Real time footstep planning and control of the Solo quadruped robot in 3D environments

Fanny Risbourg, Thomas Corbères, Pierre-Alexandre Léziart, Thomas Flayols, Nicolas Mansard, Steve Tonneau

Real time footstep planning and control of the Solo quadruped robot in 3D environments

Fanny Risbourg[1*], Thomas Corbères[2*], Pierre-Alexandre Léziart[1],
Thomas Flayols[1], Nicolas Mansard[1,3], Steve Tonneau[2]

***Abstract*— Quadruped robots have proved their robustness to cross complex terrain despite little environment knowledge. Yet advanced locomotion controllers are expected to take advantage of exteroceptive information. This paper presents a complete method to plan and control the locomotion of quadruped robots when 3D information about the surrounding obstacles is available, based on several stages of decision. We first propose a contact planner formulated as a mixed-integer program, optimized on-line at each new robot step. It selects a surface from a set of convex surfaces describing the environment for the next footsteps while ensuring kinematic constraints. We then propose to optimize the exact contact location and the feet trajectories at control frequency to avoid obstacles, thanks to an efficient formulation of quadratic programs optimizing Bezier curves. By relying on the locomotion controller of our quadruped robot Solo, we finally implement the complete method, provided as an open-source package. Its efficiency is asserted by statistical evaluation of the importance of each component in simulation, while the overall performances are demonstrated on various scenarios with the real robot.**

I. Introduction

Legged locomotion is primarily a robust control issue. "Blind" controllers (those not relying on visual inputs) enable robots to climb stairs or navigate uneven terrains based on the assumption that the floor is flat and therefore adapting reactively to perturbations [1], [2]. However, blind controls are insufficient in circumstances when the environment contains holes or obstacles too high to climb.

Complex environments require the planning of a robot's motion several steps ahead (within an established horizon) [3]. Unfortunately, computing the motion of any upcoming footsteps prior to the completion of the current step sets tight computational time constraints [4], [5]. While simultaneously planning the motion and footstep locations requires tackling the combinatorics associated with the discrete choice of contacts, a problem of exponential complexity [6], thus motivating the use of simplifying assumptions. A popular choice is to solely consider the dynamics of a system's center of mass [7]–[11], which does not guarantee the feasibility of the planned motion. To further simplify the problem, one can relax the discrete contact dynamics into a continuous formulation. Regardless, these approaches remain non-linear and computationally demanding. The proposed alternative that we choose is to plan for the footstep locations before computing the motion [12]–[16]. The additional decoupling between contact planning and motion generation challenges further the feasibility of the plan. In addition, the planning frequency (10Hz) remains incompatible with reactive adaptations to external perturbations or state estimation errors. The recent successes of machine learning techniques suggest that these issues could be mitigated [17], [18], but in the meantime, there is a need for the robot controller to locally adapt the plan, but "not too much" as we know that deviation from that plan can lead to the failures that justified the planning in the first place.

What does "not too much" mean? Our research question is to find the compromise between strictly following a plan and adapting it reactively.

Assuming that the world is a collection of convex potential contact surfaces, we hypothesise that the key information given by the contact planner is the choice of the next contact surfaces to step on. We thus consider the selected surfaces as a hard constraint that cannot be challenged before a new plan is computed. This hypothesis leads to two issues:

a) the exact position of the next contacts should be adapted in real-time at the control level to allow handling collision avoidance, state estimation errors and other perturbations.

b) the contact planner should be able to predict the future contact position, so that the output plan is robust to the resulting misalignment.

We propose a complete implementation of this idea as a 3-stage locomotion framework able to navigate complex 3D terrains. The first stage consists of a contact planner, formulated as a Mixed-Integer Program (MIP) [12], [16]. To enforce the planner decisions are aligned with the behaviour of the controller (problem b), the contact plan minimises a cost adapted from the Raibert heuristic [19]. The second stage, formulated as a Quadratic Program (QP) under collision-avoidance constraints (problem a), optimizes footstep locations and end-effector trajectories at (50Hz) to account for local disturbances. The third stage is adapted from the locomotion controller of Solo [20], which tracks a reference centroidal movement by combining a low-dimensional model-predictive controller (MPC) with a whole-body controller (WBC) to generate the low-level commands that synthesise the motion.

Our method hypothesis is empirically validated with the 12 degrees of freedom quadruped robot Solo [20], [21], in scenarios where the robot has to follow a reference velocity sent in real time by a user with a joystick. The next 6 steps of the robot are planned every 160 ms, while the robot whole-body control is computed at 1 kHz. Repeated tests

*Joint first authors
[1]LAAS-CNRS, Université de Toulouse, CNRS, Toulouse, France
[2]School of Informatics, University of Edinburgh, UK
[3]Artificial and Natural Intelligence Toulouse Institute, France

individually demonstrate the contribution of each aspect of the framework (contact planning, local footstep re-planning, collision avoidance) to the robustness of the computed motions.

Our contributions are thus a complete open-source framework for the real time control of a legged robot and an exhaustive quantitative analysis of our framework validating our research hypothesis. In the remainder of this paper we first present the two key components of our method (Section II): the mixed-integer program optimising the Raibert heuristic to compute the contact plan, and the efficient QP for avoiding effector collisions. We then present the open-source implementation of the complete architecture in Section III-C. The experimental setup is presented in Section IV and the results are analysed in Section V.

