ISTITUTO ITALIANO
DI TECNOLOGIA
DYNAMIC LEGGED SYSTEMS

MPC-based Controller with Terrain Insight for Dynamic Legged Locomotion

Octavio Villarreal, Victor Barasuol, Patrick M. Wensing,
Darwin G. Caldwell, and Claudio Semini

Submitted: 16/09/2019. Accepted: 22/01/2020.

To be published in:

International Conference on Robotics and Automation (ICRA) 2020.

To cite this paper:

O. Villarreal, V. Barasuol, P. Wensing, Darwin G. Caldwell, and C. Semini, "MPC-based Controller with Terrain Insight for Dynamic Legged Locomotion," *IEEE International Conference on Robotics and Automation (ICRA)*, May 2020.

Video:
`https://youtu.be/CqlLRdohFwM`

For this and other publications from the Dynamic Legged Systems (DLS) lab:
`https://dls.iit.it/dls-publications`

©2020 IEEE. Personal use of this material is permitted. Permission from IEEE must be obtained for all other uses, in any current or future media, including reprinting/republishing this material for advertising or promotional purposes, creating new collective works, for resale or redistribution to servers or lists, or reuse of any copyrighted component of this work in other works.

MPC-based Controller with Terrain Insight for Dynamic Legged Locomotion

Octavio Villarreal[1], Victor Barasuol[1], Patrick M. Wensing[2], Darwin G. Caldwell[3], and Claudio Semini[1]

***Abstract*— We present a novel control strategy for dynamic legged locomotion in complex scenarios that considers information about the morphology of the terrain in contexts when only on-board mapping and computation are available. The strategy is built on top of two main elements: first a contact sequence task that provides safe foothold locations based on a convolutional neural network to perform fast and continuous evaluation of the terrain in search of safe foothold locations; then a model predictive controller that considers the foothold locations given by the contact sequence task to optimize target ground reaction forces. We assess the performance of our strategy through simulations of the hydraulically actuated quadruped robot *HyQReal* traversing rough terrain under realistic on-board sensing and computing conditions.**

I. Introduction

Considering terrain morphology allows legged robots to traverse more complex scenarios (e.g., [1], [2], [3]). Nevertheless, building a model of the terrain is often computationally costly, mainly because of the dense nature of visual data. On top of the mapping problem, feasible contact sequences are needed to traverse the terrain safely. Computing these contact sequences can also be costly [4], [5]. In general, strategies that consider visual information of the terrain are mostly focused on trajectory optimization [6], [7], [8], [9]. In most approaches, contact sequences and the Center of Mass (COM) trajectory are computed prior to the motion, or are limited to (quasi-) statically stable gaits to not compromise stability due to time constraints [2], [10], [11].

In this work, we combine the low computational time from our previous Vision-based Foothold Adaptation (VFA) strategy from [12] with a Model Predictive Control (MPC)-based trunk controller. These two approaches are mutually beneficial to each other. On one hand, we exploit the computational gain that we obtain from the Convolutional Neural Network (CNN) in the VFA strategy, to generate safe contact sequences to be used in the MPC-based COM tracking controller. On the other hand, the VFA benefits from the MPC-based controller with respect to the foothold prediction. A *foothold prediction* is a future landing position based on the nominal trajectory of the legs and the trunk velocity. We stress that the foothold predictions are different from the state predictions computed using the MPC.

We start from the premise that optimizing ground reaction forces (GRFs) accounting for future states using MPC will lead to better foothold predictions, since they depend both on foot trajectories and robot states. If the robot states have large acceleration peaks, the foothold prediction is affected negatively. An improved selection of the desired GRFs would reduce acceleration peaks, improving foothold predictions. This allows the robot to handle more difficult scenarios, such as changes in elevation and orientation, in a safer and more reliable way, as demonstrated in simulations.

We perform simulations using the quadruped robot *HyQReal* [13]. Its four legs weigh in total 48 kg (between 37% and 45% of the total weight of the robot, with and without on-board hydraulic/electric power units, respectively). This means that when fast motions are required, swing legs play a significant role in the robot dynamics. In this paper, we directly compensate for these effects by computing the wrench on the body due to the desired joint accelerations of the legs, improving state tracking and foothold prediction.

The result is a stable locomotion strategy, which is robust to a wide range of disturbances and is able to act preemptively to obstacles based on visual information. We summarize the contributions of this paper as follows:

- We devised a locomotion strategy that evaluates the terrain, generates safe contact sequences and allows for dynamic locomotion in difficult scenarios. The new strategy displayed an improvement in foothold prediction with respect to [12], reducing the prediction error by a percentage between 7% to 36%.
- We combine a CNN-based foothold adaptation strategy and an MPC-based trunk controller and show how they mutually benefit from each other. To the best of our knowledge, this is the first time that an MPC-based locomotion controller uses the terrain geometry in combination with a machine learning strategy.
- We improve the performance of the MPC, by compensating for the wrench exerted by the legs during swing phase due to the large weight of the legs with respect to the total weight of the robot. This compensation renders the model used for prediction in the MPC more representative, since the leg inertia is handled separately, further reducing the error in foothold prediction.

This paper is structured as follows: Section II summarizes the previous work relevant to this research; Section III details the methodology used to derive our locomotion strategy; Section IV summarizes our results. Conclusions and future work are presented in Section V.

