Implementation of a Reactive Walking Controller for the New Open-Hardware Quadruped Solo-12

Pierre-Alexandre Léziart, Thomas Flayols, Felix Grimminger, Nicolas Mansard, Philippe Souères

Implementation of a Reactive Walking Controller for the New Open-Hardware Quadruped Solo-12

Pierre-Alexandre Léziart [a,*], Thomas Flayols [a], Felix Grimminger,[b] Nicolas Mansard,[a,c] and Philippe Souères [a]

***Abstract*— This paper aims at showing the dynamic performance and reliability of the low-cost, open-access quadruped robot Solo-12, which is developed within the framework of Open Dynamic Robot Initiative. It presents the implementation of a state-of-the-art control pipeline, close to the one that was previously implemented on Mini Cheetah, which implements a model predictive controller based on the centroidal dynamics to compute desired contact forces in order to track a reference velocity. Different contributions are proposed to speed up the computation process, notably at the level of the state estimation and the whole body controller. Experimental results demonstrate that the robot closely follow the reference velocity while being highly reactive and able to recover from perturbations.**

I. Introduction

Performing dynamic locomotion with legged robots in real-life environments raises challenging issues in terms of computational efficiency, embeddability, state estimation and control robustness to both external disturbances and unexpected obstacles. Increasingly impressive running behaviors have been shown in recent years with both bipeds [1], [2], [3] and quadrupeds [4], [5], [6], though some performances are limited to flat floor [1], heavily rely on specific system dynamics [2], or are undocumented [4].

Over the years various methods have been developed to perform dynamic locomotion with legged robots. In [7] a ZMP-based motion planner enabled the ANYmal quadruped [8] to display a wide range of gaits including a squat jump. A similar approach was successfully applied on IIT HyQ [9] and coupled with an Any-time-Repairing A* (ARA*) planner that explores a tree of possible body motion primitives [10]. The Mini Cheetah [11] quadruped developed at MIT has demonstrated high speed running with aerial phase and various gaits using a fast online model predictive controller to compute an optimal reaction force profile [6]. More data-oriented methods have also been tested, for instance on Cassie with Deep Reinforcement Learning to train control policies in simulation and then transfer them to the real hardware [12], or to make locomotion behaviours emerge using hierarchical reinforcement learning [13]. All these methods usually require a preliminary development phase in simulation to fully demonstrate their potential and better grasp their advantages and drawbacks before being applied to real robots. With experimental validation comes the risk of breaking mechanical parts when pushing robots to their limits. For this reason, experimental test are often limited to conservative secure movements especially when using high-end robots whose cost easily reaches ten of thousands of dollars [14], [15].

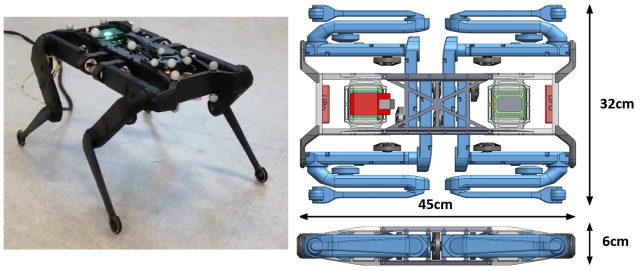


Fig. 1: The open hardware 12 dof version of the Solo quadruped weight less that 2.5 kg. Low cost, easy to repair and highly documented, it is a platform of choice for experimentation of highly dynamic controllers, as well as for the teaching of legged robotics.

The objective of the Open Dynamic Robot Initiative, which is at the heart of this paper, is specially to answer this problem by developing an open-access, low-cost and low-complexity quadruped robot with mostly 3D printed and off-the-shelves components [16], [17]. This project aims at providing the community with reliable legged robotic platforms that can be easily maintained and repaired and could benefit from numerous contributions in their development. The robot Solo-12 presented in figure 1, is a 12 degrees of freedom (DoF) quadruped that includes 3DoF in each leg (adduction-abduction at the hip, flexion at the knee and at the ankle). It is an extension to the Solo-8 prototype, which only included 2 DoF in each leg [17]. The aim of this paper is to demonstrate that, despite its simplicity, this quadruped can be endowed with state of the art locomotion capabilities. To this end, this paper describes the implementation on Solo-12 of a complete control pipeline which closely follows the scheme developed in [6] for Mini Cheetah. This control scheme relies on a model predictive control block that uses a simplified centroidal model of the quadruped to output contact forces that should be applied on the ground to reach a reference velocity. A whole-body control block then generates the position, velocity and torques that should be followed by the actuators. Beyond showing the potential of the Solo platform, this paper includes two main contributions with respect to [6]. The first one lies in the introduction of a simplified estimation procedure which involves two complementary filters and provides a simple, rapid and reliable reconstruction

[a] LAAS-CNRS, Université de Toulouse, CNRS, Toulouse, France
[b] Max Planck Institute for Intelligent Systems Tubingen, Germany
[c] Artificial and Natural Intelligence Toulouse Institute, France
[*] corresponding author: paleziart@laas.fr

This work has been supported by the MEMMO European Union project within the H2020 Program under Grant Agreement No. 780684 and by the RoboCom++ European project.