II. Reactive planning in complex environments

Given the current state of the robot, a 3D (X-Y translation, Yaw rotation) velocity reference for its base and a decomposition of the environment as a set of convex contact surfaces (Fig. 1), our framework first selects the surfaces where a contact will occur over a defined horizon. Contact positions are then computed for the next steps in the horizon, as well as the trajectory that brings the currently moving effector to the contact. The contact surfaces are selected at a frequency of approximately 10 Hz, while the effector trajectory and contact positions are updated at 1 kHz.

Both the surface selection and contact positioning modules use the Raibert heuristic [19]. After giving the definitions and notations used throughout the paper, we thus first recall the definition of the Raibert heuristic. We then present our contact planner and conclude this section with a description of the foot position adaptation and collision avoidance modules.

A. Definitions and notations

The Robot state is formally described by the:

- Center Of Mass (COM) position, velocity and acceleration $\mathbf{c}$, $\dot{\mathbf{c}}$ and $\ddot{\mathbf{c}}$, each in $\mathbb{R}^3$;
- base transformation matrix in the world frame;
- 3D position of each end-effector in the world frame;
- gait, i.e. the list of effectors currently in contact, as well as the contacts to be activated and deactivated over the planning horizon.

The horizon n is defined as the number of future contact creations that are considered. In the case of the trotting gait that we use by default, a horizon $n = 4$ describes two steps, as at each step two contacts are created simultaneously.

The environment is the union of m disjoint quasi-flat[1] contact surfaces $\mathscr{S} = \bigcup_{j=1}^{m} \mathscr{S}^j$. Each set $\mathscr{S}^j$ is a polygon embedded in a 3D plane (Fig. 1):

$$\mathscr{S}^j := \{\mathbf{p} \in \mathbb{R}^3 | \mathbf{S}^j \mathbf{p} \leq \mathbf{s}^j\}. \quad (1)$$

$\mathbf{S}^j \in \mathbb{R}^{h\times 3}$ and $\mathbf{s}^j \in \mathbb{R}^h$ are respectively a constant matrix and a vector defining the h half-spaces bounding the surface.

[1]a quasi-flat contact surface is such that its associated friction cone contains the gravity vector

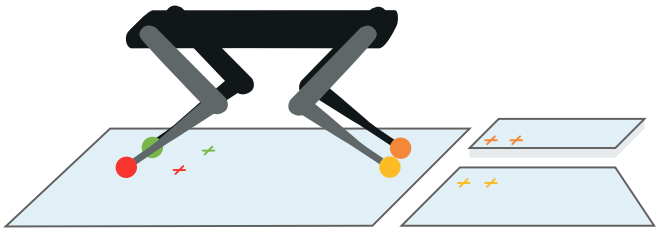

Fig. 1. Contact planning on convex surfaces

The contact plan is given as a list of contact surfaces $\mathscr{S}_k^i \in \mathbb{R}^3, 1 \leq i \leq l$, with l the total number of end-effectors and k describing the k-th contact phase.

B. The Raibert heuristic

Assuming that the robot is moving on flat ground, the 2D contact position $\mathbf{p}(\mathbf{v}_{ref})$ for a moving-end effector is defined using the Raibert heuristic:

$$\mathbf{p}(\mathbf{v}_{ref}) = \begin{bmatrix} p_{sh,x}(\psi_k) + \frac{t_{st}}{2} v_x + k(v_x - v_{ref,x}) + \frac{1}{2}\sqrt{\frac{h}{\mathbf{g}}} v_x \times \omega_{ref} \\ p_{sh,y}(\psi_k) + \frac{t_{st}}{2} v_y + k(v_y - v_{ref,y}) + \frac{1}{2}\sqrt{\frac{h}{\mathbf{g}}} v_y \times \omega_{ref} \end{bmatrix}$$

where $\mathbf{p}_{sh} = \begin{bmatrix} p_{sh,x} & p_{sh,y} \end{bmatrix}^T$ is the predicted position of the shoulder, k is a user-defined feedback coefficient, $\mathbf{v} = \begin{bmatrix} v_x & v_y \end{bmatrix}^T$ the current velocity of the robot base, $\mathbf{v}_{ref} = \begin{bmatrix} v_{ref,x} & v_{ref,y} \end{bmatrix}^T$ the linear reference velocity, ω_{ref} the angular reference velocity around z, h the height of the robot, $\mathbf{g}$ the gravity and t_{st} the period of the stance phase.

C. Contact surface selection

The contact planner solves the combinatorial problem of selecting the contact surfaces relevant for the motion while enforcing the kinematic constraints for a reduced model of the robot. This problem can be formulated as a Mixed-Integer Program (MIP) [12], solved using a variation of the SL1M open-source framework [15], [16], with the difference that quasi-static balance is not verified in our implementation.