[1]Dynamic Legged Systems lab, Istituto Italiano di Tecnologia, Via Morego 30, 16163 Genoa, Italy. firstname.lastname@iit.it

[2] Department of Aerospace and Mechanical Engineering, University of Notre Dame, IN 46556 USA. pwensing@nd.edu

[3] Department of Advanced Robotics, Istituto Italiano di Tecnologia, Via Morego 30, 16163 Genoa, Italy. darwin.caldwell@iit.it

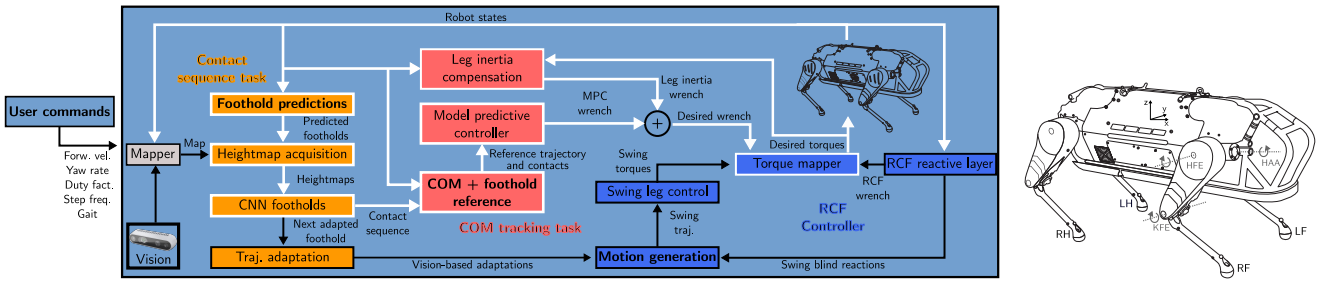

Fig. 1: Left: schematic drawing describing the control strategy. The user commands are used by the *foothold predictions*, *COM + reference*, and *motion generation* blocks (denoted with bold text). The contact sequence task is denoted by the orange blocks. The COM tracking task is denoted by the red blocks. The RCF [14] (blue blocks) serves as an interface for the VFA, the MPC + leg inertia compensation controller and a reactive layer for "blind" locomotion. The torque mapper block distributes the total wrench among the GRFs (solving a QP similarly to [15]) and maps to joint torques using $\boldsymbol{\tau} = \mathbf{J}^{\mathsf{T}}\mathbf{F}$. The white arrows connect the blocks where the MPC and the CNN interact. Right: joint and leg definitions of HyQReal.

II. Related Work

The spectrum of strategies when using MPC varies mostly depending on the trade-off between model accuracy and computational cost. The work of Di Carlo et al. [16] considers a simplified version of the centroidal dynamics model, neglecting leg inertia, and ignoring the effects of non-zero roll/pitch on the dynamics of the body. The optimization problem is still convex and it is solved in real-time as a Quadratic Program (QP). The strategy keeps the robot in balance during a range of highly dynamic motions (e.g., trot, bound, and gallop) on the Cheetah 3 robot [17]. Herein, a feedforward torque compensates leg inertia to improve swing leg trajectory tracking. Our work differs from this implementation, since we compute the wrench exerted on the trunk by the swing legs and compensate for it using the GRFs of the stance legs to improve COM tracking.

Some other approaches tackle the trade-off between computational cost and model accuracy by relying on reducing the computational cost of the optimization problem solver. A remarkable example is the one devised by Neunert et al. in [18]. Their approach performs Nonlinear Model Predictive Control (NMPC) and relies on a custom solver based on the Iterative Linear Quadratic Regulator (iLQR) algorithm and exploits automatic differentiation [19].

Using MPC has provided a systematic and robust way to address the quadruped locomotion problem. Nevertheless, in general it does not take into account future terrain within the prediction. Instead, it reacts to the terrain and relies on the fact that the continuous update of the state for the initialization of the optimization provides enough robustness. It is still challenging to leverage terrain information to improve the performance of MPC strategies.

There are some trajectory optimization methods that consider terrain information for quadrupeds. The method devised by Winkler et al. in [7] is able to optimize gait, COM trajectory and contacts on non-flat terrain based on a simplified centroidal dynamics model using an off-the-shelf Nonlinear Program (NLP) solver. In a similar fashion, Aceituno et al. showcase a motion planning algorithm that computes gait pattern, contact sequences and COM trajectory as an outcome of a Mixed-Integer Convex Program (MICP) on several non-planar convex surfaces in [20]. However, in their work, either the trajectory is computed only once before execution, or the terrain is assumed to be known and there are no experiments with vision sensors during the motion.

The early works of Kalakrishnan et al. [1], [10] on LittleDog pioneered methods to include vision sensors for locomotion by relying on external motion capture systems. Belter et al. [3] and Fankhauser et al. [2] devised control architectures that allowed their legged platforms to traverse complex scenarios only using on-board sensing enhanced with vision, which were mostly demonstrated for statically stable motions. In our previous work [12], we presented a strategy that was able to adapt footholds based on a CNN. The approach generated swing leg trajectory adaptations in less than 0.1 ms. This allowed us to execute dynamic locomotion in complex scenarios.

III. Locomotion Strategy

Our goal is to produce robust and stable locomotion in complex scenarios using terrain information provided by on-board vision sensors. We combine MPC with a CNN-based foothold adaptation strategy [12]. The combination of these two strategies benefits each other.

State predictions in the MPC are computed using the centroidal dynamics model and safe contact sequences are based on the VFA. Future footholds can be continuously computed by the VFA approximately every 0.5 ms, enabling the MPC to reason about the effects of future contacts, without having to consider them as optimization variables. We then build upon these contacts to provide the reference pose for the robot along the prediction horizon.

The block diagram shown in Fig. 1 describes our locomotion strategy. It entails three main elements: the contact sequence task, the COM tracking task and the Reactive Controller Framework (RCF) [14]. The contact sequence task provides the future contact locations according to the robot current states and the gait timing parameters. The COM tracking task is in charge of both generating and following a COM trajectory according to the contact sequence task, the current robot states, and the gait parameters. We use the RCF [14] as controller interface. This modular framework allows us to combine the *RCF reactive layer* block in Fig. 1 with our vision-based strategy. This layer is comprised by several

modules that allow the robot to perform robust locomotion in rough terrain only using proprioception. The RCF combines these reactive modules with the MPC and the VFA.

The user commands are: forward velocity $\mathbf{V}_f \in \mathbb{R}^2$ (x and y velocities), yaw rate $\dot{\psi}_{ref} \in \mathbb{R}$, duty factor D_f, step frequency f_s and gait $\mathcal{G}$. $\mathbf{V}_f$ and $\dot{\psi}_{ref}$ are provided via joystick commands. The rest of the parameters are preset by the user according to the desired gait and range of speeds.