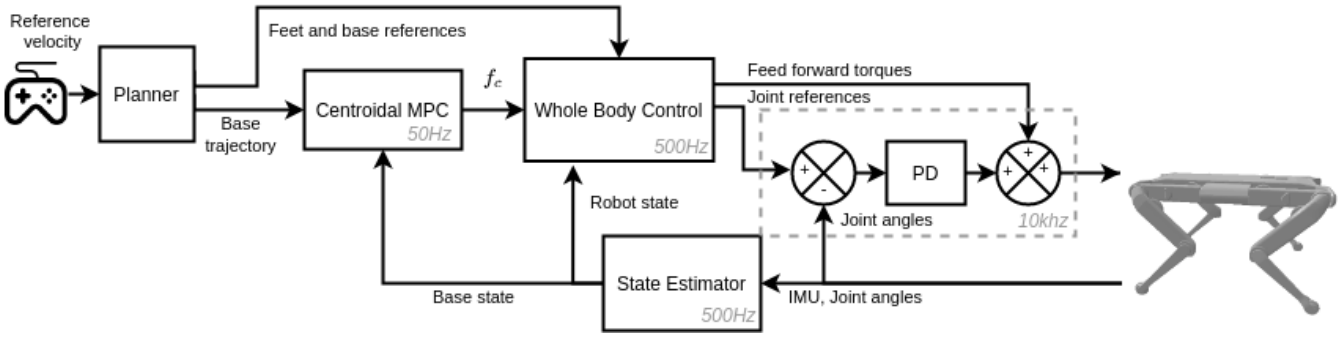


Fig. 2: Reactive walking controller architecture

of the robot state. The second contribution is two-fold and concerns the whole body control module. First, taking benefit from the constraints at the feet and the base of the robot, the inverse kinematics is solved without having to tackle a complex hierarchical problem. Then, the computation of the feedforward torques is simplified thanks to a reformulation of the optimization problem as box-QP that can be easily solved through the computation of pseudo-inverses. Finally, simulations and experimental result are described to demonstrate the performance of Solo-12 through the implementation of this control scheme. The results show that the robot is highly reactive, able follow closely the reference input velocity, and shows a robust behaviour despite unexpected perturbations. The algorithms are open-source and coded in Python for ease of use and modification [18].

The paper is organized as follows: A rapid description of the robot Solo robot is given in section II. The control architecture is presented in Section III. It includes an overview of the control scheme, a description of the foot trajectory generator and of the estimation module. The whole body controller then is presented in section IV. Finally, simulations and experimental results are described in sections V and VI.

II. The Solo quadruped

The Open Dynamic Robot Initiative is a collaborative project that originated in an effort to design an open source, low cost and low complexity actuator module that can be used to build different types of torque-controlled robots with mostly 3D printed and off-the-shelves components [16], [17]. One of the motivations of this project is to allow an easy benchmarking of different control paradigms implemented on a same low-cost and easy to repair platform. The actuator module consists of a brushless outrunner motor, a high resolution optical encoder and a dual stage timing belt transmission. The module has a segment length of 160mm, weighs 150g and outputs 2,5Nm at 12A. A custom motor controller using a field oriented control algorithm and a local joint impedance controller similarly to [19] is used to drive this actuator. The quadruped robot Solo12 is a new member of the growing family of robots using this actuator module. It is composed of 12 identical modules with only variation of their shell enclosure. The platform is equiped with an inertial measurement unit embedding an extended Kalman filter for attitude estimation and a custom network bridge to close the control loop with a distant computer at 1kHz via Wifi or Ethernet. Note that this first prototype does not include computing power (other than the joint controller) nor embedded power source. Although not suited for industrial applications, this robot allows the implementation of highly dynamic controllers at the state of the art level, as demonstrated in this paper. The quality of its documentation and the open access of all its components, from high level control interfaces to mechanical design and control electronics, make it a platform of choice for research in legged robotics as well as for teaching.

III. Control Architecture

A. Overview

The control scheme of the quadruped is described in Fig. 2. As stated before, it closely follows the pipeline proposed in [6]. The system receives as inputs the desired gait sequence and the reference velocity, either specified by a user with a joystick or a by a higher level controller. This information is processed by a footstep planner which outputs the desired locations of the upcoming footsteps. These locations should be reached by feet at the end of the swing phase during which they follow the reference position, velocity and acceleration outputted by the trajectory generator. The model predictive control (MPC) block uses a simplified centroidal model of the quadruped to find the contact forces that should be applied by the feet in stance phase to follow as closely as possible the reference velocity over a prediction horizon. The whole-body control block makes the link between the high-level control (desired feet trajectories and desired contact forces) and the low-level control (torques sent to the drivers of the actuators). It relies on a model of the whole-body dynamics and outputs the feedforward torques and the desired angular accelerations. A PD+ controller is used to provide the feedback torques based on the difference between the desired angular positions and velocities of actuators and the current ones. The estimator includes complementary filters that combine information coming from the inertial measurement unit and from forward kinematics with feet in contact in order to evaluate the velocity and position of the base. The footstep planner and model predictive control (MPC) are textbook reimplementation of the ones presented in [6] and as such they will not be developed in this paper.