Planner inputs and outputs: The inputs are:

- The current state of the robot;
- A 3D velocity reference for base (X/Y translation, yaw);
- A list of predicted positions/orientations for the base over the horizon, at the time of contact creation;
- A list of potential contact surfaces to step on for each of the contacts created in the horizon

The computation of these inputs is detailed in Section III. The planner outputs the next contact positions to be reached by the robot and the list of selected contact surfaces over the horizon. In our framework, following our research hypothesis we only consider the selected contact surfaces.

MIP formulation of the problem: We briefly describe the used MIP and refer the reader to [15] for further details.

We define the set of n variables $\mathbf{a}_i = [a_i^1, \cdots, a_i^m] \in \{0,1\}^m, 1 \leq i \leq n$, such that $a_i^j = 0$ implies that the i-th footstep position $\mathbf{p}_i$ in the plan belongs to the j-th contact

surface and $a_i^j = 1$ implies that the i-th footstep is not in contact with the j-th contact surface[2].

We solve the following MIP:

$$\begin{aligned} \textbf{find} \quad & \mathbf{P} = [\mathbf{p}_1, \cdots, \mathbf{p}_n] \in \mathbb{R}^{3\times n} \\ & \mathbf{A} = [\mathbf{a}_1, \cdots, \mathbf{a}_n] \in \{0,1\}^{n\times m} \\ \textbf{min} \quad & l(\mathbf{P}) \\ \textbf{s.t.} \quad & \mathbf{P} \in \mathscr{I} \cap \mathscr{F} \\ & \forall i, 1 \leq i \leq n : \\ & \textbf{card}(\mathbf{a}_i) = m-1 \\ & \quad \forall j, 1 \leq j \leq m : \\ & \quad\quad \mathbf{S}_j \mathbf{p}_i \leq \mathbf{s}_j + M a_i^j \mathbf{1}. \end{aligned} \tag{2}$$

where l is a quadratic objective function to be minimised, $M \in \mathbb{R}^n$ is a sufficiently large number and **card** is the cardinality of a vector, that is the number of entries different from 0. The constraint $\textbf{card}(\mathbf{a}_i) = m-1$ guarantees that at each step the position planned lies exactly on one contact surface. The $\mathbf{a}_i$ variables indicate the contact surfaces that have been selected by the planner.

Additionally $\mathscr{I}$ is a user-defined set of initial constraints and $\mathscr{F}$ is a set of feasibility constraints. These sets are assumed to be convex. In this work the initial constraints impose that non moving effector positions match the current state after one step. The feasibility constraints guarantee that the relative distance between each effector is linearly constrained to approximate the kinematic constraints [22].

Raibert heuristic as a cost function: We formulate the cost function l as the minimisation of the distance between the 2D foot position and a desired 2D position:

$$l = \|\mathbf{p}_\mathbf{i}^* - \mathbf{p}_{\mathbf{i},\mathbf{xy}}\|^2$$

The desired position is the following approximation of the Raibert heuristic:

$$\mathbf{p}_\mathbf{i}^*(\mathbf{v}_{ref}) = \begin{bmatrix} p_{sh,i,x}(\psi_k) + \frac{t_{st}}{2} v_{ref,x} \\ p_{sh,i,y}(\psi_k) + \frac{t_{st}}{2} v_{ref,y} \end{bmatrix} \tag{3}$$

Using the notations presented in Section II-B

An additional cost function is used with a weight ten times lower to limit the distance between the foot and the shoulder:

$$l_{sh} = \|\mathbf{p}_{\mathbf{sh},\mathbf{i}} - \mathbf{p}_\mathbf{i}\|^2$$

D. Real-time Footstep position adaptation

Given the target contact surface and the current state of the robot, we compute the next contact locations $\mathbf{p}_i, 1 \leq i \leq n$ for the moving effectors over a horizon of n[3]. We assume that the optimal contact positions are given by the Raibert heuristic, which we extend to 3 dimensions. We formulate a QP problem to satisfy at best the surface constraints while minimising the violation of the heuristic in the least-square sense. The decision variables are both the velocity of the base and the contact positions. Each position variable is defined as an offset $\alpha_i = [\alpha_{i,x}, \alpha_{i,y}, \alpha_{i,z}]^T \in \mathbb{R}^3$ with respect to the position predicted by the Raibert heuristic:

$$\mathbf{p}_i = \begin{bmatrix} p_x(v_{ref,x}) + \alpha_{i,x} \\ p_y(v_{ref,y}) + \alpha_{i,y} \\ \alpha_{i,z} \end{bmatrix}$$

Likewise the reference velocity $\mathbf{v}_{ref}^+$ is updated with an offset $\beta = [\beta i,x, \beta i,y]^T \in \mathbb{R}^2$ from the initial reference $\mathbf{v}_{ref}$:

$$\mathbf{v}_{ref}^+ = \begin{bmatrix} \beta_x + v_{ref,x}^* \\ \beta_y + v_{ref,y}^* \end{bmatrix}$$

The resulting QP is written:

$$\begin{aligned} \min \quad & \frac{1}{2}(w_1 \sum_{i=1}^{n} \|\alpha_i\|^2 + w_2 \|\beta\|^2) \\ \text{s.t.} \quad & \mathbf{S}_i(\mathbf{p}_i + \alpha_i) \leq \mathbf{s}_i, \qquad \forall 1 \leq i \leq n \end{aligned}$$

where a surface $\mathscr{S}_i$ is the surface selected as the i-th contact (Section II-A), and w_1 and w_2 are user-defined weights respectively set to 1 and 1000.

