{2T_f^4}\left[30p_0 - 30p_f + T_f(16\dot{p}_0 + 14\dot{p}_f) + T_f^2(3\dot{p}_0 - 2\dot{p}_f)\right],\\
a_5 &= \frac{1}{2T_f^5}\left[12p_f - 12p_0 + T_f(-6\dot{p}_0 - 6\dot{p}_f) + T_f^2(-\ddot{p}_0 + \ddot{p}_f)\right],
\end{aligned}
\tag{18}
$$

with a similar reasoning, to ensure continuity in the position, we can compute the new initial point p_0' such that:

$$
a' \mu(\bar{t}) = p(\bar{t}), \tag{19}
$$

where $\bar{t}$ is the time elapsed (from 0) at the moment of the rescheduling, and a' are the coefficient of the new polynomial of duration $T_f^{'}$. Then collecting p_0 from all the parameters in (18) we can obtain a closed form expression:

$$
\begin{aligned}
\beta_1 &= \frac{1}{2T_f^3}\left[20p_f + T_f(-12\dot{p}_0 - 8\dot{p}_f) + T_f^2(-3\ddot{p}_0 + \ddot{p}_f)\right],\\
\beta_2 &= \frac{1}{2T_f^4}\left[-30p_f + T_f(16\dot{p}_0 + 14\dot{p}_f) + T_f^2(3\dot{p}_0 - 2\dot{p}_f)\right],\\
\beta_3 &= \frac{1}{2T_f^5}\left[12p_f + T_f(-6\dot{p}_0 - 6\dot{p}_f) + T_f^2(-\ddot{p}_0 + \ddot{p}_f)\right],\\
\beta_4 &= 1 - 10T_f^3\bar{t}^3 + 15T_f^4\bar{t}^4 - 6T_f^5\bar{t}^5,\\
p_0 &= \frac{1}{\beta_4}\left[p(\bar{t}) - a_1\bar{t} + a_2\bar{t}^2 + \beta_1\bar{t}^3 + \beta_2\bar{t}^4 + \beta_3\bar{t}^5\right]
\end{aligned}
\tag{20}
$$

Now the new polynomial parameters can be recomputed as in (17) exploiting p_0, computed in (20), while keeping the other boundary conditions unchanged. This will result in a

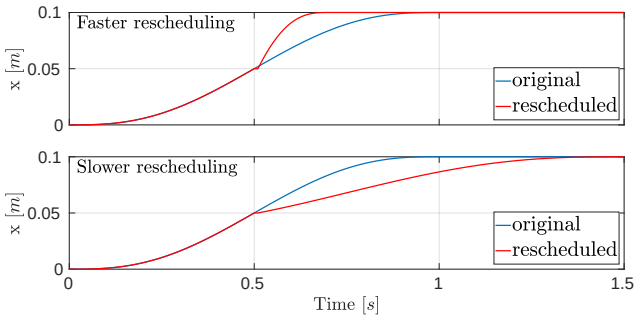

Fig. 12. Time rescheduling. The original trajectory (blue) at $\bar{t} = 0.5s$ is rescheduled into a new trajectory (red) of duration of (Upper plot) $0.7s$ or (lower plot) $1.5s$.

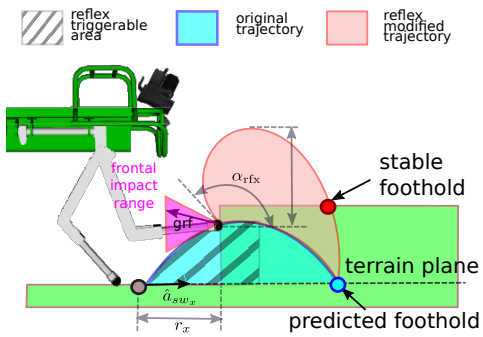

Fig. 13. Step Reflex. In shaded blue the original trajectory while in shaded red is the modification due to the reflex. The red cone, shows the range of GRF that are eligible as frontal impacts. Out of this cone the GRF will be used to trigger the stance. Area of possible activation along the trajectory, is depicted with stripes.

polynomial that *continues* from $\bar{t}$ with a new duration $T_f^{'}$. In Fig. 12 we show an example of time rescheduling happening at $0.5s$, where the duration of an original trajectory $T_f = 1s$ is reduced to $T_f^{'} = 0.7s$ (fast scheduling) or increased to $T_f^{'} = 1.5s$ (slower scheduling).

V. Reactive Behaviors

In this section, we briefly describe the framework's reactive modules for robust locomotion. These modules implement strategies to mitigate the negative effects due to unpredictable events, such as: 1) slippage (Section V-A); 2) loss of mobility (Section V-B); 3) frontal impacts (see Section V-C). 4) unexpected contacts (*e.g.,* shin collisions, see Section V-D).

A. Slip Detection

The causes of slippage during locomotion can be divided into three categories: 1) wrong estimation of the terrain normal; 2) wrong estimation of the friction coefficient; 3) external disturbances.

In our previous work [23], we have addressed the first and the second category. In that work, we proposed a slippage detection algorithm which estimates *online* the friction coefficient and the normal to the terrain. After the estimation, the coefficient were passed to the whole-body controller (see slip detection and recovery module in Fig. 3). Concerning the third category, an external push can create a loss of contact, and, in this can cause the foot moving very quickly inward. For instance, the trunk controller, depending on the regularization, can create internal forces, that in case of loss of contact, make the foot move away from the desired position. This can result in a catastrophic situation, because the tracking error can become very big. Thanks to the impedance controller (PD) running in parallel to the Trunk Controller, This error will be limited and can be recovered at the next replanning stage (see Section III).

B. Height Reflex

The *height reflex* is a motion generation strategy for all the robot's feet. It redistributes the *swing* motion onto the *stance* legs, with the final effect of "lowering" the trunk to assist the foothold *searching* motion.

The *height reflex* is useful when the robot is facing considerable changes in the terrain elevation (*e.g.,* stepping down from a high platform) [28]. In such situations, the swing leg can lose mobility, causing issues during the subsequent steps (*e.g.,* walking with excessively stretched legs). For this reasons the height reflex is most likely to be activated when stepping down.

C. Step Reflex

The *step reflex* [29] is a local elevator recovery strategy, triggered in cases of frontal impacts with an obstacle during the swing up phase. This reflex is key in cases of visual deprivation (*e.g.,* smoky areas or thick vegetation): it allows the robot to overcome an obstacle and establish a stable foothold at the same time, without *stumbling*.

Since the step reflex can be enabled only during the *swing* phase, it is important to set its duration T_{rfx} to stop concurrently with the end of the default swing duration:

$$T_{\text{rfx}} = T_{sw} - \bar{t}, \tag{21}$$

where $\bar{t}$ is the time elapsed from the beginning of the swing until the moment the reflex is tr0iggered (the searching motion will be still possible to accommodate for further errors). The angle of retraction α_{rfx} (see Fig. 13) depends on the distance r_x (along the swing X axis $\hat{a}_{sw_x}$ of the swing frame) already covered by the swing foot:

$$r_x = \hat{a}_{sw_x}^T (x_{f_{sw}}^d(\bar{t}) - x_{f_{sw}}^d(0)), \tag{22}$$

and the reflex maximum vertical retraction r_z:

$$\alpha_{\text{rfx}} = \text{atan2}(r_z, r_x).$$

The maximum vertical retraction r_z is an input parameter related to the step height and the roughness of the terrain. Note that, for convenience, the step reflex is also generated in the *swing frame*.

The reflex can be triggered only when the impact is frontal (GRF in the purple cone of Fig. 13), and only during the swing *up* phase (*e.g.,* before the apex point). The reason for these constraints is twofold:

1) a force from a frontal impact is larger (and pointing downwards) if the impact occurs in the swing up phase; in contrast, it is lower (and pointing upwards) in the swing down phase. Therefore, a small and upward force would cause an increment of false positives for a regular touchdown detection.
2) the time available for a reflex after the swing apex is more limited and would require significantly higher accelerations to be executed.

Missed reflex: If a frontal impact is detected in the swing down phase, the reflex is not triggered immediately, but it is scheduled for the beginning of the next swing phase.