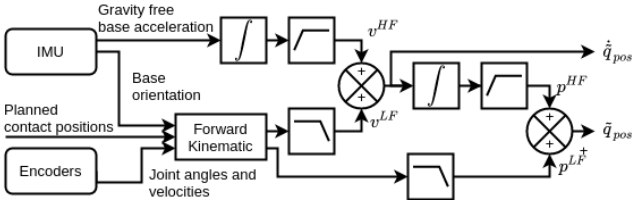


Fig. 3: A dual cascaded complementary filter provides base position and velocity estimate while being easy to tune and simple to implement

B. Foot trajectory generator

During the swing phases the feet have to be guided from their current position on the ground to the target position outputted by the footstep planner. Foot trajectory generators (one per foot) are used to generate reference trajectories in position, velocity and acceleration. Non-slipping constraints are enforced (zero velocity and acceleration foot during take-off and landing) with the additional constraint to reach a predefined height at the apex of the swing phase. This is done by using two 5-th order polynomials for the two horizontal components and one 6-th order polynomial for the vertical one.

Target locations that are outputted by the footstep planner are constantly changing because they are directly linked to the current velocity of the robot, which is not exactly the same from one time step to another. For this reason these locations are locked 70 ms before landing since a displacement at the last moment would create a conflict between the zero horizontal velocity command enforced to avoid slipping and the need for a non-zero horizontal velocity to correct the position of the swinging foot and ensure landing at the expected location.

C. State Estimation

The estimation is made easier thanks to the on-board inertial measurement unit (IMU) which embeds an extended Kalman filter providing a gravity free acceleration as well as roll, pitch and yaw angles estimate. Since the orientation and angular velocities of the base are already estimated/measured by the IMU, we can use a decoupled linear approach [20].

To estimate linear base velocity and position we use a dual cascaded complementary filter [21] that fuses the information coming from the IMU and the forward kinematics, based on the contact points with the ground, as shown in Fig. 3.

IV. Efficient whole-body control

As our main theoretical contribution we introduce a more computationally efficient version of the whole-body controller used in [6]. The role of this controller is to convert the desired contact forces provided by the MPC and the reference feet position, velocity and acceleration given by the trajectory generators into torque, position and velocity commands that are sent to the low level motor drivers. The whole body control relies on two successive blocks: inverse kinematics (IK) and a feedforward torque computation.

A. Computing desired accelerations

The first step is to compute command accelerations $\ddot{q}_{IK}$ by IK of the full model of the quadruped. The IK scheme is defined by 3 tasks:

- Keep the base at constant height and follow the reference horizontal motion resulting from the reference horizontal velocity of the MPC (3 DoF for base position)
- Keep the base orientation horizontal and follow the reference yaw orientation resulting from the reference yaw angular velocity of the MPC (3 DoF for base orientation)
- Follow the reference trajectory of the swing feet while maintaining feet in contact immobile (3 DoF per leg, 12 DoF in total)

For a quadruped robot with 18 DoF these tasks fully constrain the system but are compatible as the number of DoF is sufficient to satisfy each of them independently. While [6] suggested a hierarchical IK scheme, we can indeed note that the hierarchy here is useless. We rather propose to take advantage of the particular tasks that are almost decoupled and can be very efficiently inverted. By stacking all the task functions in a global vector according to the above description order, a global task Jacobian can be defined as:

$$J = \begin{bmatrix} {}^{o}\mathcal{R}_b & & & & & \\ & {}^{o}\mathcal{R}_b & & & & \\ {}^{o}\mathcal{R}_b & {}^{b}\mathcal{T}_1 \times {}^{o}\mathcal{R}_b & J_1 & & & \\ {}^{o}\mathcal{R}_b & {}^{b}\mathcal{T}_2 \times {}^{o}\mathcal{R}_b & & J_2 & & \\ {}^{o}\mathcal{R}_b & {}^{b}\mathcal{T}_3 \times {}^{o}\mathcal{R}_b & & & J_3 & \\ {}^{o}\mathcal{R}_b & {}^{b}\mathcal{T}_4 \times {}^{o}\mathcal{R}_b & & & & J_4 \end{bmatrix} \quad (1)$$

with ${}^{o}\mathcal{R}_b$ the rotation matrix of the base in world frame, ${}^{b}\mathcal{T}_i$ the position of the i-th foot with respect to the base and J_i the Jacobian of position of the i-th foot. Inverting J to get command acceleration vector $\ddot{q}_{IK}$ thus amounts to inverting ${}^{o}\mathcal{R}_b$ (transposition) to get base command accelerations in position $\ddot{q}_{pos}$ and orientation $\ddot{q}_{ang}$ and computing J_i^{-1} to get feet command accelerations $\ddot{q}_i, \forall i \in 1..4$. As J_i are 3 by 3 invertible matrices damping can be introduced when inverting them although we did not see any gains in practice. Finally we obtain:

$$\ddot{q}_{IK} = (\ddot{q}_{pos}, \ddot{q}_{ang}, \ddot{q}_1, \ddot{q}_2, \ddot{q}_3, \ddot{q}_4) \quad (2)$$