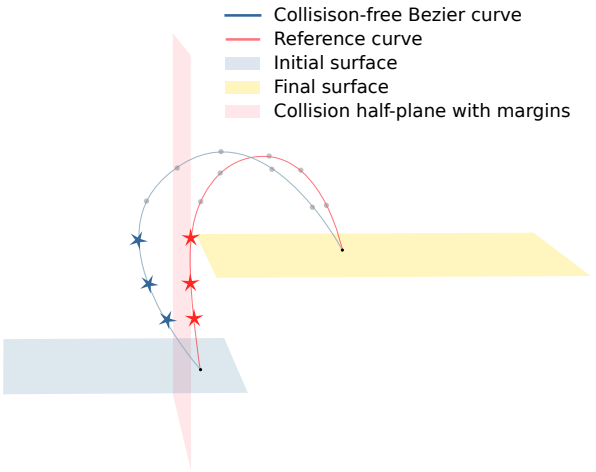

Fig. 2. End-effector trajectory adaptation for collision avoidance

E. End-effector trajectory optimisation

The trajectory of a moving end-effector $\mathbf{p}(t) : \mathbb{R} \mapsto \mathbb{R}^3$ is computed as a compromise between a uniquely defined reference $\mathbf{p}_{ref}(t)$ and the adjustment required to avoid collisions with the environment (Fig. 2). The trajectories are 3D Bézier curves of degree d defined in the Bernstein basis:

$$\mathbf{p}(t) = \sum_{i=0}^{d} B_i^d(\frac{t}{T})\mathbf{x}_i \tag{5}$$

where $\mathbf{x} = \begin{bmatrix} \mathbf{x}_0 & \dots & \mathbf{x}_d \end{bmatrix} \in \mathbb{R}^{3d}$ are the $d+1$ control points of the curve and T is the total duration of the trajectory.

1) Tracking a reference trajectory: The reference end-effector trajectory $\mathbf{p}_{ref}(t)$ is decoupled into a 2D X and Y trajectory and a Z trajectory. A 2D curve of degree 5 is used for computing the X and Y reference trajectory. To satisfy continuity constraints, the initial position, velocity and acceleration of the curve are constrained to match the

[2]To avoid the multiplication of variables indices in the problem definition, we assume that the set of potential contact surfaces is the same over the horizon. This is not a limitation of our approach.

[3]Not necessarily equal to the contact planning horizon

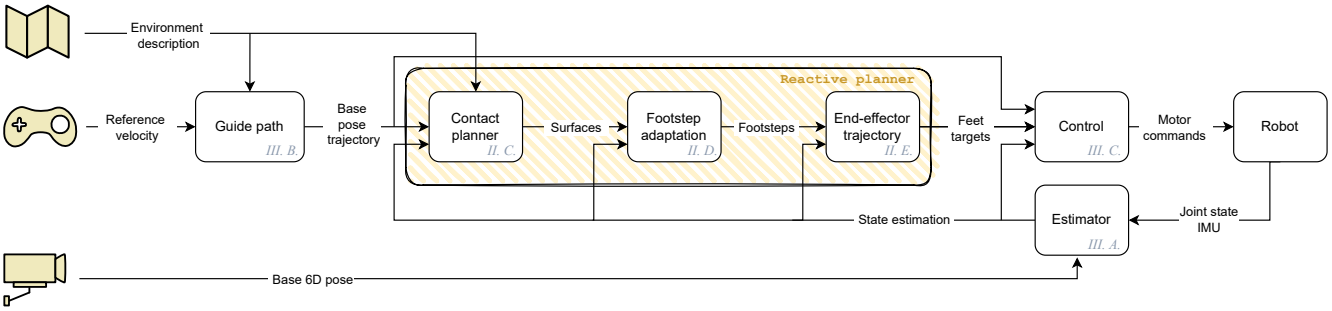

Fig. 3. Solo12 walking controller architecture. The contribution of this paper lies in the integration of a reactive planning module in the controller

current state of the effector. We also constrain the terminal position to match the contact location with a null velocity and acceleration. These 6 constraints thus uniquely define $\mathbf{p}_{ref}(t)$ in X and Y. The reference trajectory in Z is computed similarly but is of degree 6 to allow the user to define an apex[4] for the trajectory at $T/2$.

The adjusted end-effector trajectory $\mathbf{p}(t)$ is a degree 7 curve (we keep a low dimension to avoid over-fitting). $\mathbf{p}(t)$ and $\mathbf{p}_{ref}(t)$ are discretised over T into n_T+1 samples $0 \leq t_k \leq T_k$. By definition of a Bezier curve $\mathbf{p}(t_k)$ is linearly defined by the control points $\mathbf{x}$, such that $\forall k, \exists \mathbf{A}_k, \mathbf{p}(t_k) = \mathbf{A}_k\mathbf{x}$, with $\mathbf{A}_k$ constant.