The accompanying video[14]

shows a simulation where the robot performs stair climbing blindly, using the heuristic-based stepping strategy described in Section IV-A). Even if the swing leg stumbles against the step, the robot is still able to climb the stairs, thanks to the step reflex. Conversely, when the step reflex is *disabled*, the robot gets stuck.

Remark: the reflex is omni-directional and depends on the desired locomotion speed. For instance, if the robot walks sideways, the step will be triggered along the swing-plane that will be oriented in the transversal direction.

D. Shin Collision

The foot may not be the only point of contact with the terrain. For certain configurations, the shin of a quadruped robot can collide together with the foot (as shown in this simulation [15]). Due to the fact that contact forces directly influence the dynamics of the robot, we take this into account in the trunk controller, *i.e.,* realizing an appropriate a weight redistribution by setting the modified/added contact points. This requires the detection of the contact point location, either with a dedicated sensor or an estimation algorithm. On the same line, to improve the overall tracking, we also updated the contact points in the generation of the body trajectories.

E. Experimental Results

In this section, we present the experiments carried out on our robotic platform HyQ, to show the effectiveness of the proposed framework in addressing rough terrain locomotion. All the experiments have been carried out on HyQ, a 85 kg, fully-torque controlled, hydraulically actuated quadruped robot [38]. HyQ is equipped with a variety of sensors[16], including: precision joint encoders, force/torque sensors, a depth camera (Asus Xtion), a combined (stereo and LiDAR) vision sensor (MultiSense SL), and a tactical grade Inertial Measurement Unit (KVH 1775). The size of HyQ is 1.0 m × 0.5 m × 0.98 m (L × W × H). The leg's length ranges from 0.339 m to 0.789 m and the hip-to-hip distance is 0.75 m. HyQ has two onboard computers: a real-time compliant (Xenomai-Linux) used for locomotion and a non-RT machine to process vision data. The RT PC processes the low-level controller (hydraulic actuator controller) at 1 kHz and communicates with sensors and actuators through EtherCAT. Additionally, this PC runs the high-level controller at 250 Hz and the state estimation at 500 Hz. The non-RT PC processes the extereoceptive sensors to generate a 2.5-D terrain map [40] with 4 cm resolution and 3 m × 3 m size, surrounding the robot.

The template terrain used for the experiments is shown in Fig. 2. It is composed by: 1) ascending ramp; 2) a step-wise elevation change; 3) area with big stones (diameter up to 12 cm); 4) another step-wise elevation change; 5) a descending ramp with random bricks. Walking on stones and bricks is challenging from the locomotion point of view: the stones can collapse or roll away, causing a loss of balance, if specific strategies are not implemented. This video[17] shows the robot successfully traversing the template terrain. The terrain adaptation capabilities (*e.g.,* haptic feedback, searching motion) are mostly demonstrated when the robot is walking on rocks. The changes in elevation (where the *height reflex* is triggered) prevents the swing leg from losing mobility.

The importance of replanning is particularly evident during the star descent, where the robot step on bricks which roll away under a foothold, causing a pitch error which is recovered in the next step. Since the crawl is statically stable, the robot can move in extremely cluttered environments, almost at ground level. In the last scene of the video, we show a simulation the second generation of our robots, HyQ2Max [41], crawling inside a 45 cm wide duct.

VI. Terrain Estimation

The terrain estimation module (see Fig. 3) estimates the *terrain plane*, which is a linear approximation of the terrain enclosed by the stance feet.

The inclination of the terrain plane (ϕ_t,θ_t) is updated at each *touchdown* event (*e.g.,* a when a foot is "sampling" the terrain). Typically, this is done by fitting a plane through the stance feet and computing the principal directions of the matrix containing the feet positions. The fitting plane can be found in two ways:

- **Vertical Fit:** it minimizes the "distance" along the *vertical* component, between the fitting plane ($\Pi : ax + by + cz + d = 0$) and the set of points represented by the feet position;
- **Affine Fit:** it minimizes the Euclidean distance (along the terrain plane normal) between the feet and the plane.

In the first case, we have to solve a linear system, while the second is an *eigenvalue* problem.

A. Vertical Fit

The "vertical" distance can be minimized by setting the c coefficient to be equal to 1 and solve a *linear system* for the $x = \begin{bmatrix} a & b & d \end{bmatrix}$ parameters:

[14]Step reflex video: `http://www.youtube.com/watch?v=EUhw-aAFYrw&t=2m53s`

[15]Shin coll. video:`http://www.youtube.com/watch?v=EUhw-aAFYrw&t=4m24s`

[16]For a complete description of the sensor setup, see [39], Chapter 3

[17]Rough terrain experiments: `http://www.youtube.com/watch?v=EUhw-aAFYrw`

$$x = A^{\#} b, \tag{23}$$

where the positions of the stance feet have been collected in:

$$A = \begin{bmatrix} x_{f_{1x}} & x_{f_{1y}} & 1 \\ x_{f_{2x}} & x_{f_{2y}} & 1 \\ x_{f_{3x}} & x_{f_{3y}} & 1 \\ x_{f_{4x}} & x_{f_{4y}} & 1 \end{bmatrix}, \quad b = \begin{bmatrix} -x_{f_{1z}} \\ -x_{f_{2z}} \\ -x_{f_{3z}} \\ -x_{f_{4z}} \end{bmatrix}. \tag{24}$$

and where $[.]^{\#}$ is the Moore-Penrose pseudoinverse operator. Then, the normal to the terrain plane n_t and the corresponding roll/pitch angles ϕ_t, θ_t (in ZYX convention), can be obtained as[18]:

$$\begin{aligned} n_t &= \begin{bmatrix} a & b & 1 \end{bmatrix}^T / \| \begin{bmatrix} a & b & 1 \end{bmatrix}^T \|, \\ \theta_t &= \text{atan}(n_{t_x}/n_{t_z}), \\ \phi_t &= \text{atan}(-n_{t_y} \sin(\theta_t)/n_{t_x}). \end{aligned} \tag{25}$$

B. Affine Fit

In the affine case, we first need to reduce the feet samples by subtracting their average $\bar{x}$ (belonging to the fitting plane):

$$R = \begin{bmatrix} x_{f_1}^T - \bar{x}^T \\ x_{f_2}^T - \bar{x}^T \\ x_{f_3}^T - \bar{x}^T \\ x_{f_4}^T - \bar{x}^T \end{bmatrix}, \quad \bar{x} = \frac{1}{4} \sum_{i=1}^{4} x_{f_i}. \tag{26}$$

The principal directions of this set of samples are the eigenvectors of $R^T R$:

$$\begin{aligned} \begin{bmatrix} V & D \end{bmatrix} &= eig\left(R^T R\right) \\ n_t &= S_1 V \end{aligned} \tag{27}$$

where V is the matrix of the eigenvectors and D the diagonal matrix of the eigenvalues of $R^T R$.

We can obtain the terrain normal n_t by extracting the first column from the eigenvector matrix (*e.g.,* the eigenvector associated to the smallest eigenvalue). Then, similarly to the vertical fit case, Eq. (25) can be applied to find the terrain parameters. Note that the result is slightly different from the vertical fit case. Indeed, in the affine fit case, we minimize the Euclidean distance, while in the vertical fit case we minimize the distance along the z direction. On the other hand, the two approaches give the same result if the feet are coplanar.

It is noteworthy that the affine approach does not provide meaningful results when $R^T R$ is rank deficient. However, this happens only when (at least) three feet are aligned, which is a very unlikely situation.

C. Correction for Rough Terrain

There are some situations where just fitting an average plane is not the ideal thing to do during a statically stable motion. For instance, when the robot has three feet on the ground and one on a pallet, the *average* (fitting) plane is not horizontal. In this case, moving the CoM projection along the terrain plane (to enter in the support triangle) results in an inconvenient "up and down" motion video. This happens because the robot tries to follow the orientation of the terrain plane with its torso. In this case, it would be preferable to keep the posture horizontal until *at least* two feet are on the pallet.