Below we explain the two main elements of our strategy: the contact sequence and the COM tracking tasks.

A. Contact Sequence Task

We extend the use of the VFA [12] to provide the subsequent eight reference footholds (two strides). These footholds are used to generate the COM reference trajectory and provide the contacts for the model described in Section III-B.

a) Vision-based Foothold Adaptation: The purpose of the VFA is to continuously compute adjustments for the trajectory of the feet in order to avoid collisions and unsafe or unreachable landing positions. For a more detailed description on this method we refer the reader to [12].

For a leg in swing phase, we initially compute a prediction of its landing position based on the current velocity of the trunk (taken from the state estimator) and the trajectory of the foot (in our case a half-ellipse) using the approximation:

$$\hat{\mathbf{p}}_i = \bar{\mathbf{p}}_i + \frac{1}{2}\boldsymbol{\ell}_s + \Delta t_i \dot{\mathbf{r}} \tag{1}$$

where $\hat{\mathbf{p}}_i \in \mathbb{R}^3$ is the *predicted foothold* of i-th leg, for $i = 1,...,l$, with l being the total number of legs (see Fig. 1), $\bar{\mathbf{p}}_i \in \mathbb{R}^3$ is the center of the ellipse, Δt_i is the time remaining to the next stance change, $\boldsymbol{\ell}_s \in \mathbb{R}^3$ is the step length vector, and $\dot{\mathbf{r}} \in \mathbb{R}^3$ corresponds to the velocity of the base. All vector variables are given in world coordinates. In the case of the next touchdown of a swing leg, $\Delta t_i = \frac{1-D_f}{f_s} - t_{sw,i}$, where D_f is the duty factor, f_s is the step frequency and $t_{sw,i}$ is the elapsed swing time since the latest lift-off. The first two terms in (1) are related to the leg trajectory, while the third term is related to the displacement of the base.

After computing the prediction of the next foothold, a 2D representation of the terrain around that foothold is acquired, namely a *heightmap*. We pre-train a CNN to learn the *optimal* footholds from heightmaps [12] considering collisions, terrain roughness, and process uncertainty. The architecture of the CNN is designed as a trade-off between prediction accuracy and speed. The CNN takes on average 0.1 ms to evaluate a heightmap and output a safe foothold. This fast computation time allows us to continuously adapt the trajectory of the swing leg to reach the adapted foothold.

b) Reference Contact Sequence: We use the computational gain obtained by the VFA to evaluate further ahead in the terrain. Knowing that the gait is periodic and defined by the step frequency f_s and the duty factor D_f, we can estimate the timings for the non-immediate foot contacts. Using these timings, one can compute the predicted foothold locations for each of the legs at every stance change (lift-off or touchdown) replacing them for Δt in (1). We then use our CNN-based foothold adaptation to adjust the predicted foothold location. This calculation is done for the next two gait cycles (eight contacts in total and 16 stance changes). An example of a safe foothold sequence can be seen on the right side of the series of snapshots of Fig. 2. Namely, $\mathbf{p}_i[\kappa]$ is the contact location of leg i at stance change κ, for $\kappa = 0,...,16$. In (1), $\dot{\mathbf{r}}$ is assumed constant in between stance changes.

The CNN continuously provides safe contact sequences at task frequency (250 Hz). These sequences are used both as future foot positions and to inform the MPC controller to improve the COM regulation, as explained in Section III-B. This interaction is shown in Fig. 1. One key feature of the approach is that safe footholds are computed without including them as optimization variables in the MPC controller, which significantly decreases the complexity of the problem.

B. COM Tracking Task

a) COM Reference Generation: To provide the reference trajectory for the COM along the prediction horizon, we compute its location at every stance change based on the desired gait timings using f_s and D_f. For two gait cycles, there are a total of 16 stance changes, so we compute a total of 16 COM positions. Similarly to the third term of (1), we compute the reference yaw using the desired yaw rate as

$$\psi_{ref}[\kappa] = \psi + \Delta t[\kappa]\dot{\psi}_{ref} \tag{2}$$

where $\psi_{ref}[\kappa] \in \mathbb{R}$ is the yaw reference at stance change κ, for $\kappa = 0,...,16$, $\psi \in \mathbb{R}$ is the current yaw, and $\Delta t[\kappa]$ is the time for a stance change to happen from $\kappa = 0$. Using the reference for the yaw angle, we compute the reference position of the COM with respect to the world

$$\mathbf{r}_{ref}[\kappa] = \mathbf{r} + \Delta t[\kappa]\mathbf{R}_z(\Delta\psi[\kappa])\dot{\mathbf{r}}_{ref} \tag{3}$$

where $\mathbf{r}_{ref}[\kappa] \in \mathbb{R}^3$ is the reference position for the COM at stance change κ, $\mathbf{R}_z(\Delta\psi) \in \mathbb{R}^{3x3}$ is the rotation matrix around the z axis about $\Delta\psi[\kappa]$ (with $\Delta\psi[\kappa] = \psi_{ref}[\kappa] - \psi$) and $\dot{\mathbf{r}}_{ref}$ is the reference velocity obtained from $\mathbf{V}_f$ and $\dot{\psi}_{ref}$. This provides the reference for the next x and y positions of the COM with respect to the world.