2) Collision avoidance constraints: For each contact surface(s) active during the motion Fig. 2 and for each sample $\mathbf{p}_{ref}(t_k)$, we check whether the segment connecting the sample to its neighbours is in collision. If this is the case, a collision constraint defined by the traversed half-space (Fig. 2-Pink) is created for $\mathbf{p}(t_k)$ (using a small margin for safety). All the collected constraints are then stacked into a single matrix and vector $\mathbf{G}$ and $\mathbf{g}$ that linearly constrain the control points of $\mathbf{x}$, leading to the QP:

$$\begin{aligned} \min_{\mathbf{x}} \quad & \frac{1}{2}\sum_{k=0}^{n_T} \|\mathbf{p}(t_k) - \mathbf{p}_{ref}(t_k)\|^2 \\ \text{s.t.} \quad & \mathbf{p}(0) = \mathbf{p}_{ref}(0), \dot{\mathbf{p}}(0) = \dot{\mathbf{p}}_{ref}(0) \\ & \ddot{\mathbf{p}}(0) = \ddot{\mathbf{p}}_{ref}(0) \\ & \mathbf{p}(T) = \mathbf{p}_{ref}(T), \dot{\mathbf{p}}(T) = \ddot{\mathbf{p}}(T) = \mathbf{0} \\ & \mathbf{G}\mathbf{x} \leq \mathbf{h} \end{aligned}$$

III. FRAMEWORK ARCHITECTURE

Our framework (Fig. 3) is generic but designed around Solo [23]. It continuously takes as input a desired reference velocity for the geometric root of the robot, given with a joystick. We do not consider visual perception in this work.

The state estimator (Section III-A) combines data obtained from the Inertial Measurement Unit (IMU) and motion capture (MoCap) to estimate the state of the robot.

The guide path module (Section III-B) computes a trajectory for the base, used as input for the reactive planner.

[4]In our experiments the apex is defined as the difference between the initial contact surface and the target contact surface + 5 cm

The reactive planner is the main addition to our previous framework [20]. It uses the base trajectory and reference velocity to compute a contact plan, adapt the next footstep locations and compute the flying feet trajectories to avoid collisions with the environment (Section II).

The control module (Section III-C) consists of a centroidal MPC and a WBC in charge of computing the required contact forces and the joint torque required to achieve them, as well as a low-level controller that convert the torques targets into motor commands. The software implemented the proposed method will be made available under open license (BSD3) on the project page [24] upon acceptance of the paper.

A. State estimation

The IMU used on the Solo quadruped includes an extended Kalman filter that estimates the body angular velocity, orientation and acceleration without gravity. With our setup, position and yaw orientation in the world are not observable quantities and would drift if we were to assess them by integration. For those, we use a MoCap system. Estimates of the linear velocity $\mathbf{v} \in \mathbb{R}^3$ comes from a decoupled linear approach [25] that combines MoCap and IMU data with a complementary filter [26]. To update the velocity estimate at time $t+1$ the accurate acceleration of the IMU $\dot{\mathbf{v}}_{IMU}$ is integrated and slowly dedrifted by MoCap velocity measurements $\mathbf{v}_{MoCap}$:

$$\mathbf{v}^{t+1} = \alpha(\mathbf{v}^t + \Delta t\ \dot{\mathbf{v}}^t_{IMU}) + (1-\alpha)\mathbf{v}^t_{MoCap} \tag{6}$$

with α a weight close to 1 (0.97 in our case).

B. Base guide trajectory computation

The guide path module computes the trajectory in translation and rotation for the base of the robot on the horizon. The estimated current state is the initial state of the trajectory. The reference velocities ($v_{ref,x}$, $v_{ref,y}$, ω_{ref}) are integrated to get a (x, y, yaw) trajectory starting from the current state.

The height of the base and the roll and pitch angles are estimated from the environment, which is represented as a height-map in this module. We assume that the robot orientation can be defined relatively to the average height of the closest points (less than 20cm away from the current x/y position) in the height-map. A 3D plane is then fitted with the resulting points in the least-square sense. The roll and pitch values of the fitted plan are use for the orientation of the

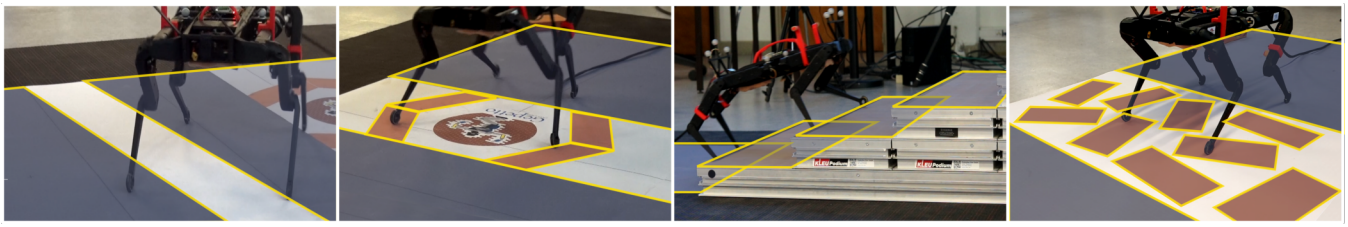

Fig. 4. Some scenarios built for the experiments with the convex surfaces highlighted. From the left to the right: straight hole, bridge, stairs and stepping stones scenarios

base. The x/y position of the base, projected on this plane, gives the z position of the base (offset by a constant). The vertical velocity is obtained by differentiating the z positions obtained through the integration of the x and y quantities, while w_{roll} and are set to 0.