$$\ddot{q}_{pos} = {}^{o}\mathcal{R}_b^T \ddot{x}_{pos}^{cmd} \quad (3)$$

$$\ddot{q}_{ang} = {}^{o}\mathcal{R}_b^T \ddot{x}_{ang}^{cmd} \quad (4)$$

$$\forall i \in 1..4,\ \ddot{q}_i = J_i^{-1}(\ddot{x}_i^{cmd} - \ddot{x}_{pos}^{cmd} - {}^{b}\mathcal{T}_3 \times \ddot{x}_{ang}^{cmd}) \quad (5)$$

with $\ddot{x}_{pos}^{cmd}$ the acceleration of the base position task, $\ddot{x}_{ang}^{cmd}$ the acceleration of the base orientation task and $\ddot{x}_i^{cmd}$ the acceleration of the i-th foot task ($\forall i \in 1..4$). These accelerations are defined by:

$$\ddot{x}_j^{cmd} = K_{p,j}(x_j^{des} - x_j) + K_{d,j}(\dot{x}_j^{des} - \dot{x}_j) + \ddot{x}_j^{des} \quad (6)$$

with $K_{p,j}$ and $K_{d,j}$ the position and velocity feedback gains associated with task $j \in \{pos, ang, 1..4\}$. x_j^{des}, x_j, $\dot{x}_j^{des}$ and $\dot{x}_j$ are respectively the desired and current position and

velocity associated with task j. The accelerations $\ddot{q}_{pos}$, $\ddot{q}_{ang}$ and $\ddot{q}_i$ are sent to the second step of the whole-body control to compute feedforward torque commands.

B. Computing reference positions and velocities

As the motors of Solo are not torque-driven, the low-level controller feedbacks in impedance. Reference positions and velocities for the impedance controller are also computed. The reference articular configuration is computed to match the base placement decided by the MPC with the feet at their respective contact or swing positions.

$$\forall i \in 1..4,\ q_i = q_i^{t-1} + J^{-1}(x_i^{des} - x_i^{t-1}) \tag{7}$$

where q_i^{t-1} is the reference configuration of the i-th foot computed at the previous control cycle and x_i^{t-1} is the foot position computed from this configuration.

The reference articular velocity is similarly computed by:

$$\forall i \in 1..4,\ \dot{q}_i = J^{-1}\dot{x}_i^{des} \tag{8}$$

where the desired velocity $\dot{x}_i^{des}$ is determined from the swing reference trajectory.

These references q_i^{des} and $\dot{q}_i^{des}$ are only computed from reference quantities coming from the swing trajectories. There is no direct feedback here. The only direct feedback is going through the command accelerations which are then translated to feedback torques.

C. Feedforward torques computation

Following [6] we compute the feedforward torques using relaxation variables $\delta_{\ddot{q}}$ and δ_f. The QP problem tries to find contact forces $f = f_{MPC} + \delta_f$ and accelerations $\ddot{q} = \ddot{q}_{IK} + \delta_{\ddot{q}}$ that are as close as possible to the force references decided by the MPC and the command accelerations decided by the IK while taking into account the floating base dynamics.

$$\min_{\delta_{\ddot{q}}, \delta_f} \delta_{\ddot{q}}^T Q_1 \delta_{\ddot{q}} + \delta_f^T Q_2 \delta_f \tag{9}$$

$$\text{such that } f_{MPC} + \delta_f \in \mathbf{K} \tag{10}$$

$$S(M(\ddot{q}_{IK} + \begin{bmatrix}\delta_{\ddot{q}} \\ 0\end{bmatrix}) + b) = SJ_c^T(f_{MPC} + \delta_f) \tag{11}$$

with M the generalized mass matrix, b the vector of nonlinear and gravitational forces, S the selection matrix of the underactuated dynamics, J_c the augmented contact Jacobian and $\mathbf{K}$ the space inside the friction cone linearized to the first order. Similarly to IK, we introduce a more computionally efficient way to compute feedforward torques from the reaction forces f_{MPC} outputted by the MPC and the accelerations $\ddot{q}_{IK}$. We transform the quadratic programming problem introduced in [6] into an equivalent problem faster to solve as it amounts to a few matrix inversions. It will be useful to separate the variables between base and joints:

$$M = \begin{bmatrix} Y & M_u \\ M_u^T & M_a \end{bmatrix} \tag{12}$$

$$J_c^T = \begin{bmatrix} X & J_a \end{bmatrix} \tag{13}$$

where subscript u and a refer to the underactuated and actuated parts respectively.

Using the underactuated part of (11) $\delta_{\ddot{q}}$ is then expressed as an affine expression of δ_f:

$$\delta_{\ddot{q}} = A\delta_f + \gamma \tag{14}$$

$$A = Y^{-1}X \tag{15}$$

$$\gamma = Y^{-1}(Xf_{MPC} - Y\ddot{q}_{IK,u} - M_u\ddot{q}_{IK,a} - b_u) \tag{16}$$

Here $X = \begin{bmatrix} \mathcal{R} & p \times \mathcal{R} \\ & \mathcal{R} \end{bmatrix}$ the adjoint matrix and $Y = \begin{bmatrix} mI_3 & \\ & I_c \end{bmatrix}$ the spatial inertia are very structured matrices while $Y\ddot{q}_{IK,u} + M_u\ddot{q}_{IK,a} + b_u$ is computed by a cheap RNEA evaluation [22].