In this section, we propose a more robust implementation of the terrain estimation strategy, which allows to address these particular situations. The idea is to have the terrain plane preferably to fit the *subset* of the stance feet that are closer to be *coplanar*. In this case, the influence of the "outlier" foot (*e.g.,* the one on the pallet) would be reduced. For more dynamic gaits, where the CoM trajectory is not planned (*e.g.,* like trotting [22]), a terrain estimate low-pass filtering would be sufficient to mitigate the effect of the outliers samples. However, since the crawl makes heavily use of the *terrain plane* to change the pose of the robot, a different strategy should be adopted.

A preliminary step (after computing the terrain normal n_t as in Section VI-A) is to compute the norm of the vector of least square errors. This can be easily done by exploiting the matrix computed in (24): $e_{LS} = \|Ax - b\|_2$. If e_{LS} (Least Square error) is bigger than a certain (user defined) threshold, it means that the feet are not *coplanar*, and n_t should be corrected.

First, we compute the normal vectors in common to all the combinations of two adjacent edges (l_i, l_j), (sorted in Counter Clock Wise (CCW) order, see Fig. 14):

$$n_{ij} = l_i \times l_j \quad (i,j) \in C, \tag{28}$$

where C is the set of all the combinations of two adjacent edges (sorted in CCW order). In our case, C has 4 elements. Due to two lines define a plane, we associate each normal n_{ij} is associated to one support triangle (*e.g.,* two edges of the support polygon, see Fig. 14 (left)).

The corrected terrain normal is a *weighted* average of n_{ij}, where the weights are proportional to "how close" each normal n_{ij} is from the terrain plane normal, computed at the previous touchdown event $n_{t_{\text{old}}}$.

$$cos(\alpha_{ij}) = n_{t_{old}}^T n_{ij}. \tag{29}$$

We compute the weight w_{ij} associated to each normal n_{ij} through the nonlinear function $W(\cdot) \in \mathbb{R} \rightarrow \mathbb{R}$ (Fig. 14 (right)), in accordance to its angular distance from $n_{t_{\text{old}}}$.

$$\begin{aligned} W(x) &= 1/\left(1 + s(x-1)^p\right), \\ w_k &= W(cos(\alpha_{ij})), \end{aligned} \tag{30}$$

where k is the index of the $ij-th$ element of the set C; and s is a sensitivity factor proportional to the LS error e_{LS}.

According to (30), the normals closer to $n_{t_{\text{old}}}$ (*e.g.,* cosine close to 1) are assigned a bigger weight (the weight is bounded to 1 by construction of $W(\cdot)$). The exponent p allows us to adjust the degree of nonlinearity of $W(\cdot)$ and the degree of correction (for us, it sufficed to set $p = 2$).

[18] Note that the normal vector [a,b,1] should be normalized for the computation of ϕ_t, θ_t.

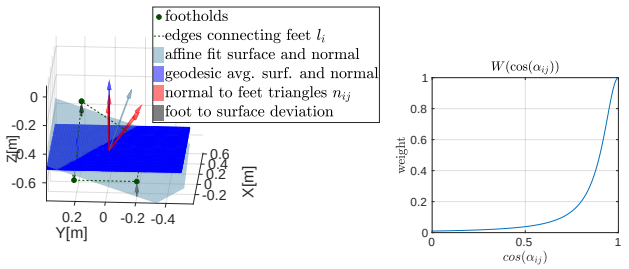

Fig. 14. Smart height correction. The blue arrow represent the corrected normal while the grey arrow the not corrected one (computed with the affine fit method). In this case almost coincident with one of the red arrows representing the normals n_{ij} to the triangles defined by three stance feet; (right) weight function (W) of the angular distance (*e.g.,* cosine) w.r.t $n_{t_{old}}$.

Since the normals n_{ij} are directly affected by the position of their corresponding feet on the pallet, a weighted average of the normals (with weights inversely proportional to the distance from the previous estimation) allows to naturally reduce the influence of the "outlier" foot. This promotes the terrain estimator not to modify much the previous estimate, in situations when a foot position is far away from the previous fitting plane.

Note that, since we are aiming at "averaging" orientations, we need to consider a Spherical Linear intERPolation (SLERP) using geodesic curves [42] (see Appendix IX).

D. Experimental Results

The video[19] shows how the smart terrain estimator improves locomotion by removing undesired up/down motions.

With reference to our initial example, as long as the robot has one foot on the pallet, the normal is corrected to be closer to the one provided by the other stance feet that are closer to the previous estimation where all the feet were on the flat ground (see Fig. 14). When the robot steps with two (lateral) feet on the pallet, then the fitting error e_{LS} is reduced due to the feet are more coplanar, and an inclined terrain plane is estimated.

The approach can address situations with non coplanar feet (*e.g.,* support has a "diamond" shape, shown later in the video). In this case, the terms from the feet on the pallet cancel each other, keeping the previous terrain plane estimate unchanged, which is the desirable behavior. For the single pallet example, we have $e_{LS} = 0.02$ (for the "diamond" shape $e_{LS} = 0.031$), therefore we set the threshold of intervention for the terrain correction to 0.002.

Regarding the sensitivity factor s inside the function $W(\cdot)$, since we want the correction to be stronger when the *LS* error is bigger, we linked s to be directly proportional to e_{LS}.

VII. Stair climbing

Stair climbing is an essential skill in the in the set of capabilities of a legged robot. Indeed, a *versatile* legged robot should be able to address both *unstructured* and *structured* environments, such as the one we can find in a disaster scenario.

The problem of stair climbing can be addressed through foothold planning, in order to avoid collisions with the step edges. An optimization taking into account the full kinematic of the robot could possibly solve the problem, but it is currently hard to be performed *online*. Moreover, as previously mentioned, a full optimization approach that plans for all the length of the stairs can be prone to tracking errors. These errors can result in missing the steps, thus jeopardizing the whole plan.

An alternative strategy is to conservatively select the foothold in the middle of the step (depth-wise). However, depending on the inclination of the stairs and on the step-size, the robot can end up in inconvenient configurations from the kinematic point of view (*e.g.,* with degeneration of the support triangle and associated loss of mobility).

In the case of HyQ, the kinematic limits at the Hip-Flexion-Extension joints (*i.e.,* hip joints rotating around the Y-axis, see [38]) are likely to be hit during the *body motion* phase[20]. According to our experience, keeping the joint posture as close as possible to the *default* configuration[21] it improves mobility, and is an important factor for the robustness of locomotion. Our heuristic approach, rather than avoiding potentially dangerous situations (*e.g.,* missed steps, collisions), aims to be robust enough to cope with them.

In particular, we will show that our vision-based stepping approach, with minor modifications (see section VII-B), is sufficient to successfully climb up/down industrial-size stairs. In case the *vision based swing strategy* is not sufficient to avoid frontal impacts (*e.g.,* against one step), the *step reflex* is triggered to achieve a stable foothold and prevent the robot from getting stuck.

A. Influence of the knee configuration

The chance of "getting stuck" when climbing stairs depends on the knee configuration (bent backwards or forward). In particular, when the leg has a Knee Bent Backward (KBB) configuration, it is more prone to have shin collision when climbing *downstairs*. Conversely, a Knee Bent Forward (KBF) configuration increases the risk of shin collision when climbing *upstairs*. These are important aspects for the design of robots that are expected to climb stairs.

Note that the contact forces at the shin are *in* the direction of motion when the configuration is KBB and the robot is climbing downstairs, whereas they point *against* the motion with the KBF configuration and the robot climbs upstairs (see Fig. 15). In the former case we might have slippage, but the robot eventually moves forward, while in the latter

[19]Terrain Estimation video:
`https://www.youtube.com/watch?v=EUhw-aAFYrw&t=5m25s`

[20]We adopt a "telescopic strut" strategy as in [43], which means that, on an inclined terrain, the vector between each stance foot and the corresponding hip aiming to be maintained parallel to gravity. On one hand, this improves the margin for a static equilibrium. On the other hand, for high stair inclinations, it can result in bigger joint motions, where the kinematic limits are most likely hit.