C. Control

In this section we briefly describe the components of the lower level controllers in our architecture, but refer to [20] for additional implementation details.

1) Centroidal MPC: We assume a constant offset between the center of the base and the COM position in the initial configuration[5] and formulate a convex QP to track the base guide trajectory while satisfying the dynamics constraints. This MPC outputs contact forces that should be applied by feet in stance phase and runs at 50 Hz with a 0.32s horizon using the OSQP solver [27].

2) Whole-Body control: The WBC runs at 1 kHz. It includes a task-based Inverse Kinematics (IK) that uses a full model of the quadruped [28] to compute desired joint positions, joint velocities and base accelerations to follow the centroidal guide path and the feet trajectories from the collision QP, (Section II-E). Then, a QP solver outputs feedforward torques after making a trade-off between contact forces provided by the MPC and base accelerations of the IK, with the under-actuated part of the dynamics as a constraint. Here we use OSQP as well.

3) Low-Level control: The low-level controller consists of a Proportional-Derivative (PD) plus feedforward controller. The target torques τ sent to the actuators are a combination of a feedback and feed-forward terms using the target joint positions $\mathbf{q}^*$, joint velocities $\dot{\mathbf{q}}^*$ and feed-forward torques τ^* computed by the WBC, with proportional and derivative gains K_p and K_d. These torques are directly computed by motor control boards at 10 kHz based on measured joint positions $\mathbf{q}$ and velocities $\dot{\mathbf{q}}$:

$$\tau = K_p(\mathbf{q}^* - \mathbf{q}) + K_d(\dot{\mathbf{q}}^* - \dot{\mathbf{q}}) + \tau^* \tag{7}$$

IV. Experimental setup

We tested our research hypothesis and framework on the Solo quadruped [23]. In simulation and during experiments, the robot has been challenged with complex environments. The controller proved capable of progressing on terrains with holes, bridges, stepping stones or stairs. Several scenarios were designed to demonstrate the robustness of the controller.

[5]A reasonable assumption for Solo as most of the mass is in the trunk

A. The Solo quadruped

The quadruped robot Solo [23] has been developed by the Open Dynamic Robot Initiative. Solo is a relatively low cost and easy to repair robot composed mainly of 3D-printed and off-the-shelves components. It is small (approximately 22 cm of height) and light (2.5 kg). It has 12 degrees of freedom actuated by brushless outrunner motors adapted to torque control. Encoders are located at each joint and an IMU embedding an extended Kalman filter is used for attitude estimation. The control loop runs on a distant computer and motor commands are sent at 1kHz through an Ethernet cable.

To perform tests in simulation, the physics engine PyBullet [29] was used with a model of the robot. The base pose and velocity usually obtained with the motion capture were replaced by the ones given by the simulator.

B. How to evaluate the controller

A qualitative evaluation of the framework was first performed both in simulation and in real life experiments. More specific results were then obtained in simulation to highlight the use of a module or another and evaluate the influence of a given parameter. Specific environments were designed in simulation and built for real life experiments when possible:

- **Flat terrain** used to assess the general robustness of the controller to perturbations.
- **Straight hole** on the ground up to 15 cm of width. It shows the ability to avoid a forbidden area while maintaining balance for any angle of approach.
- **Bridge**-like structures, two non-parallel thin (10 cm) walkways where a precise location of the feet is required and relative foot positions are challenging to set.
- **Stairs**, straight or spiral, with several heights and width. Holes between the steps were added, as well as irregularities in the height of the steps. They were used to show the usefulness of the collision avoidance module.
- **Stepping stones** to illustrate the necessity of the contact plan and test if the footstep plan is adapted fast enough to adapt to quick changes in the reference velocity. In real-life experiments, the stepping stone were rectangles of 22 cm by 11 cm and were either flat, or with a height of 6 cm. They were placed randomly and could be added or removed at will to add more or less difficulty.

Pictures of some scenarios realized for real-life experiments are presented in Fig. 4