The reference for the body roll ϕ_{ref} and pitch θ_{ref} relies on the contact configuration at each stance change. We estimate the orientation of the terrain and define that orientation as reference for the body. We also use the contacts to define a height z reference position for the body (namely, $r_{ref,z}[\kappa]$), setting it to remain at a constant distance from the center position of the approximated plane in the direction of the z world axis. To obtain $\dot{r}_{ref,z}[\kappa]$, $\dot{\phi}_{ref}[\kappa]$ and $\dot{\theta}_{ref}[\kappa]$ we derive numerically between samples of $r_{ref,z}[\kappa]$, $\phi_{ref}[\kappa]$ and $\theta_{ref}[\kappa]$, respectively. Finally, we evenly sample the 16 reference points given by the stance changes, filling the gaps in between samples using a zero-order hold (ZOH). We define a reference vector at evenly sampled time k as

$$\mathbf{x}_{ref}[k] = \begin{bmatrix}\Theta_{ref}^\top[k] & \mathbf{r}_{ref}^\top[k] & \dot{\Theta}_{ref}^\top[k] & \dot{\mathbf{r}}_{ref}^\top[k]\end{bmatrix}^\top \tag{4}$$

with $\Theta_{ref}[k] = \begin{bmatrix}\theta_{ref}[k] & \phi_{ref}[k] & \psi_{ref}[k]\end{bmatrix}^\top$ and $\mathbf{r}_{ref}[k] = \begin{bmatrix}r_{ref,x}[k] & r_{ref,y}[k] & r_{ref,z}[k]\end{bmatrix}^\top$. Figure 2 shows multiple COM

references.

b) Dynamic Model: Our MPC trunk balance controller is inspired by the work of Di Carlo et al. [16]. We model the robot as a rigid body subject to contact patches at each stance foot and we neglect the effects of precession and nutation as in [15]. However, there are two key differences in our approach: firstly, we do not define the reference roll and pitch angles to be zero. Additionally, although we do not explicitly consider the leg inertia in our model for control, we compensate for it by computing the wrench exerted by the legs using the actuated part of the joint-space inertia matrix and the desired accelerations of the joints. We explain how this is done by the end of this section.

The dynamics of the rigid body and its rotational kinematics are given by

$$\ddot{\mathbf{r}} = \frac{\sum_{i=1}^{l} \mathbf{F}_i}{m} + \mathbf{g} \tag{5}$$

$$\mathbf{I}\dot{\omega} = \sum_{i=1}^{l} \mathbf{p}_i \times \mathbf{F}_i \tag{6}$$

$$\dot{\mathbf{R}} = [\omega]_\times \mathbf{R} \tag{7}$$

where $\mathbf{r} \in \mathbb{R}^3$ is the position of the COM, $\mathbf{F}_i \in \mathbb{R}^3$ is the GRF at foot i, $m \in \mathbb{R}$ is the robot mass, $\mathbf{g} \in \mathbb{R}^3$ is the gravitational acceleration, $\mathbf{I} \in \mathbb{R}^{3\times3}$ is the inertia tensor of the robot, $\mathbf{p}_i \in \mathbb{R}^3$ is the i-th foot contact position, $\mathbf{R} \in \mathbb{R}^{3\times3}$ is the rotation matrix from body to world coordinates according to roll ϕ, pitch θ and yaw ψ angles and $\omega \in \mathbb{R}^3$ is the robot's angular velocity. The operator $[\mathbf{x}]_\times$ is the skew-symmetric matrix such that $[\mathbf{x}]_\times \mathbf{y} = \mathbf{x} \times \mathbf{y}$. In (6) we are neglecting precession and nutation effects, namely $\omega \times \mathbf{I}\omega \approx 0$. We rewrite equations (5), (6) and (7) in state-space representation. Initially, from (7) we can obtain the angular velocity in terms of the body's Euler angles from

$$[\omega]_\times = \dot{\mathbf{R}}\mathbf{R}^\top \tag{8}$$

which can be rewritten as

$$\omega = \mathbf{T}(\Theta)\dot{\Theta} \tag{9}$$

where $\Theta = [\phi \ \ \theta \ \ \psi]^\top$ and $\mathbf{T}(\Theta)$ is the matrix that maps from Euler angle rates to angular velocities. The only condition on $\mathbf{T}(\Theta)$ to be invertible is $\theta \neq \pi/2$, which in practice does not happen (it implies that the robot is pointed vertically). Thus, the angular rate can be obtained as

$$\dot{\Theta} = \mathbf{T}^{-1}(\Theta)\omega \tag{10}$$

We define state vector[1] $\mathbf{x} = [\Theta^\top \ \mathbf{r}^\top \ \omega^\top \ \dot{\mathbf{r}}^\top \ \mathbf{g}^\top]^\top$ and rearrange (5), (6) and (10) to write them in state-space as

$$\dot{\mathbf{x}}(t) = \mathbf{A}(\Theta)\mathbf{x}(t) + \mathbf{B}(\Theta, \mathbf{p}_{LF}, ..., \mathbf{p}_{RH})\mathbf{u}(t) \tag{11}$$

where $\mathbf{u}$ is the vector of GRFs. Note that no assumptions are made about the orientation of the robot[2] (except for $\theta \neq \pi/2$) and we explicitly denote the dependence of $\mathbf{T}$ and $\mathbf{I}$ on Θ.

In a similar fashion to [16], we approximate the dynamics in (11) to a discrete-time linear system. Namely, for each reference vector $\mathbf{x}_{ref}[k]$ (for $k = 1, ..., n$, where n is the prediction horizon length), we compute the approximate linear, discrete system matrices $\mathbf{A}_d[k]$ and $\mathbf{B}_d[k]$.

We first substitute the feet locations $\mathbf{p}_i$ obtained from the contact sequence task into $\mathbf{B}(\Theta, \mathbf{p}_{LF}, ..., \mathbf{p}_{LH})$ for every contact configuration at time instant k. However, $\mathbf{A}(\Theta)$ and $\mathbf{B}(\Theta, \mathbf{p}_{LF}, ..., \mathbf{p}_{LH})$ are still dependent on the body orientation (in a nonlinear fashion). To obtain the linear, discrete time versions of these matrices, we follow a similar argument to [16]. Assuming that the MPC-based controller will follow sufficiently close the reference trajectory given by $\mathbf{x}_{ref}[k]$, we substitute the values of $\Theta_{ref}[k]$ into system matrices $\mathbf{A}(\Theta)$ and $\mathbf{B}(\Theta, \mathbf{p}_{LF}, ..., \mathbf{p}_{LH})$. We also consider the values of θ_{ref} and ϕ_{ref} computed by the COM reference trajectory. We then discretize the system matrices using a ZOH. The discrete-time linear system dynamics can be described as