$\delta_{\ddot{q}}$ is then replaced in (9) using (14):

$$\min_{\delta_f} \delta_f^T H \delta_f + 2\delta_f^T g \tag{17}$$

$$\text{such that } f_{MPC} + \delta_f \in \mathbf{K} \tag{18}$$

$$H = A^T Q_1 A + Q_2 \tag{19}$$

$$g = A^T Q_1 \gamma \tag{20}$$

$f_{MPC} + \delta_f \in \mathbf{K}$ is equivalent to $f_{MPC} + \delta_f = \begin{bmatrix} F_1^T & F_2^T & F_3^T & F_4^T \end{bmatrix}^T$ where $\forall k \in 1..4, F_k = G_k\lambda_k$ with $\lambda_k \geq 0$ and G_k the edges of the linearized friction cone of the k-th foot with friction coefficient μ. The final QP problem is thus of the form:

$$\min_{\lambda} \frac{1}{2}\lambda G^T H G \lambda + (G^T g - G^T H\ f_{MPC})^T \lambda \tag{21}$$

$$\text{such that } \forall k \in 1..4,\ \forall i \in 1..4,\ \lambda_{k,i} \geq 0 \tag{22}$$

$$G_k\lambda_k = \begin{bmatrix} \mu & \mu & -\mu & -\mu \\ \mu & -\mu & \mu & -\mu \\ 1 & 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} \lambda_{k,1} \\ \lambda_{k,2} \\ \lambda_{k,3} \\ \lambda_{k,4} \end{bmatrix} \tag{23}$$

$$\lambda = \begin{bmatrix} \lambda_1^T & \lambda_2^T & \lambda_3^T & \lambda_4^T \end{bmatrix}^T \quad G = \begin{bmatrix} G_1 & & & \\ & G_2 & & \\ & & G_3 & \\ & & & G_4 \end{bmatrix} \tag{24}$$

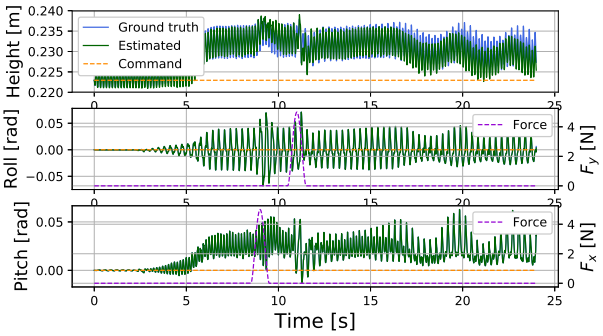
This last problem is a box-QP [23], [24]. Box constraints are easier to handle than generic linear constraints, and lead to simpler and more efficient implementations. In particular, box-QP algorithm are straightforward to implement, do not imply computing the Lagrange multipliers, and have a much better worst-case performance than regular QP (linear versus exponential) [25]. Once δ_f has been determined, $\delta_{\ddot{q}}$ can be deduced. The multi-body dynamics can be written as:

$$\begin{bmatrix} \tau_u \\ \tau_a \end{bmatrix} = M(\ddot{q}_{IK} + \begin{bmatrix} \delta_{\ddot{q}} \\ 0 \end{bmatrix}) + b - J_c^T(f_{MPC} + \delta_f) \tag{25}$$

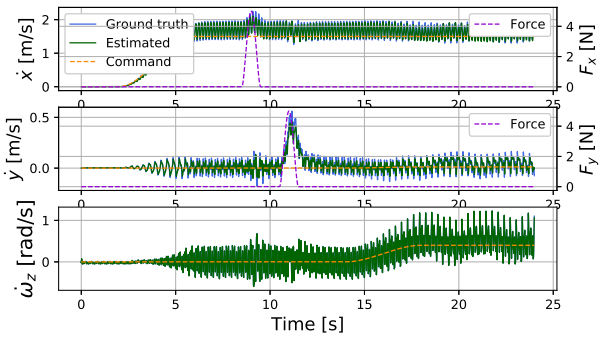
Since only τ_a has to be computed (25) is reduced to:

$$\tau_a = M_u^T(\ddot{q}_{IK,u} + \delta_{\ddot{q}}) + M_a\ddot{q}_{IK,a} + b_a - J_a(f_{MPC} + \delta_f) \tag{26}$$

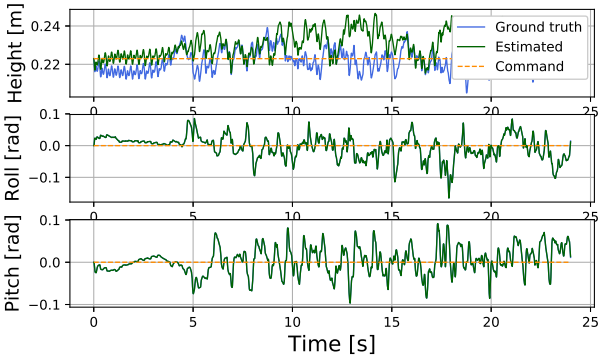
These feedforward torques τ_a are then added to the feedback ones computed by the PD controller. This is beneficial to the locomotion for two reasons. On the one hand, compared to a pure feedforward command, it helps correcting the errors due to the model. On the other hand, compared to a pure feedback command, it enables the use of lower gains.