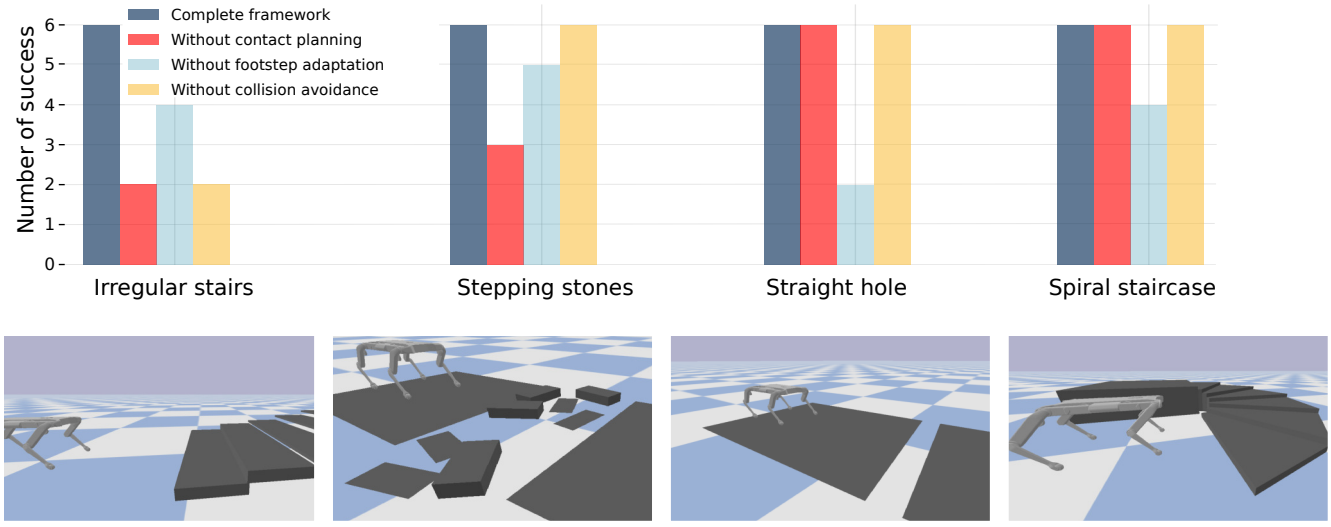

Fig. 5. Repeated tests with a modified controller to evaluate the benefit of each module. For each scenarios, six velocity profiles, summarizing the range of robot capabilities, were run in simulation and the results of each test is reported here.

C. Qualitative results

All scenarios were tried successfully in simulation, with different reference velocities and initial positions of the robot. In real-life, the robot managed to recover from strong perturbations on flat ground, could cross a hole materialized by a line drawn on the ground and turn over it. It walked on a drawn bridge which required to move its left feet away from his right one. On stepping stones the robot managed to place correctly its feet and maintain its balance even with quick variations in the reference velocity, for example deciding to stop or to turn. The stairs were climbed with varying approach angles and the robot could turn over them without falling. The companion video highlights some of these experiments. An extended video is available on the project page [24].

V. Results and Discussion

A. Influence of each module on the framework robustness

To evaluate the contribution of each component to the robustness of the framework we instantiated alternative frameworks where one module had been deactivated. The frameworks were then compared with each other in simulation, on scenarios designed to highlight the edge cases (Fig. 5). Four different scenarios were selected, and for each of them, six different velocity profiles were applied.

1) Contact planner utility: We replaced the contact planner with a naive heuristic where the selected surface is the closest to the position predicted 2D Raibert's heuristic (the planning is thus restricted to a horizon $n = 1$). In the straight hole and spiral staircase scenarios, the contact planner is effectively replaced by the heuristic without any incidence on the success rates (Fig. 5). The benefit of contact planning is highlighted on the irregular stairs and stepping stones scenarios. The failure cases occur when the projection of the 2D heuristic is on a surface that cannot be reached from the current state of the robot, while this surface choice rejected by the contact planner, justifying the need to test several candidate surfaces (and thus to use our planner).

2) The necessity of the footstep adaptation: On flat ground, planning the footstep locations at a high frequency (1 kHz) has proven useful to keep the robot stable and react quickly to disturbances [20], [30]. To evaluate the importance of the footstep adaptation module, we tried the framework using directly the footstep positions computed by the contact planner. Since the contact plan is only computed once per step, the footstep positions were then fixed throughout the whole step. Fig. 5 illustrates that in most cases not adapting the footstep location leads to failures of the framework.

We also computed the distance between the contact positions predicted by the contact planner and the final contact positions reached on four different scenarios. Table I shows that the average distance is small (< 5 mm), which confirms that the predicted position is relevant in the nominal case. The maximum distance measure varies from 1 to 4 cm in the stepping stones scenario, the latter being in the same order of magnitude than the footstep distance. This illustrates further the importance of adapting reactively the footstep position to maintain balance.

TABLE I
Distance between planning contact position and adapted position

Scenario	Mean distance (m)	Maximum distance (m)
Flat	0.0018	0.012
Hole	0.0019	0.026
Bridge	0.0021	0.025
Stepping stones	0.0037	0.043

3) Collision avoidance is needed for irregular steps: Surprisingly, disabling collision avoidance does not result in an increased failure rate in three of the considered environments (Fig. 5). The irregular stairs environment is characterised by overlapping steps with gaps between them. The foot can

then get stuck in-between two steps without the proper end-effector trajectory. With the collision avoidance module, the effector trajectory avoids the edges of the stairs and goes backwards at first, as shown in Fig. 2 and in the video, resulting in a better behaviour. This scenario highlights the need for collision avoidance in such cases.

B. Effect of the surface planner horizon length

The contact planner horizon length n is intuitively a crucial parameter. However in all the scenarios listed in section IV-B, after having defined a goal position and compared the success rate obtained with different values of n, no significant difference was observed. Planning with a two-step horizon seems sufficient to avoid the failure cases observed without the planner in section V-A.1. As expected the planning time increases with the horizon, and a horizon $n = 8$ (4 steps) was too expensive for real-time computation on scenarios with many potential surface such as the stepping stones scenarios.