$$\mathbf{x}[k+1] = \mathbf{A}_d[k]\mathbf{x}[k] + \mathbf{B}_d[k]\mathbf{u}[k] \tag{12}$$

c) Model Predictive Control: We can obtain a discrete time evolution of the system by successive substitution of states $\mathbf{x}[k]$ into (12) to obtain the state evolution from $k = 0$ to $k = n$. Then, we can describe the dynamics as

$$\mathbf{X} = \bar{\mathbf{A}}\mathbf{x}_0 + \bar{\mathbf{B}}\bar{\mathbf{u}} \tag{13}$$

where $\mathbf{X} \in \mathbb{R}^{15n}$ is the stacked vector of states along the prediction horizon $\mathbf{X} = \left[\mathbf{x}^\top[1], \ ..., \ \mathbf{x}^\top[n]\right]^\top$, $\bar{\mathbf{A}} \in \mathbb{R}^{15n\times15n}$ and $\bar{\mathbf{B}} \in \mathbb{R}^{15n\times12n}$ are the matrices built by successive substitution, $\mathbf{x}_0 \in \mathbb{R}^{15}$ is the actual robot state vector and $\bar{\mathbf{u}} \in \mathbb{R}^{12n}$ is the stacked vector of ground reaction forces $\bar{\mathbf{u}} = \left[\mathbf{u}^\top[0], \ ..., \ \mathbf{u}^\top[n-1]\right]^\top$. We formulate the optimization problem to minimize the weighted least-squares error between the states and the reference along the prediction horizon. We enforce the gait pattern $\mathcal{G}$ and friction consistency by setting appropriate constraints. We solve

$$\begin{aligned} \min_{\bar{\mathbf{u}}} \quad & \left\|\mathbf{X} - \mathbf{X}_{ref}\right\|_{\mathbf{L}}^2 + \|\bar{\mathbf{u}}\|_{\mathbf{K}}^2 \\ \text{subject to} \quad & -\mu\bar{\mathbf{u}}_z \leq \bar{\mathbf{u}}_x \leq \mu\bar{\mathbf{u}}_z \quad -\mu\bar{\mathbf{u}}_z \leq \bar{\mathbf{u}}_y \leq \mu\bar{\mathbf{u}}_z \\ & \mathbf{u}_{min} \leq \bar{\mathbf{u}}_z \leq \mathbf{u}_{max} \qquad \mathrm{G}(\mathcal{G})\bar{\mathbf{u}} = \mathbf{0} \end{aligned} \tag{14}$$

where $\mathbf{X}_{ref} \in \mathbb{R}^{15n}$ is the stacked vector of desired states along the prediction horizon[3], vectors $\bar{\mathbf{u}}_x \in \mathbb{R}^{4n}$, $\bar{\mathbf{u}}_y \in \mathbb{R}^{4n}$ and $\bar{\mathbf{u}}_z \in \mathbb{R}^{4n}$ correspond to the components of vector $\bar{\mathbf{u}}$ associated to x, y and z, respectively, of the GRFs, $\mu \in \mathbb{R}$ is the friction coefficient, $\mathbf{u}_{min} \in \mathbb{R}^{4n}$ and $\mathbf{u}_{max} \in \mathbb{R}^{4n}$ are the limits on the z component of $\bar{\mathbf{u}}$, matrix $\mathrm{G} \in \mathbb{R}^{12n\times12n}$ is a matrix that selects the components of the GRFs that are in contact according to gait $\mathcal{G}$, and matrices $\mathbf{L}$ and $\mathbf{K}$ are weighting matrices. This optimization problem is a QP and can be efficiently solved by several off-the-shelf solvers. After solving (14), we compute the desired wrench coming from the MPC as

$$\mathbf{w}_{MPC} = \sum_{i=1}^{l} \begin{bmatrix} \mathbf{p}_i \times \mathbf{F}_i^* \\ \mathbf{F}_i^* \end{bmatrix} \tag{15}$$

[1]Herein, $\mathbf{g}$ is appended in the state vector to reach the form given in (11)

[2]If $\theta \approxeq \phi \approxeq 0$ then: $\dot{\mathbf{x}}(t) = \mathbf{A}(\psi)\mathbf{x}(t) + \mathbf{B}(\psi, \mathbf{p}_{LF}, ..., \mathbf{p}_{RH})\mathbf{u}(t)$

[3]We redefine $\mathbf{x}_{ref}[k] = [\Theta_{ref}^\top[k] \ \mathbf{r}_{ref}^\top[k] \ (\mathbf{T}(\Theta_{ref}[k])\dot{\Theta}_{ref}[k])^\top \ \dot{\mathbf{r}}_{ref}^\top[k] \ \mathbf{g}^\top]^\top$ to match the state definition of $\mathbf{x}$

where $\mathbf{F}_i^*$ is the optimized GRF of foot i, extracted from the first 12 entries of the optimized control input vector $\bar{\mathbf{u}}^*$.

d) Leg Inertia Compensation: The MPC model used for prediction neglects leg inertia. This assumption is acceptable for quasi-static motions. However, if the leg-body weight ratio is significantly large, the wrench exerted by the legs on the body plays a significant role in the dynamics. We compensate for these effects in a simple, yet effective, manner. The floating-base dynamics of a robot can be described by

$$\begin{bmatrix} \mathbf{M}_u & \mathbf{M}_{ua} \\ \mathbf{M}_{au} & \mathbf{M}_a \end{bmatrix} \begin{bmatrix} \dot{\mathbf{v}} \\ \ddot{\mathbf{q}}_j \end{bmatrix} + \begin{bmatrix} \mathbf{h}_u \\ \mathbf{h}_a \end{bmatrix} = \begin{bmatrix} \mathbf{0} \\ \boldsymbol{\tau}_j \end{bmatrix} + \begin{bmatrix} \mathbf{J}_{c,u}^\top \\ \mathbf{J}_{c,a}^\top \end{bmatrix} \mathbf{F} \tag{16}$$