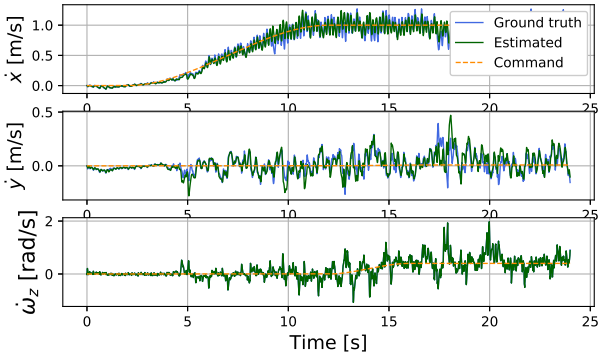
(a) Trunk height and orientation in scenario 1



(b) Trunk velocity in scenario 1



(c) Trunk height and orientation in scenario 2



(d) Trunk velocity in scenario 2

Fig. 4: (a) (b) The quadruped recovers from both perturbations occurring at 9s and 11s and reaches a maximum lateral velocity of +0.56 m/s at t =11.15s. It returns to its nominal behaviour in around 0.5s. (c) (d) The quality of the estimation is worse than in scenario 1 as feet sometimes slip on bumps sides which breaks our immobile contact assumption.

V. EVALUATION IN SIMULATION

A. Simulation setup

A fully-actuated 3D dynamic model of the Solo quadruped was used to assess the effectiveness of the proposed control scheme. The control framework was mainly implemented in Python for ease of use and prototyping. Achieving real time performance was made possible by exploiting NumPy vectorial capabilities, compiling computation intensive parts with Cython (foot trajectory generators) and using libraries that provides Python bindings for their C implementation. The main control loop (footstep planner, foot trajectory generators, whole-body control and state estimator) runs at 500 Hz on a i7-7700 CPU (3.60 GHz) while the MPC runs at 50 Hz in a parallel process and communicates with the main loop through a shared memory. Low-level kinematics and dynamics computation were performed using the Pinocchio library that provides standard rigid body operations and algorithms for poly-articulated systems [22]. For performance reason the MPC was coded in C with Python bindings and exploits the sparcity of its constraint matrices using the OSQP solver [26]. The simulation environment was set up using PyBullet which offers contacts simulation and a Python API to send torques and retrieve relevant data [27].

B. Scenarios

Before testing the proposed control scheme on the real hardware we wanted to assess its stabilization capabilities in a simulated environment for a walking trot gait with period set to $0.32s$. The first scenario consists in a straight walk and a turn on a flat ground with an external perturbation of $+5$ N along X at $t = 9s$ and another one of $+5$ N along Y at $t = 11s$. The second scenario places the robot on a rough terrain: the ground is full of small bumps whose height is random (uniform distribution between 0 and 5 cm). For both scenarios the reference velocity is initially zero and slowly rises up to 1.5 m/s and 1.0 m/s forwards respectively and to 0.4 rad/s when turning.

In scenario 1 the robot reaches its reference forward velocity of 1.5 m/s and returns to its nominal behaviour after both external perturbations. As the robot moves faster it gets increasingly tilted in pitch despite a reference angle at 0^o. This may be due to a compromise with other quantities in the cost function of the MPC which leads to a minima with a non-zero pitch angle in average.

Scenario 2 highlights a limit of the proposed control scheme: the ground is supposed to be flat so feet can slip during stance phase when landing on an unexpected tilted surface such as the sides of bumps. Since the controller expects to work in nominal conditions (flat ground), the contact forces it wants to apply can be out of the friction cone of actual tilted surface. Without knowledge of the environment a possible solution would be to continuously check for slipping during stance phases and react accordingly if such an event is detected.

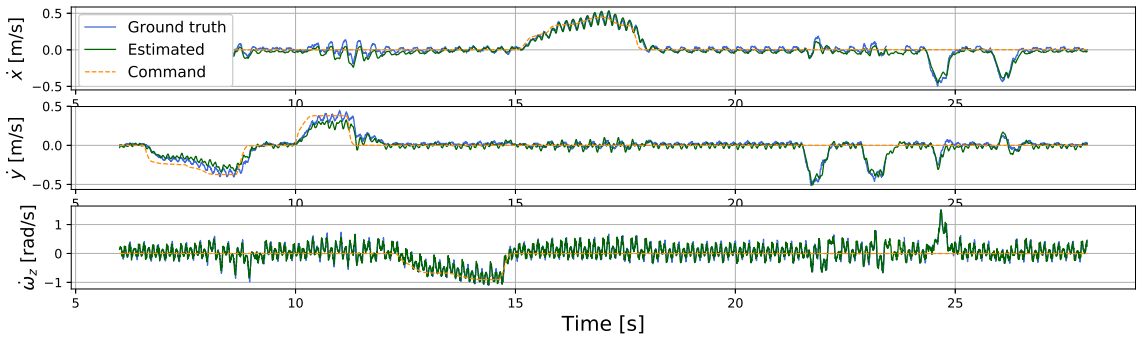


Fig. 5: Reference, estimated and motion-captured velocities of the base obtained on the real hardware