[21]As the one shown in Fig. 2 (left), where the joints are in the middle of their range of motion

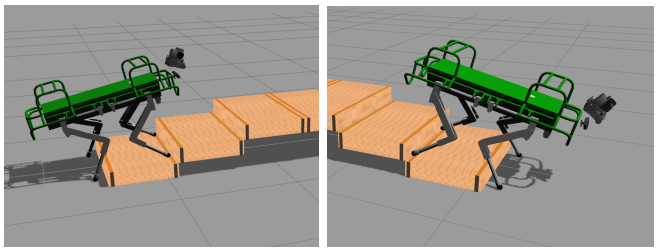

Fig. 15. Comparison of KBB and KBF configurations during stair climbing. In the KBF while climbing up the GRF point against the direction of motion making the robot get stuck.

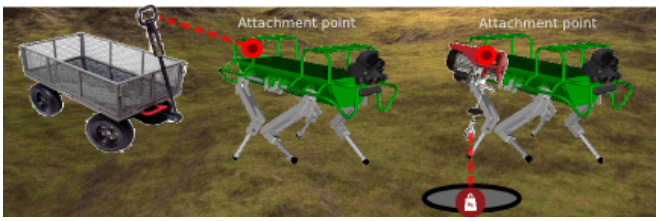

Fig. 16. Use cases for load estimation: (left) the robot is pulling a cart, (right) the robot is pulling up a payload.

case the robot might get stuck or fall backwards, unless these situation are properly dealt with (see Section V-D).

B. Stair locomotion mode

The *stair locomotion mode* can be triggered either by the user or by a dedicated stair detection algorithm [44]. It consists in the activation of three features (relevant for the task of climbing stairs) on top of the vision-based stepping strategy:

1) **Rescheduling of the gait sequence:** the kinematic configurations which cause mobility loss are undesired. To avoid them, it is important to climb stairs with both front feet (or back feet) on the same step. Specifically, the *next* foot target is checked at each touchdown (*i.e.,* before the *move body* phase). If this is on a different height than the foot on the opposite side, we perform a rescheduling of the gait sequence.
   For instance, if the last swing foot was the right front (RF) then, according to our *default* configuration (RH, RF, LH, LF)[22] the next leg to swing would be the LH. However, if the left-front (LF) foot is on a different height (*e.g.,* still on the previous step), the whole sequence is rescheduled to move the LF instead. The same applies for the back feet.
2) **Conservative stepping:** taking inspiration from the ideas presented in [24], this module corrects the foot location to step far away from an edge. The terrain flatness is checked along the direction of motion and the foothold is corrected (along the swing plane) in order to place it in a more conservative location (away from the step edge).
3) **Clearance optimization**: this feature (presented in Section IV-C) is useful to avoid the chance of stumbling against the step's edge. It adjusts the swing apex and the step height to maximize the clearance from the step.

C. Experimental Results

We successfully applied the reactive modules of our framework to the problem of climbing up (and down) stairs, in simulation, with our quadruped robot, HyQ[23]. In the video, we show that HyQ is able to climb up and down industrial size stairs (step raise 14 *cm*) and climbing up a staircase with a 90° turn.

Indeed, the approach is generic enough to be used also with irregular stair patterns (different step raise) and turning stairs. The user provides only a reference speed and heading.

In the simulation video we show the advantage of activating the stair locomotion mode and the importance of using a vision based stepping strategy.

In the 90° turn stair, we demonstrate omni-directional capability of our statically stable approach. Thanks to this, the robot is also able to move backward on the stair.

VIII. Momentum Based Disturbance Observer

A significant source of errors can come from unmodeled disturbances, such as an external push. Specifically, in the case of a model-based controller (*e.g.,* like our Whole Body controller [27]), inaccurate model parameters cause a wrong prediction of the joint torques. This shifts the responsibility of the control to the feedback-based controllers, thus increasing tracking errors and delays. However, if a proper identification is carried out *offline*, these model inaccuracies are mainly restricted to the trunk. Indeed, in the case of our quadruped robot, the leg inertia do not change significantly, while the trunk parameters are strongly dependent on the devices mounted on the robot (*e.g.,* a backpack, an additional computer, different sets of cameras for perception, *etc.*).

In [46], we presented a *recursive* strategy which performs *online* payload identification to estimate the new CoM position of the robot's trunk. The updated model is then used for a more accurate inversion of the dynamics [3], [27]. Even though this approach is effective to detect constant payload changes, it is not convenient to estimate *time-varying* unknown external forces, which might change both in *direction* and *intensity*. Indeed, this kind of disturbances can dramatically increase the tracking errors and jeopardize the locomotion, unless they are compensated *online*.

Hereafter, we mention two scenarios where an external disturbance observer is useful: 1) the robot is required to pull a cart or a load (*e.g.,* with some delicate material inside); and 2) the robot is requested to pull up a payload from underneath with a hoist mounted on the torso (see 16).

In this section, we present a MBDO able to estimate an external *wrench* (*e.g.,* cumulative effect of all the disturbance forces and moments). We also show how this *wrench* is

[22]This sequence is the one animals employ that reduces the backward motions [45].

[23]Stair climb video:
`https://www.youtube.com/watch?v=EUhw-aAFYrw&t=7m00s`

compensated during the locomotion, thus improving tracking accuracy and locomotion stability.

Our approach builds on top of the one described in [10], which is designed to estimate an external *linear* force acting on the robot base. We extended this work to the angular case, estimating the full wrench at the CoM.

One of the drawbacks of estimating only a *linear* force is that our robot has an angular dynamics that is not negligible. Therefore, unless the force is applied at the CoM, it requires the knowledge of its *application point*, to work out his moment about the CoM. Since we propose to estimate the full wrench, our approach is more general and this additional input not required.

The idea underlying the estimation is that any external *wrench*[24] $W_{ext} \in \mathbb{R}^6$ has an influence on the *centroidal* momentum (*i.e.,* linear and angular momentum at the CoM) [48]. We can observe that the discrepancy between the predicted (centroidal) spatial momentum $\hat{h}(t) \in \mathbb{R}^6$, based on *known* forces (*e.g.,* gravity and GRF), and the measured one $h(t)$ (coming from proprioceptive measurement of the CoM twist[25]), is caused only by an external wrench W_{ext}, in absence of modeling errors[26]. We can exploit this fact to design an observer.

To obtain a prediction of $\hat{h}(t) = \begin{bmatrix}\hat{p}_G(t) & \hat{k}_G(t)\end{bmatrix}$, we exploit the centroidal dynamics (*e.g.,* Newton-Euler equations) [48]:

$$\begin{cases} \dot{\hat{p}}_G(t) = \underbrace{mg + \sum_{i=1}^{c} f_i(t)}_{f_{\text{known}}} + \hat{f}_{ext}(t) \\ \dot{\hat{k}}_G(t) = \underbrace{\sum_{i=1}^{c} (x_{f_i}(t) - x_c(t)) \times f_i(t)}_{\tau_{\text{known}}} + \hat{\tau}_{ext}(t) \end{cases} \tag{31}$$

where $\dot{\hat{p}}_G(t) \in \mathbb{R}^3$ and $\dot{\hat{k}}_G(t) \in \mathbb{R}^3$ are the linear and angular momentum rate, respectively; $\hat{W}_{ext}(t) = \begin{bmatrix}\hat{f}_{ext}(t) & \hat{\tau}_{ext}(t)\end{bmatrix}$ is a prediction of the wrench disturbance at time t, expressed at the CoM point.