where $\mathbf{v} \in \mathbb{R}^6$ is the floating-base robot velocity, $\mathbf{q} \in \mathbb{R}^{n_j}$ is the joint configuration, $\mathbf{M}_u \in \mathbb{R}^{6\times 6}$ and $\mathbf{M}_a \in \mathbb{R}^{6\times n_j}$ are the direct un-actuated and actuated parts of the joint-space inertia matrix, whereas $\mathbf{M}_{ua} \in \mathbb{R}^{6\times n_j}$ and $\mathbf{M}_{au} \in \mathbb{R}^{n_j\times 6}$ correspond to the cross terms between actuated and un-actuated parts of the joint-space inertia matrix, $\mathbf{h}_u \in \mathbb{R}^6$ and $\mathbf{h}_a \in \mathbb{R}^{n_j}$ are the un-actuated and actuated vectors of Coriolis, centrifugal and gravitational terms, $\tau_j \in \mathbb{R}^{n_j}$ is the vector of joint torques, $\mathbf{J}_{c,u} \in \mathbb{R}^{n_c\times 6}$ and $\mathbf{J}_{c,a} \in \mathbb{R}^{n_c\times n_j}$ are the un-actuated and actuated contact Jacobians and $\mathbf{F}$ is the vector of GRFs. The matrix $\mathbf{M}_{ua}$ maps the joint accelerations to the robot spatial force acting on the floating-base of the robot, namely

$$\mathbf{w}_l = \mathbf{M}_{ua}\ddot{\mathbf{q}}_j \tag{17}$$

In (17), the $\mathbf{w}_l$ can be computed directly using measurements coming from the sensors. However, using the actual joint acceleration might lead to high frequency wrench signals. Instead, We use the desired joint acceleration $\ddot{\mathbf{q}}_{j,d}$ coming from the torque mapper (see Fig. 1). Then, the leg inertia compensation wrench is given by $\mathbf{w}_l = \mathbf{M}_{ua}\ddot{\mathbf{q}}_{j,d}$. Thus, the total desired wrench is given by $\mathbf{w}_d = \mathbf{w}_{MPC} + \mathbf{w}_l$[4].

IV. Results

We performed simulations on HyQReal [13], a hydraulically actuated quadruped robot. The leg configuration of the robot is shown in Fig. 1. We use Gazebo [21] to perform our simulations. Control commands are executed at a frequency of 250 Hz. Wrench values from the MPC controller $\mathbf{w}_{MPC}$ are sent at a maximum frequency of 25 Hz and we use a ZOH in between control signals. The prediction horizon is set to comprise 2 gait cycles, partitioned in 20 samples. We solve the QP in (14) with a modified version of uQuadProg++ [22] to work with the C++ linear algebra library, Eigen. The signal $\mathbf{w}_{MPC}$ is computed at 25 Hz to give enough time to the solver to compute a feasible solution. The leg inertia compensation wrench $\mathbf{w}_l$ is computed at task frequency. The mapping is done using the Grid Map interface from [23]. Weighting marices are chosen as $\mathbf{L} = \mathbf{1}_{15n}$ and $\mathbf{K} = (1\times 10^{-9})\mathbf{1}_{12n}$, where $\mathbf{1}_a$ defines the $a \times a$ identity matrix.

Simulations

We perform three different simulations to assess the improvements in foothold prediction and locomotion robustness. Below we explain in detail the outcome of these tests.

[4]This total wrench is distributed between the contact feet using the torque mapper shown in Fig. 1 [15]

TABLE I: Root mean square and maximum absolute value of the foothold prediction error

		LF	RF	LH	RH
QP+LI+GC	$RMS(e)$	0.012	0.012	0.012	0.012
	$\max\|e\|$	0.095	0.095	0.088	0.082
MPC+IC	$RMS(e)$	0.009	0.007	0.007	0.009
	$\max\|e\|$	0.0740	0.0611	0.0604	0.0761

a) Leg Inertia Compensation: We perform simulations commanding the robot to trot on flat terrain with a forward velocity $\mathbf{V}_f$ of magnitude 0.5 m/s, a step frequency f_s of 1.4 Hz, and a duty factor D_f of 0.6. We start the simulation with our previous trunk controller [15]. We keep $\mathbf{V}_f$ and change the controller configuration as the robot continues to trot. There are six possible configurations shown in Fig. 3, which combine the following control components: *1*) QP: standard QP trunk controller *2*) LI: stance leg joint impedance (PD controller at joint level) *3*) GC: gravity compensation *4*) IC: leg inertia compensation and *5*) MPC: model predictive controller. Figure 3 shows the error in velocity with respect to the commanded $\mathbf{V}_f$ for one of the trials. The vertical dashed red lines indicate the moments when the controller configuration was changed. We check six different configurations, although we would like to stress that configuration C3 (see Fig. 3) acts merely as a a transition between the standard QP and the MPC. This is because the MPC controller already compensates for the gravitational effects in the model. It can be noticed that when the inertia compensation wrench is applied, the accelerations of the body are greatly reduced. The best performing configuration corresponds to the configuration C5 (fifth portion of graph in Fig. 3). Under this configuration, the robot dynamics resemble more those of the MPC model (which neglects leg inertia), since the leg inertia is being accounted for outside of the optimization. C2 presents larger errors in velocity tracking, but its response is smoother with respect to C5. The sharp changes in velocity in C5 could be reduced by carefully choosing the weights in $\mathbf{K}$ and $\mathbf{L}$ of (14).

b) Foothold Predictions and Robustness in the Presence of Disturbances: For the second simulation, the robot is also commanded to trot on flat terrain with the same gait parameters as in the first simulation. This time, we perturb the robot three times with 700 N of force with a duration of 0.1 s. Table I shows the root mean square (RMS) error and the maximum absolute value of the foothold prediction error for this simulation. The table helps us to compare the previous controller configuration with the MPC-based controller with leg inertia compensation. The RMS prediction error when using MPC and leg inertia compensation is between 25% and 41% less with respect to the previous controller configuration. This represents between 3 mm and 5 mm of improvement. However, even if the average of the error is low in both cases, a single wrong prediction compromises the robot stability. The maximum absolute value of the error is more representative of the reliability of