VI. EXPERIMENTAL EVALUATION

A. Experimental setup

Experiments were performed indoors on a flat ground. Small rubber bands have been glued on the robot feet to improve friction with the plastic flooring. To be conservative we use a friction coefficient of 0.9 with the actual coefficient assessed around 1.0. Ground truth was retrieved thanks to a motion capture system consisting of a set of 20 infrared cameras spread around the workspace that track at 200 Hz 9 reflective markers installed on top of the robot base. Cut frequencies f_c^v and f_c^p of the velocity and position complementary filters were set to 3 Hz and 0.4 Hz respectively. The MPC weights chosen for position, orientation, linear velocity and angular velocity errors are respectively $[2.0, 2.0, 20.0, 0.25, 0.25, 10.0, 0.2, 0.2, 0.2, 0.0, 0.0, 0.3]$. They are the same as the ones used for Mini Cheetah and worked out of the box for us. The weights for contact force regularization were set to $1e-4$ for all components. To perform inverse kinematics we used $K_p = 100$ and $K_d = 2\sqrt{K_p} = 10$ for all tasks. In the initial formulation of the QP problem (9) we used 0.1 and 1.0 for the weights of the acceleration and contact force relaxation variables (Q_1 and Q_2 respectively). All joints shared the same feedback control gains $K_p = 6$ and $K_d = 0.2$ for the on-board impedance controller. The performed gait was a trot with a period of $0.32s$. During the experiments the robot was powered via an external power supply. Communications with the robot (sensors data retrieval and command sending) were done using an Ethernet link to the control desktop computer.

B. Results

Fig 5 and 6 present the results of an experiment during which the Solo12 quadruped is controlled by a user with a gamepad (until $t = 18s$). The robot performs first a lateral walk to the right then to the left, a clockwise rotation along the vertical axis and finally a short walk forwards. The robot is then ordered to stay immobile (zero velocity command) while it is being pushed sideways by the user (after $t = 18s$).

Thanks to the motion capture ground truth we can assess both the quality of estimation and of the reference tracking. The quality of the height estimation seems greatly influenced by the variation of others quantities both in orientation and velocity. Estimation of other quantities seems robust to perturbation except during lateral walks at $t = 8s$ and $t = 11s$. This is likely due to the undesired swaying motion in pitch that results from the fact that the assumption that feet do not move in stance phase is not perfectly respected. The velocity reference given by the user is correctly followed when the quadruped does not have to face external perturbations.

In the second half of the experiment the robot manages to recover from the four sideways perturbations it receives at $t = 22s$, $t = 23s$, $t = 24.5s$ and $t = 26.5s$. The third push also transmits a rotating motion to the robot. In all cases it counters the undesired velocity and returns to a nominal behaviour in less than half a second. A video of locomotion and push recovery can be found online[1].

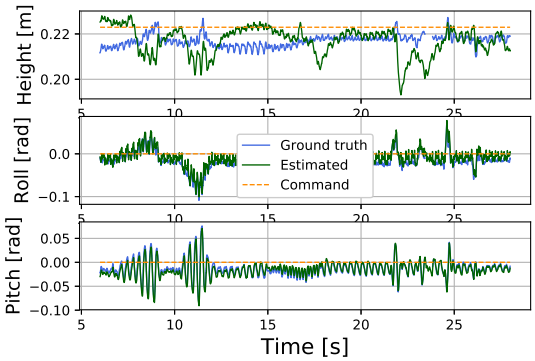


Fig. 6: Reference, estimated and motion-captured height and orientation of the trunk

VII. CONCLUSIONS

In this paper we demonstrated the capability of the Solo-12 quadruped to perform a dynamic and reactive locomotion in order to track a reference velocity. This result shows that, despite its low-complexity and low-cost, this open-access robot constitutes a reliable platform for research and teaching that can be easily maintained and repaired. Future work could focus on the implementation of a more elaborate model predictive control taking into account the non-linear effects that were omitted, considering footsteps placements as part of the optimization problem or even working on non-predefined

[1] https://peertube.laas.fr/videos/watch/1ee81814-3715-4d88-85be-cd3d64cbdfcc

timings for the contact switches. The current implementation could also be optimized to reach higher control frequencies and thus refresh more regularly the feedforward torques and target articular positions and velocities to further improve the control quality. The hardware could also be enhanced alongside the software by embedding batteries and a CPU on-board to turn Solo into a truly autonomous robot that could investigate more dynamic motion without the limitation of a cable. If processing the whole control architecture on-board is not possible then a wireless communication with a control computer could also be considered.

REFERENCES

[1] R. Tajima, D. Honda, and K. Suga, "Fast running experiments involving a humanoid robot," in *2009 IEEE International Conference on Robotics and Automation*. IEEE, 2009, pp. 1571–1576.

[2] C. Hubicki, A. Abate, P. Clary, S. Rezazadeh, M. Jones, A. Peekema, J. Van Why, R. Domres, A. Wu, W. Martin *et al.*, "Walking and running with passive compliance: Lessons from engineering: A live demonstration of the atrias biped," *IEEE Robotics & Automation Magazine*, vol. 25, no. 3, pp. 23–39, 2018.

[3] Y. Gong, R. Hartley, X. Da, A. Hereid, O. Harib, J.-K. Huang, and J. Grizzle, "Feedback control of a cassie bipedal robot: Walking, standing, and riding a segway," in *2019 American Control Conference (ACC)*. IEEE, 2019, pp. 4559–4566.