Starting from an initial measure of the momentum $h_0 = \begin{bmatrix}p_{G0} & k_{G0}\end{bmatrix} = \begin{bmatrix}m\dot{x}_{com}(0) & I_{com}(0)\omega(0)\end{bmatrix}$, we can get the predicted $\hat{h}(t)$, at a given time t, by integration of (31):

$$\begin{cases} \hat{p}_G(t) = p_0 + \int_0^t \left(mg + \sum_{i=1}^{c_{st}} f_i(t) + \hat{f}_{ext}(t)\right) dt \\ \hat{k}_G(t) = k_0 + \int_0^t (\sum_{i=1}^{c_{st}} (x_{f_i}(t) - x_{com}(t)) \times f_i(t) + \\ \qquad\qquad \hat{\tau}_{ext}(t)) dt \end{cases} \tag{32}$$

Then, the discrepancy between the measured and the predicted momentum can be used to estimate the external wrench disturbance, leading to the following observer set of equations:

$$\begin{cases} \dot{\hat{f}}_{ext}(t) = G_{\text{lin}} \left(m\dot{x}_{com}(t) - \hat{p_G}(t)\right) \\ \dot{\hat{\tau}}_{ext}(t) = G_{\text{ang}} \left(I_{com}(t)\omega(t) - \hat{k_G}(t)\right) \end{cases} \tag{33}$$

where the gains $G_{\text{lin}}, G_{\text{ang}} \in \mathbb{R}^{3\times3}$ are user-defined positive definite matrices, which describe the observer's dynamics, and $I_{com}(t) \in \mathbb{R}^{3\times3}$ is the rotational inertia of the robot (as a rigid body) computed at time t. As an alternative, (31) can be rewritten using spatial algebra [47], considering the whole robot as a rigid body:

$$\dot{h} = \frac{d}{dt}(\bar{I}_{com}v) = \bar{I}_{com}\dot{v} + v \times \bar{I}_{com}v = W_{\text{known}} + \hat{W}_{ext} \tag{34}$$

where $v = \begin{bmatrix}\dot{x}_{com} & \omega\end{bmatrix} \in \mathbb{R}^6$ is the measured CoM twist composed of CoM linear velocity and robot angular velocity (for simplicity of notation, we omit henceforth the dependency on t); $\bar{I}_{com} \in \mathbb{R}^{6\times6}$ is the composite rigid body inertia (expressed in an inertial frame attached to the CoM), evaluated at each loop at the actual configuration of the robot; $W_{\text{known}} = \begin{bmatrix}f_{\text{known}} & \tau_{\text{known}}\end{bmatrix}$ is the wrench due to contacts and gravity.

Using (34), it is possible to formulate a wrench observer where the nonlinear term $v \times \bar{I}_{com}v$ is compensated:

$$\begin{cases} \bar{I}\dot{\hat{v}} = \bar{I}_0 v_0 + \int_0^t \left(W_{\text{known}} + \hat{W}_{ext} - v \times \bar{I}_{com}v\right) dt \\ \hat{W}_{ext} = G\bar{I}(v - \hat{v}) \end{cases} \tag{35}$$

where $G = \text{diag}(G_{\text{lin}}, G_{\text{ang}}) \in \mathbb{R}^{6\times6}$ is the observer gain matrix.

At each control loop, after the *estimation* step, we perform *online* compensation of the estimated disturbance wrench $\hat{W}_{ext}$ in our whole body Trunk Controller [27] (see Fig. 3):

$$W^d = W_{vm} + W_g - \hat{W}_{ext} \tag{36}$$

where $W^d \in \mathbb{R}^6$ is the desired wrench (expressed at the CoM), mapped to desired joint torques by the Trunk Controller; $W_g, W_{vm} \in \mathbb{R}^6$ are the gravity compensation wrench and the virtual model attractor (which tracks a desired CoM trajectory) wrench, respectively.

A. ZMP compensation

Compensating for the external wrench is not sufficient to achieve stable locomotion. During a static crawl, the accelerations are typically small, and the ZMP mostly coincides with the projection of the CoM on the support polygon. However, this does not hold if an external disturbance is present. In this case, the ZMP can be shifted. If this is not properly accounted for in the body motion planning, the locomotion stability might be at risk.

Knowing that the ZMP is the point on the support polygon (or, better, the line) where the tangential moments nullify, the shift Δx_{com} can be estimated by computing the equilibrium of moments about this point (see Fig. 17):

$$\underbrace{(x_{com} - x_{zmp})}_{\Delta x_{com}} \times (mg + f_{ext}) + \tau_{ext} = 0 \tag{37}$$

[24]Henceforth, for simplicity, we talk about coordinate vectors and not spatial vectors, that is why $W_{ext} \in \mathbb{R}^6$ rather than $W_{ext} \in F^6$ [47]

[25]The twist (*i.e.,* 6D spatial velocity) is usually computed by a state estimator, which fuses Inertial Measurement Unit (imu), encoder and force/torque sensors [18].

[26]It is well known that it is impossible to distinguish between a CoM offset and an external wrench [11]. Therefore our starting assumption is that there are no modeling errors (*i.e.,* a preliminary trunk CoM identification has been carried out previously using [46]).

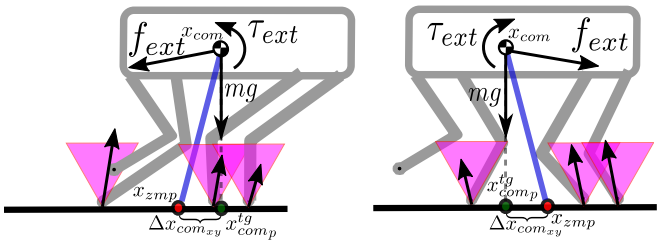

Fig. 17. Diagram for the computation of the ZMP due to external disturbances. Left and right figures show different values of $\Delta x_{com_{xy}}$ for different external disturbances.

where Δx_{com} is the vector going from the ZMP to the CoM[27]. Rewriting (37) in an explicit form[28], we get [49]:

$$\begin{cases} \Delta x_{com_x} = \frac{1}{f_{ext_z} - mg} \left[f_{ext_x} (x_{com_z} - x_{zmp_z}) + \tau_{ext_y} \right] \\ \Delta x_{com_y} = \frac{1}{f_{ext_z} - mg} \left[f_{ext_y} (x_{com_z} - x_{zmp_z}) - \tau_{ext_x} \right] \end{cases} \quad (38)$$

then, the new target computed at the beginning of the base motion will account for this term:

$$x^{tg}_{com_{x,y}} = x^{tg}_{com_{x,y}} + \Delta x_{com_{x,y}} \quad (39)$$

B. Stability issues

Any observer/state feedback arrangement can lead to some stability issues if the gains are not set properly (*e.g.,* by separation principle). We did not carried out any system stability analysis, since a proper evaluation of the stability region (and an improved implementation taking care of this aspects) is an ongoing work and it is out of the scope of this paper. However, we noticed that there are some combinations of gains for which the system becomes unstable.

In the experiments, we decided to be conservative and set lower gains than in simulation. We also did not see a significant improvement in using the implementation (35) instead of (33).

C. Simulations

To evaluate the quality of the *estimation*, we inject in simulation a known disturbance: a pure force to the back of the robot (with application point ${}_b x_p = [-0.6, 0.0, 0.08]\, m$, expressed in the base frame), while the robot is standing still. Specifically, we generate a time-varying perturbation force f_{ext} with random stepwise changes both in *magnitude* (between 40 N and 200 N) and *direction*, with additive white Gaussian noise $n \in \mathcal{N}(0, 20)$ N. The gains used in the observer are the following: $G_{ang} = \text{diag}(100, 100, 100)$, $G_{ang} = \text{diag}(10, 10, 10)$.

The result of the estimation is shown in Fig. 18: the estimator is able to follow promptly the step changes with the gains set, while a small filtering effect of the noise is given by the observer dynamics.

[27]Note that only gravity and the external wrench have an influence on $\Delta x_{com_{x,y}}$, because the resultant of the GRF passes through the ZMP point, by definition.

[28]Note that computing this equations as $\Delta x_{com} = [mg + f_{ext}]_\times \tau_{ext}$ where $[.]_\times$ is the skew symmetric operator associated to the cross product, returns inaccurate results because $[\cdot]_\times$ is rank deficient.

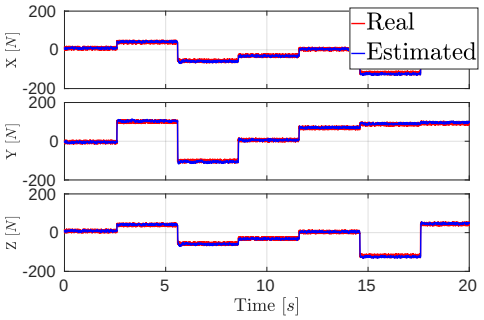

Fig. 18. Simulated estimation of a time-varying external force.