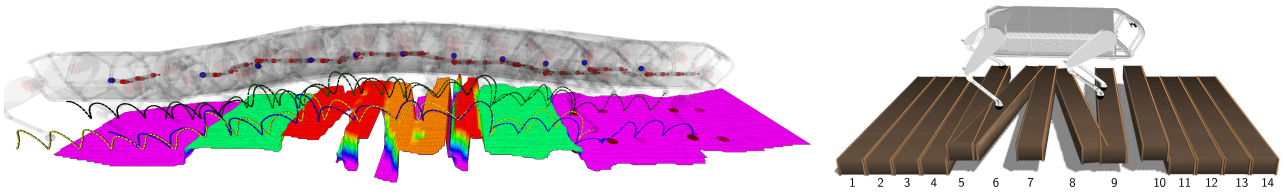

Fig. 2: Left: series of snapshots of the HyQReal robot moving through the scenario. Blue spheres correspond to the position of the center of mass at the moment when the snapshot was taken and the red spheres show the reference position for the COM along the prediction horizon. The positions of the feet are indicated by the colored dashed lines. The elevation map is built using the vision sensors. Right: scenario designed to test the locomotion strategy proposed in this paper. Each beam is 15 cm height and 20 cm wide. Beams 1 to 4 and 11 to 14 are located at ground level. Beams 5 to 7 and 10 are located 15 cm above ground level. Beam 8 is located 12 cm above ground.

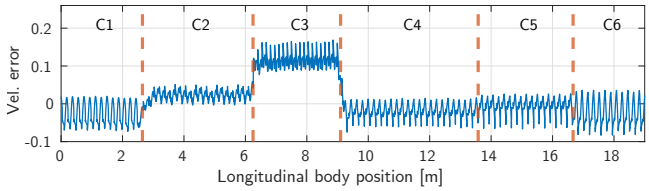

Fig. 3: Velocity with different control strategies. The red dashed lines indicate the moments when the configuration was changed. The controller configurations are: C1 = QP + LI + GC; C2 = QP + LI + GC + IC; C3 = QP + LI + IC; C4 = MPC + LI + IC; C5 = MPC + IC; C6 = MPC. The abbreviations stand for: a) QP: standard QP trunk controller, b) LI: stance leg joint impedance, c) GC: gravity compensation, d) IC: leg inertia compensation and, e) MPC: model predictive controller.

the prediction under disturbances. In this case, the reduction of the error is between 7% and 36%. This represents between 0.6 cm and 3 cm of reduction of the foothold prediction error when using the MPC with the leg inertia compensation.

c) Locomotion on Challenging Terrain: To verify the improvement in performance regarding locomotion on difficult terrain, we designed the scenario in Fig. 2. The robot is commanded to trot with a forward velocity of 0.4 m/s, a step frequency of 1.4 Hz and a duty factor of 0.6. We use the VFA to select appropriate footholds with two different control configurations *1*) QP + LI + GC (C1) and *2*) MPC + IC (C5). To test the performance repeatability we did four trials with each configuration. Figure 4 shows the pitch angle, forward velocity, body height and an example of the foot trajectories for one of the trials with configuration C5. Figure 2 shows 11 overlapped snapshots of the RVIZ visualization as the robot crosses the scenario, builds the map, and adjusts its footholds on the fly. Figure 2 also shows the reference trajectory of the center of mass given at the specific moment when the snapshot was taken, and the foot trajectories as the robot moves through the scenario.

Figure 4 shows that all four different trials using configuration C5 were successful and the variations in linear velocity, pitch and body height are significantly reduced with respect to C1. For this last configuration, the robot was not able to reach the end of the scenario in any of the trials, mainly due to errors in foothold prediction and variations on the body velocity. This task shows the mutual benefits between the VFA and the MPC. The foothold prediction error is reduced when using the strategy here presented. Specifically, in the case of the MPC + IC, for all four trials and all legs, the

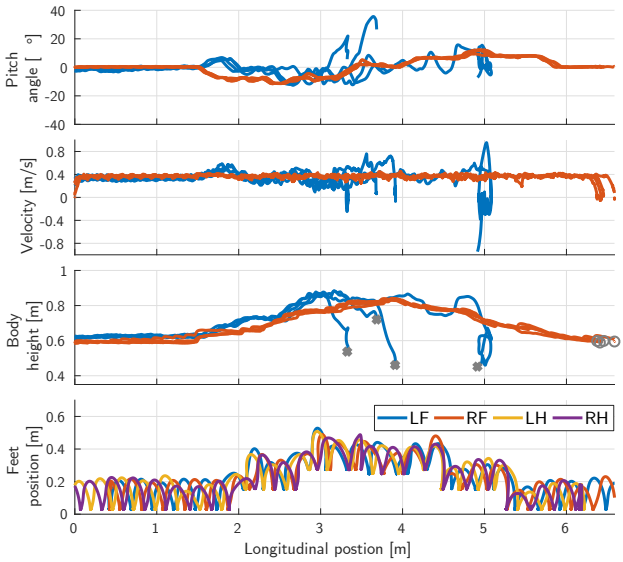

Fig. 4: Results for the scenario crossing simulation. The top three plots show pitch angle, velocity error, and body height. Blue lines correspond to trials with configuration C1 and red lines correspond to C5. Foot trajectories corresponding to one of the succesful trials are shown at the bottom part of the figure.

maximum absolute value of the error in foothold prediction was 10 cm, while in the case of the previous configuration the error was 14 cm.

V. Conclusions and Future Work

We developed a dynamic locomotion strategy to traverse difficult terrain using visual information only coming from on-board sensors. We based this strategy on the combination of an MPC-based controller and a CNN-based foothold adaptation scheme (namely the VFA). We showed that the interaction between these approaches is mutually beneficial and improves locomotion reliability and robustness. We also demonstrated that considering a compensation term accounting for the wrench due to the inertia of the legs improves the performance of the MPC-based controller, due to a closer resemblance to the model used for state prediction. The various simulations validated these improvements. A major limitation is related to the maximum frequency at which the MPC controller can be computed. For the current prediction horizon length, this is limited to 25 Hz. As future work, we plan to validate the strategy developed here in experiments with the hydraulically actuated quadruped robot HyQReal.