[4] B. Dynamics. Spot autonomous navigation. Accessed: 2020-07-28. [Online]. Available: $https://www.youtube.com/watch?v=Ve9kWX_KXus$

[5] U. Robotics. Quadruped robot a1 walks with you to the future. Accessed: 2020-07-28. [Online]. Available: $https://www.youtube.com/watch?v=2H3dzZEi-qw$

[6] D. Kim, J. Di Carlo, B. Katz, G. Bledt, and S. Kim, "Highly dynamic quadruped locomotion via whole-body impulse control and model predictive control," *arXiv preprint arXiv:1909.06586*, 2019.

[7] C. D. Bellicoso, F. Jenelten, C. Gehring, and M. Hutter, "Dynamic locomotion through online nonlinear motion optimization for quadrupedal robots," *IEEE Robotics and Automation Letters*, vol. 3, no. 3, pp. 2261–2268, 2018.

[8] M. Hutter, C. Gehring, D. Jud, A. Lauber, C. D. Bellicoso, V. Tsounis, J. Hwangbo, K. Bodie, P. Fankhauser, M. Bloesch *et al.*, "Anymal-a highly mobile and dynamic quadrupedal robot," in *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2016, pp. 38–44.

[9] 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," *Proceedings of the Institution of Mechanical Engineers, Part I: Journal of Systems and Control Engineering*, vol. 225, no. 6, pp. 831–849, 2011.

[10] A. W. 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 *2015 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2015, pp. 5148–5154.

[11] B. Katz, J. Di Carlo, and S. Kim, "Mini cheetah: A platform for pushing the limits of dynamic quadruped control," in *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, 2019, pp. 6295–6301.

[12] Z. Xie, P. Clary, J. Dao, P. Morais, J. Hurst, and M. van de Panne, "Iterative reinforcement learning based design of dynamic locomotion skills for cassie," *arXiv preprint arXiv:1903.09537*, 2019.

[13] D. Jain, A. Iscen, and K. Caluwaerts, "Hierarchical reinforcement learning for quadruped locomotion," *arXiv preprint arXiv:1905.08926*, 2019.

[14] B. Dynamics. Spot shopping page. Accessed: 2020-07-29. [Online]. Available: $https://shop.bostondynamics.com/spot$

[15] R. IEEE. Laikago specs page. Accessed: 2020-07-29. [Online]. Available: $https://robots.ieee.org/robots/laikago/$

[16] Open dynamic robot initiative. Accessed: 2020-07-29. [Online]. Available: $https://open-dynamic-robot-initiative.github.io/$

[17] F. Grimminger, A. Meduri, M. Khadiv, J. Viereck, M. Wüthrich, M. Naveau, V. Berenz, S. Heim, F. Widmaier, T. Flayols, J. Fiene, A. Badri-Spröwitz, and L. Righetti, "An open torque-controlled modular robot architecture for legged locomotion research," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 3650–3657, 2020.

[18] Quadruped reactive walking controller. Accessed: 2020-12-07. [Online]. Available: https://github.com/Gepetto/quadruped-reactive-walking

[19] P. M. Wensing, A. Wang, S. Seok, D. Otten, J. Lang, and S. Kim, "Proprioceptive actuator design in the mit cheetah: Impact mitigation and high-bandwidth physical interaction for dynamic legged robots," *IEEE Transactions on Robotics*, vol. 33, no. 3, pp. 509–522, 2017.

[20] T. Flayols, A. Del Prete, P. Wensing, A. Mifsud, M. Benallegue, and O. Stasse, "Experimental evaluation of simple estimators for humanoid robots," in *2017 IEEE-RAS 17th International Conference on Humanoid Robotics (Humanoids)*, 2017, pp. 889–895.

[21] J. Carpentier, M. Benallegue, N. Mansard, and J.-P. Laumond, "A kinematics-dynamics based estimator of the center of mass position for anthropomorphic system—a complementary filtering approach," in *2015 IEEE-RAS 15th International Conference on Humanoid Robots (Humanoids)*. IEEE, 2015, pp. 1121–1126.

[22] J. Carpentier, F. Valenza, N. Mansard *et al.*, "Pinocchio: fast forward and inverse dynamics for poly-articulated systems," https://stack-of-tasks.github.io/pinocchio, 2015–2019.

[23] Z. Dostál, "Box constrained quadratic programming with proportioning and projections," *SIAM Journal on Optimization*, vol. 7, no. 3, pp. 871–887, 1997.

[24] Y. Ye, "Approximating quadratic programming with bound constraints," *Mathematical programming*, vol. 84, pp. 219–226, 1997.

[25] J. Nocedal and S. Wright, *Numerical optimization*. Springer Science & Business Media, 2006.

[26] B. Stellato, G. Banjac, P. Goulart, A. Bemporad, and S. Boyd, "OSQP: an operator splitting solver for quadratic programs," *Mathematical Programming Computation*, 2020. [Online]. Available: https://doi.org/10.1007/s12532-020-00179-2

[27] E. Coumans and Y. Bai, "Pybullet, a python module for physics simulation for games, robotics and machine learning," http://pybullet.org, 2016–2020.