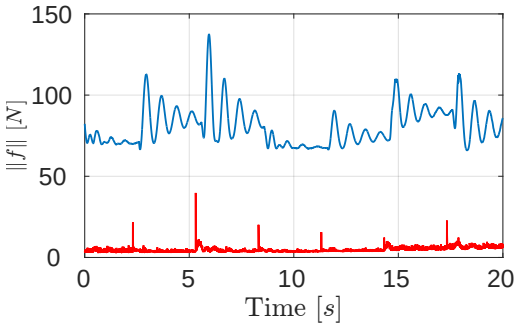

Fig. 19. Norm of tracking error the with (red) and without (blue) compensation of the external wrench.

To evaluate the effectiveness of the *compensation*, we observe that a not properly compensated external force (*e.g.,* by the Trunk Controller) results in GRF which differ from the desired ones (outputs of the optimization), even in absence of modeling errors. This can create slippages and loss of contact, as shown in the accompanying video[29].

Therefore, a good metric to assess the effectiveness of the compensation is the norm of the GRF tracking error $\|f\|$. Figure 19 shows the plots of $\|f\|$ showing that the compensation improves significantly the GRF tracking by almost two orders of magnitude.

The video also shows a simulation of the robot pulling the wheelbarrow, illustrating the GRF (green arrows) and the location of the ZMP (purple sphere). The simulation shows that the robot "leans forward", compensating for the offset in the ZMP created by the pulling force, while the back legs are more loaded (*i.e.,* with larger GRF) with respect to the front ones, due to the weight of the wheelbarrow.

D. Experimental Results

To demonstrate the effectiveness of our approach, we designed several test scenarios in which the robot is walking and compensating a time-varying disturbance force[30]. In all the experiments, we set the gains to $G_{\text{lin}} = \text{diag}(10, 10, 10)$, $G_{\text{ang}} = \text{diag}(1, 1, 1)$.

[29]MBDO simulations: `https://www.youtube.com/watch?v=EUhw-aAFYrw&t=9m02s`

[30]MBDO experiments: `https://www.youtube.com/watch?v=EUhw-aAFYrw&t=9m54s`

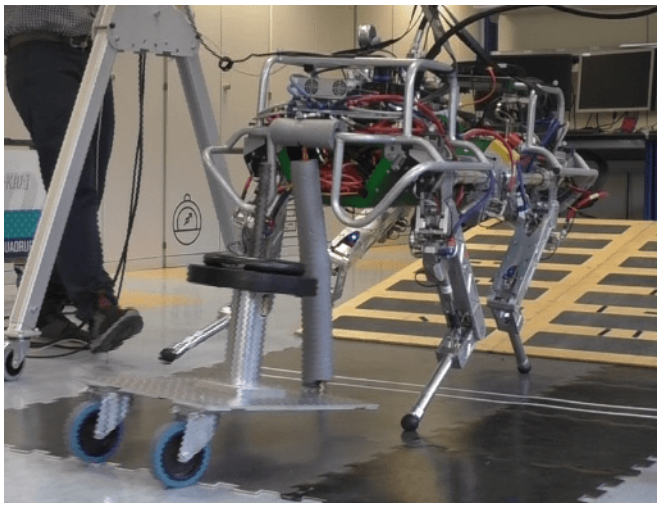

Fig. 20. HyQ quadruped robot pulling a wheelbarrow (of 12 kg) loaded with 15 kg of additional weight. The total vertical force acting on the robot is about 13.5 kg. The cart is attached to the robot through a rope.

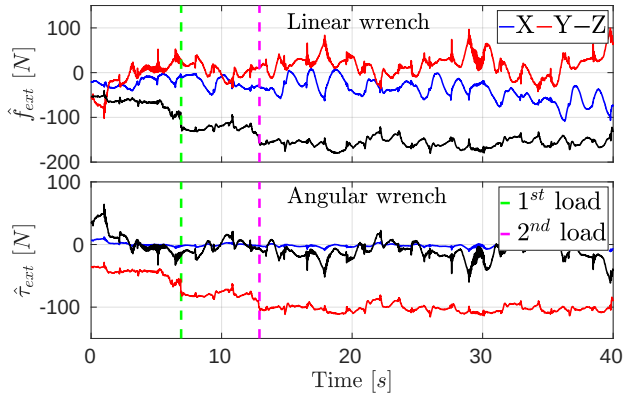

Fig. 21. Wheelbarrow experiments. Estimated wrench in real experiments while the robot carries out a wheelbarrow on rough terrain, (upper plot) linear part (lower plot) the angular one. The two vertical lines represent the moments where the 1-st (10 *kg*) and 2-nd (5 *kg*) load were applied.

Experiment 1 - Pulling a Wheelbarrow: pulling a wheelbarrow on a ramp is an interestings experimental scenario for our MBDO.

This task poses several challenges: 1) the wheelbarrow attached with a rope (intermittent unilateral pull constraint) creates a disturbance force which has a constant vertical component (to counteract wheelbarrow gravity) and a time-varying component (due to horizontal accelerations); 2) The motion of the wheel barrow results in potential unload of the rope that causes discontinuity in the force; 3) a walking gait involves a contact condition that is changing, this mean that the compensation force must be realized by different set legs in different moments, without discontinuitiess. 4) the additional loading of the wheelbarrow is done *impulsively*; 5) walking on a ramp shrinks the support polygon and reduces the stability margin, requiring a high accuracy in the estimation/compensation pipeline.

The accompanying video shows the robot walking on a ramp while pulling a 12 kg wheelbarrow. We performed different trials, with the disturbance wrench compensation enabled, adding 10 kg and 5 kg supplementary weights (on top of the wheelbarrow), in different stages of the walk. Being the total weight of 27 kg equally shared between the robot and the wheels of the cart, we estimate the total load is around 13.5 kg. With the compensation enabled, the robot is able to smoothly climb the ramp while dragging this extra weight. Without the compensation, the robot was struggling to maintain the support polygon even with 5 kg of extra weight only (for a total load of about 8.5 kg actually hanging on the robot). With 15 *kg* it was unable to climb the ramp.

Figure 21 shows the components of the estimated wrench. As expected, the most relevant ones are f_{ext_x}, which is time varying, and f_{ext_z}, which is mostly the constant and vertical component of the disturbance force. Note how the vertical component f_{ext_z} changes when two different loads are added to the wheelbarrow. Since the wheelbarrow is attached to the back of the robot, it also exerts a constant negative moment around the *Y* direction. Since the interaction force with the wheelbarrow is unknown, again the GRF tracking is the only metric available to assess the quality of the compensation.

Figure 22 shows the pitch variation (upper plot) while walking first horizontally and then up the slope. The GRF tracking for the *LH* leg is also shown (lower plot). Thanks to the compensation, the tracking error is always below 40 N for the Z component (about 5 % of the total robot's weight) and 20 N for the X component.

Experiment 2 - leaning against a pulling force: In this experiment a 15*kg* load is attached to the back of the robot with a rope. An intermediate pulley transforms the gravitational load into a horizontal force. This test allows to evaluate the effectiveness of the estimation/compensation pipeline in presence of mainly *horizontal* disturbance forces. In the experiments, the robot is able to pull 15 kg when walking with the compensation enabled. When the estimation is disabled, the controller accumulates errors and suffers from instability. Fig. 24 shows that the only significant component in the estimated wrench is along the *X* direction (horizontal).