REFERENCES

[1] M. Kalakrishnan, J. Buchli, P. Pastor, and S. Schaal, "Learning locomotion over rough terrain using terrain templates," in *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Oct 2009, pp. 167–172.

[2] P. Fankhauser, M. Bjelonic, C. Dario Bellicoso, T. Miki, and M. Hutter, "Robust rough-terrain locomotion with a quadrupedal robot," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, 2018, pp. 5761–5768.

[3] 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.

[4] Y. Lin, B. Ponton, L. Righetti, and D. Berenson, "Efficient Humanoid Contact Planning using Learned Centroidal Dynamics Prediction," in *2019 International Conference on Robotics and Automation (ICRA)*, May 2019, pp. 5280–5286.

[5] S. Tonneau, A. Del Prete, J. Pettr, C. Park, D. Manocha, and N. Mansard, "An efficient acyclic contact planner for multiped robots," *IEEE Transactions on Robotics*, vol. 34, no. 3, pp. 586–601, June 2018.

[6] F. Farshidian, M. Neunert, A. W. Winkler, G. Rey, and J. Buchli, "An efficient optimal planning and control framework for quadrupedal locomotion," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*, May 2017, pp. 93–100.

[7] 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*, vol. 3, no. 3, pp. 1560–1567, July 2018.

[8] P. Fernbach, S. Tonneau, and M. Tax, "CROC: Convex Resolution of Centroidal Dynamics Trajectories to Provide a Feasibility Criterion for the Multi Contact Planning Problem," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Oct 2018, pp. 1–9.

[9] A. Herzog, N. Rotella, S. Schaal, and L. Righetti, "Trajectory generation for multi-contact momentum control," in *2015 IEEE-RAS 15th International Conference on Humanoid Robots (Humanoids)*, Nov 2015, pp. 874–880.

[10] M. Kalakrishnan, J. Buchli, P. Pastor, M. Mistry, and S. Schaal, "Learning, planning, and control for quadruped locomotion over challenging terrain," *The International Journal of Robotics Research*, vol. 30, no. 2, pp. 236–258, 2011.

[11] D. Belter, J. Bednarek, H. Lin, G. Xin, and M. Mistry, "Single-shot Foothold Selection and Constraint Evaluation for Quadruped Locomotion," in *2019 International Conference on Robotics and Automation (ICRA)*, May 2019, pp. 7441–7447.

[12] O. Villarreal, V. Barasuol, M. Camurri, L. Franceschi, M. Focchi, M. Pontil, D. G. Caldwell, and C. Semini, "Fast and Continuous Foothold Adaptation for Dynamic Locomotion Through CNNs," *IEEE Robotics and Automation Letters*, vol. 4, no. 2, pp. 2140–2147, April 2019.

[13] C. Semini, V. Barasuol, M. Focchi, C. Boelens, M. Emara, S. Casella, O. Villarreal, R. Orsolino, G. Fink, S. Fahmi, G. Medrano-Cerda, and D. G. Caldwell, "Brief introduction to the quadruped robot HyQReal," in *Istituto di Robotica e Macchine Intelligenti (I-RIM)*, 2019.

[14] V. Barasuol, J. Buchli, C. Semini, M. Frigerio, E. R. De Pieri, and D. G. Caldwell, "A reactive controller framework for quadrupedal locomotion on challenging terrain," in *2013 IEEE International Conference on Robotics and Automation*, 2013, pp. 2554–2561.

[15] 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.

[16] J. Di Carlo, P. M. Wensing, B. Katz, G. Bledt, and S. Kim, "Dynamic Locomotion in the MIT Cheetah 3 Through Convex Model-Predictive Control," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Oct 2018, pp. 1–9.

[17] G. Bledt, M. J. Powell, B. Katz, J. Di Carlo, P. M. Wensing, and S. Kim, "MIT Cheetah 3: Design and Control of a Robust, Dynamic Quadruped Robot," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Oct 2018, pp. 2245–2252.

[18] M. Neunert, M. Stuble, M. Giftthaler, C. D. Bellicoso, J. Carius, C. Gehring, M. Hutter, and J. Buchli, "Whole-Body Nonlinear Model Predictive Control Through Contacts for Quadrupeds," *IEEE Robotics and Automation Letters*, vol. 3, no. 3, pp. 1458–1465, July 2018.

[19] M. Giftthaler, M. Neunert, M. Stuble, M. Frigerio, C. Semini, and J. Buchli, "Automatic differentiation of rigid body dynamics for optimal control and estimation," *Advanced Robotics*, vol. 31, no. 22, pp. 1225–1237, 2017.

[20] B. Aceituno-Cabezas, C. Mastalli, H. Dai, M. Focchi, A. Radulescu, D. G. Caldwell, J. Cappelletto, J. C. Grieco, G. Fernndez-Lpez, and C. Semini, "Simultaneous Contact, Gait, and Motion Planning for Robust Multilegged Locomotion via Mixed-Integer Convex Optimization," *IEEE Robotics and Automation Letters*, vol. 3, no. 3, pp. 2531–2538, 2018.

[21] N. Koenig and A. Howard, "Design and use paradigms for gazebo, an open-source multi-robot simulator," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, vol. 3, Sep. 2004, pp. 2149–2154 vol.3.

[22] L. Di Gaspero and E. Moyer, "Quadprog++," *URL http://quadprog. sourceforge. net/.[Online] Available: http://quadprog. sourceforge. net*, 1998.

[23] P. Fankhauser, M. Bloesch, C. Gehring, M. Hutter, and R. Siegwart, "Robot-centric elevation mapping with uncertainty estimates," in *International Conference on Climbing and Walking Robots (CLAWAR)*, Apr 2014.