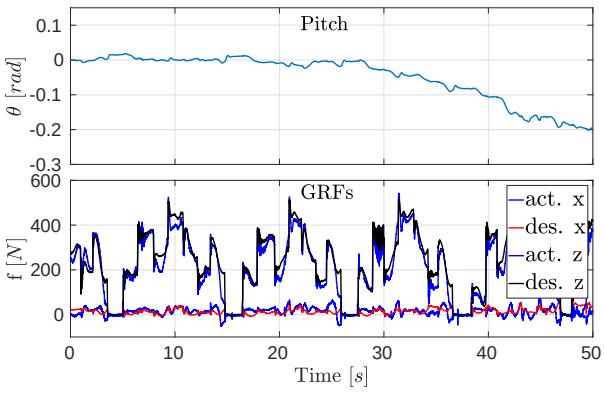

Fig. 22. Wheelbarrow experiments: (upper plot) trunk pitch angle and (lower plot) tracking of the *Z* component of the GRF of the *LH* leg, while the robot carries the wheelbarrow on ramp.

Fig. 23. HyQ quadruped robot dragging a 150N *horizontal* disturbance force (15 kg load) to testing the performances of the MBDO.

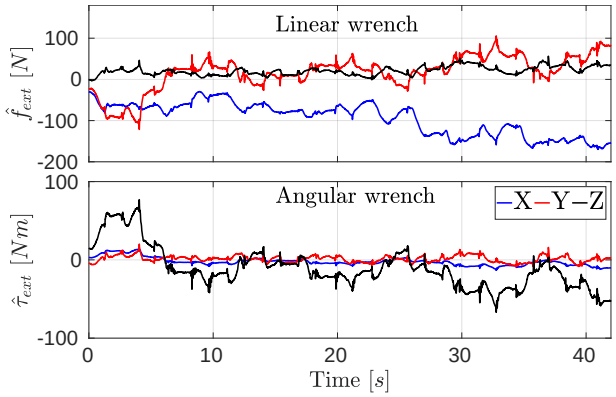

Fig. 24. Curling experiments. Estimated wrench: (upper plot) linear part, the load is mainly along the X component (horizontal) (lower plot) angular part, all the moments are oscillating around zero, showing that the interaction force direction is passing close to the CoM.

The force increases, while the robot is walking, because the load is pulled up gradually (see video), until reaching the steady value of 150 N (correspondent to the 15 kg load) . The RVIZ visualization in the accompanying video shows that the GRF have a constant component along the X direction (*e.g.,* pointing forward) to compensate for the backward pulling force. At the same time, when the compensation is active the ZMP is shifted forward.

IX. Conclusions

In this work we addressed the problem of locomotion on rough terrain. We proposed a 1-step heuristic planning strategy that can work with variable levels of exteroceptive feedbacks. When no perception is available and no map of the environment is given to the robot, the locomotion framework presented in this paper still manages to traverse challenging terrains. This is possible thanks to its capability to *blindly* adapt to the terrain, mainly because of the 1-step re-planning and of the *haptic* touchdown. When a map of the surrounding world is available, the proposed locomotion framework can exploit this information by including the actual terrain height and orientation in the planning decreasing the chances of stumbling and allowing to traverse more difficult situations (*e.g.,* clibing stairs).

Using the proposed approach we were able to achieve several experimental results such as traversing a template rough terrain made of ramps, debris and stairs. We have also shown our quadruped walking while compensating for external *time-varying* disturbance forces caused by a load up to 15 kg using the MBDO that we proposed. All these tasks where performed with the same locomotion strategy that we presented in this article.

Future works involve an extensive analysis of the stability of the MBDO considering its interactions with the trunk controller loop. Based on these results, we intend to implement improved observer that maximizes the stability region. Our future research directions also include extending the navigation capabilities of HyQ to more complex environments made of rolling stones, big size obstacles (of more than $> 12cm$ of diameter). For these scenarios we envision the need to develop a shin collision detection algorithms that may perform kino-dynamic proprioception using a foot-switch sensor. Finally we plan to carry out extensive stair climbing experiments with the proposed approach.

Appendix A

List of the main symbols used throughout the paper:

Symbol	Description
$x_{com} \in \mathbb{R}^3$	coordinates of the CoM of the robot
$x_{fi}, \dot{x}_{fi} \in \mathbb{R}^3$	positions and velocities of the i^{th} foot
$f_i \in \mathbb{R}^3$	contact forces of the i^{th} stance foot
n	number of active joints of the robot
$q_j^d, \dot{q}_j^d \in \mathbb{R}^n$	joint reference positions and velocities
$\tau_{ff}^d \in \mathbb{R}^n$	feedforward torque command
$\tau_{pd}^d \in \mathbb{R}^n$	impedance torque command
$\tau^d \in \mathbb{R}^n$	total reference torque command
ϕ	roll angle of the robot's trunk
θ	pitch angle of the robot's trunk
ψ	yaw angle of the robot's trunk
$\Phi = [\phi, \theta, \psi]$	actual orient. of the robot's trunk
$\Phi^d = [\phi^d, \theta^d, \psi^d]$	desired orient. of the robot's trunk
$\Phi^d(0) = \Phi$	des. orient. at the start of the move base
Φ^{tg}	target orient. at the end of the move base
x_{com}^{tg}	target CoM position of the trunk
$x_{com_p}^{tg}$	projection of x_{com}^{tg} on the terrain plane
h_r	robot's height
f_{ext}, τ_{ext}	external disturbance force and torque

Appendix B

The following section, shows how to compute the weighted average of N orientation vectors.

Because finite rotations do not add like vectors, it is not possible to apply ordinary laws of vector arithmetic to them. Specifically, the task of averaging orientations (described in the algorithm 1 written in pseudo-code), it is equivalent to average points on a sphere, where the line is replaced with a spherical geodesic (arc of a circle)[31].

The iterative procedure illustrated above, is generic and can be used to obtain the average of N directional vectors

[31]The geodesic distance is the length of the shortest curve lying on the sphere connecting the two points.

Algorithm 1 compute geodesic average

$w_1 \leftarrow 1$
$\bar{n} \leftarrow n_1$
for $k = 2$ to N **do**
    $\theta_k = acos(\text{dot}(\bar{n}, n_k))$
    $\bar{\theta}_k = \theta_k \frac{\sum_{p=1}^{k-1} w_p}{\sum_{p=1}^{k} w_p}$
    $\bar{n} \leftarrow$ rotate n_k towards $\bar{n}$ by angle $\bar{\theta}_k$
end for

n_k, even though at each iteration we are only able to directly compute the average of two. After the initialization, at each loop the actual average $\bar{n}$ is updated with the next normal n_k. To do so, we first compute the angle θ_k between n_k and the actual average $\bar{n}$. Then we scale this angle, according to the (accumulated) weight of $\bar{n}$. Finally, n_k is rotated by the scaled angle $\hat{\theta}$ toward $\bar{n}$ and the result is assigned back to $\bar{n}$.

ACKNOWLEDGEMENT

This work was supported by 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*.

REFERENCES

[1] T. Bretl and S. Lall, "Testing static equilibrium for legged robots," *IEEE Transactions on Robotics*, vol. 24, pp. 794–807, Aug 2008.

[2] D. Pardo, L. Möller, M. Neunert, A. W. Winkler, and J. Buchli, "Evaluating Direct Transcription and Nonlinear Optimization Methods for Robot Motion Planning," *IEEE Robotics and Automation Letters*, vol. 1, no. 2, pp. 946–953, 2016.

[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] A. W. Winkler, C. D. Bellicoso, M. Hutter, and J. Buchli, "Gait and Trajectory Optimization for Legged Systems through Phase-based End-Effector Parameterization," *IEEE Robotics and Automation Letters*, pp. 1–1, 2018.

[5] B. Aceituno-Cabezas, C. Mastalli, D. Hongkai, 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*, pp. 1–8, 2018.

[6] C. Mastalli, A. W. 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.

[7] H. Dai and R. Tedrake, "Planning robust walking motion on uneven terrain via convex optimization," *2016 IEEE-RAS International Conference on Humanoid Robots (Humanoids 2016)*, 2016.

[8] B. Ponton, A. Herzog, A. Del Prete, S. Schaal, and L. Righetti, "On Time Optimisation of Centroidal Momentum Dynamics," *https://arxiv.org/abs/1709.09265*, 2017.

[9] B. J. Stephens, "State estimation for force-controlled humanoid balance using simple models in the presence of modeling error," in *2011 IEEE International Conference on Robotics and Automation*, pp. 3994–3999, May 2011.

[10] J. Englsberger, G. Mesesan, and C. Ott, "Smooth trajectory generation and push-recovery based on divergent component of motion," in *IEEE International Conference on Intelligent Robots and Systems (IROS)*, pp. 4560–4567, September 2017.

[11] N. Rotella, "Estimation-based control for humanoid robots," in *PhD Thesis*, 2018.

[12] G. Bledt, P. M. Wensing, and S. Kim, "Policy-regularized model predictive control to stabilize diverse quadrupedal gaits for the MIT cheetah," *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 4102–4109, 2017.

[13] P.-B. Wieber, "Trajectory Free Linear Model Predictive Control for Stable Walking in the Presence of Strong Perturbations Trajectory Free Linear Model Predictive Control for Stable Walking in the Presence of Strong Perturbations," *IEEE-RAS International Conference on Humanoid Robots*, 2006.

[14] H.-W. Park, P. Wensing, and S. Kim, "Online Planning for Autonomous Running Jumps Over Obstacles in High-Speed Quadrupeds," *Robotics: Science and Systems XI*, 2015.

[15] C. D. Bellicoso, C. Gehring, J. Hwangbo, P. Fankhauser, and M. Hutter, "Perception-less terrain adaptation through whole body control and hierarchical optimization," in *2016 IEEE-RAS 16th International Conference on Humanoid Robots (Humanoids)*, pp. 558–564, Nov 2016.

[16] C. D. Bellicoso, F. Jenelten, P. Fankhauser, C. Gehring, J. Hwangbo, and M. Hutter, "Dynamic Locomotion and Whole-Body Control for Quadrupedal Robots," *IEEE/RSJ Intenational Conference on Intelligent Robots and Systems (IROS)*, pp. 3359–3365, 2017.

[17] M. Fallon, P. Marion, R. Deits, T. Whelan, M. Antone, J. McDonald, and R. Tedrake, "Continuous humanoid locomotion over uneven terrain using stereo fusion," in *2015 IEEE-RAS 15th International Conference on Humanoid Robots (Humanoids)*, pp. 881–888, nov 2015.

[18] 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*, vol. 2, pp. 1023–1030, apr 2017.

[19] M. Bloesch, M. Hutter, M. H. Hoepflinger, C. Gehring, C. David Remy, and R. Siegwart, "State estimation for legged robots: Consistent fusion of leg kinematics and IMU," vol. 8, pp. 17–24, MIT Press Journals, 2013.

[20] X. Xinjilefu, S. Feng, W. Huang, and C. G. Atkeson, "Decoupled state estimation for humanoids using full-body dynamics," in *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 195–201, May 2014.

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

[22] 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," *Proceedings - IEEE International Conference on Robotics and Automation*, pp. 2554–2561, 2013.

[23] M. Focchi, V. Barasuol, M. Frigerio, D. G. Caldwell, and C. Semini, "Slip Detection and Recovery for Quadruped Robots," vol. 3, pp. 185–199, Cham: Springer Proceedings in Advanced Robotics, 2018.

[24] V. Barasuol, M. Camurri, S. Bazeille, D. Caldwell, and C. Semini, "Reactive trotting with foot placement corrections through visual pattern classification," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 10 2015.

[25] D. Belter and P. Skrzypczyński, "Rough terrain mapping and classification for foothold selection in a walking robot," *Journal of Field Robotics*, vol. 28, no. 4, pp. 497–528, 2011.

[26] C. Mastalli, I. Havoutis, M. Focchi, D. G. Caldwell, and C. Semini, "Motion planning for quadrupedal locomotion: coupled planning, terrain mapping and whole-body control," *HAL id: hal-01673438v2*, 2018.

[27] 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.

[28] M. Focchi, R. Featherstone, R. Orsolino, D. G. Caldwell, and C. Semini, "Viscosity-based height reflex for workspace augmentation for quadrupedal locomotion on rough terrain," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2017.

[29] M. Focchi, V. Barasuol, I. Havoutis, J. Buchli, C. Semini, and D. G. Caldwell, "Local reflex generation for obstacle negotiation in quadrupedal locomotion," in *Int. Conf. on Climbing and Walking Robots (CLAWAR)*, 2013.

[30] C. Semini, N. Tsagarakis, E. Guglielmino, M. Focchi, F. Cannella, and D. Caldwell, "Design of HyQ – a Hydraulically and Electrically Actuated Quadruped Robot," *J. of Systems and Control Eng.*, 2011.

[31] A. S. Tar, G. G. Cserey, and J. Veres, May 2014. Patent N. WO 2013072712 A1.

[32] Y. Gao, February 2018. Patent N. IT 102018000002407.

[33] 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)*, May 2015.

[34] T. Boaventura, J. Buchli, C. Semini, and D. G. Caldwell, "Model-based hydraulic impedance control for dynamic robots," *IEEE Transactions on Robotics*, vol. 31, pp. 1324–1336, Dec 2015.

[35] R. Orsolino, M. Focchi, C. Mastalli, H. Dai, D. G. Caldwell, and C. Semini, "Application of Wrench based Feasibility Analysis to the Online Trajectory Optimization of Legged Robots," *IEEE Robotics and Automation Letters (RA-L)*, 2018.

[36] H. Audren and A. Kheddar, "3D robust stability polyhedron in multi-contact," *Submitted to IEEE Transactions On Robotics (TRO)*, 2017.

[37] R. M. Alexander, *Principles of animal locomotion*. Princeton University Press, 2003.

[38] 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.

[39] M. Camurri, *Multisensory State Estimation and Mapping on Dynamic Quadruperd Robots*. PhD thesis, Istituto Italiano di Tecnologia (IIT) and Universiy of Genoa, 2017. `http://iit-dlslab.github.io/papers/camurri17phd.pdf`.

[40] P. Fankhauser and M. Hutter, "A Universal Grid Map Library: Implementation and Use Case for Rough Terrain Navigation," in *Robot Operating System (ROS) – The Complete Reference (Volume 1)* (A. Koubaa, ed.), ch. 5, Springer, 2016.

[41] C. Semini, V. Barasuol, J. Goldsmith, M. Frigerio, M. Focchi, Y. Gao, and D. G. Caldwell, "Design of the hydraulically-actuated, torque-controlled quadruped robot hyq2max," *IEEE/ASME Transactions on Mechatronics*, vol. PP, no. 99, pp. 1–1, 2016.

[42] C. Gramkow, "On averaging rotations," *Journal of Mathematical Imaging and Vision*, vol. 15, no. 1-2, pp. 7–16, 2001.

[43] C. Gehring, S. Coros, M. Hutler, C. Dario Bellicoso, H. Heijnen, R. Diethelm, M. Bloesch, P. Fankhauser, J. Hwangbo, M. Hoepflinger, and R. Siegwart, "Practice Makes Perfect: An Optimization-Based Approach to Controlling Agile Motions for a Quadruped Robot," *IEEE Robotics and Automation Magazine*, vol. 23, no. 1, pp. 34–43, 2016.

[44] S. Oßwald, J. S. Gutmann, A. Hornung, and M. Bennewitz, "From 3d point clouds to climbing stairs: A comparison of plane segmentation approaches for humanoids," in *2011 11th IEEE-RAS International Conference on Humanoid Robots*, pp. 93–98, Oct 2011.

[45] D. Pongas, M. Mistry, and S. Schaal, "A Robust Quadruped Walking Gait for Traversing Rough Terrain," in *Proceedings 2007 IEEE International Conference on Robotics and Automation*, pp. 1474–1479, April 2007.

[46] G. Tournois, M. Focchi, A. Del Prete, R. Orsolino, D. G. Caldwell, and C. Semini, "Online payload identification for quadruped robots," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2017.

[47] R. Featherstone, "Rigid Body Dynamics Algorithms," *Springer US, Boston, MA*, 2008.

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

[49] M. B. Popovic, A. Goswami, and H. Herr, "Ground reference points in legged locomotion: Definitions, biological trajectories and control implications," *The International Journal of Robotics Research*, vol. 24, no. 12, pp. 1013–1032, 2005.