C. Computation time of each module

During the experiments on the robot, we measured the computation time of each module to make sure they were compliant with the real-time requirements. The results are presented in table II. The contact planner maximal computation time of 137 ms is lower than 160 ms, the duration of a step. The two QPs of the footstep adaptation and collision avoidance modules are solved fast enough to be computed at 1 kHz in the main control loop.

TABLE II
Computation times of the different modules of the controller

Module	Average time (ms)	Maximal time (ms)
Guide path	0.122	0.238
Contact planner	58.4	137.4
Footstep adaptation	0.0270	0.0825
Collision avoidance	0.099	0.656
Centroidal MPC	2.00	12.01

VI. Conclusion and Future work

We have presented a complete method for real-time planning and controlling the locomotion of a legged robot over challenging terrains. Our framework plans a feasible contact sequence for the Solo robot at about 10 Hz, while the contact positions and end-effector trajectories are locally adapted at 1 kHz, synchronously with the low-level control of the robot. We have reported the details of the complete implementation for our quadruped robot Solo, and the code will be made available under an open license. Our experiments empirically validate our approach on scenarios that highlight the contribution of each module of the framework.

A potential limitation of our framework is linked to our research hypothesis, as the contact surface selected for the currently flying effector cannot change, which could result in failures if the robot is pushed. We plan to investigate whether re-planning of the next contact surface would be fast enough to prevent a fall in that case. The same computational time constraints also limit the completeness of our collision avoidance method as the only surfaces checked for collisions are the ones involved in the contact creation, preventing the robot from robustly stepping over obstacles. Future work will consider checking all potentially colliding obstacles, possibly by lowering the frequency of the collision avoidance QP.

Future developments will also include the adaptation of the base trajectory to the contact plan, in order to ensure their accordance. For now, if the contact planner fails in finding a plan that matches it (if an obstacle prevents the robot from going in the reference velocity direction), the robot will fall. This is currently handled by considering that the reference velocity is given by a user who has a knowledge of the environment and will not send the robot towards areas impossible to reach. However the contact planner could be augmented to output an adapted base trajectory, possibly solving this issue.

Finally, the fact that a short horizon is enough to achieve most of the considered scenarios motivates us to extend our framework towards more dynamic scenarios that will require further planning, including jumping motions. These scenarios may invalidate the Raibert heuristic and require the proposition of more advanced formulations of the dynamics.

References

[1] 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 *IEEE IROS*, 2018.

[2] J. Siekmann, K. Green, J. Warila, A. Fern, and J. Hurst, "Blind bipedal stair traversal via sim-to-real reinforcement learning," *Robotics: Science and Systems*, 2021.

[3] "Cassie: Dynamic planning on stairs," https://www.youtube.com/watch?v=qV-92Bq96Co.

[4] K. Hauser, T. Bretl, J.-C. Latombe, K. Harada, and B. Wilcox, "Motion planning for legged robots on varied terrain," *The International Journal of Robotics Research*, vol. 27.

[5] A. Escande, A. Kheddar, and S. Miossec, "Planning contact points for humanoid robots," *Robotics and Autonomous Systems*, vol. 61, no. 5, 2013.

[6] T. Bretl, "Motion planning of multi-limbed robots subject to equilibrium constraints: The free-climbing robot problem," *The International Journal of Robotics Research*, vol. 25, 2006.

[7] S. Kajita, F. Kanehiro, K. Kaneko, K. Fujiwara, K. Harada, K. Yokoi, and H. Hirukawa, "Biped walking pattern generation by using preview control of zero-moment point," in *IEEE ICRA*, 2003.

[8] D. E. Orin, A. Goswami, and S.-H. Lee, "Centroidal dynamics of a humanoid robot," *Autonomous Robots*, vol. 35, 2013.

[9] B. Ponton, A. Herzog, A. Del Prete, S. Schaal, and L. Righetti, "On time optimization of centroidal momentum dynamics," in *IEEE ICRA*, 2018.

[10] J. Carpentier and N. Mansard, "Multicontact locomotion of legged robots," *IEEE Transactions on Robotics*, vol. 34, no. 6, 2018.

[11] P. Fernbach, S. Tonneau, O. Stasse, J. Carpentier, and M. Taïx, "C-croc: Continuous and convex resolution of centroidal dynamic trajectories for legged robots in multicontact scenarios," *IEEE Transactions on Robotics*, vol. 36, no. 3, 2020.

[12] R. Deits and R. Tedrake, "Footstep planning on uneven terrain with mixed-integer convex optimization," in *IEEE - RAS international conference on humanoid robots*, 2014.

[13] B. Ponton, M. Khadiv, A. Meduri, and L. Righetti, "Efficient multicontact pattern generation with sequential convex approximations of the centroidal dynamics," *IEEE Transactions on Robotics*, vol. PP, 2021.

[14] R. J. Griffin, G. Wiedebach, S. McCrory, S. Bertrand, I. Lee, and J. Pratt, "Footstep planning for autonomous walking over rough terrain," in *IEEE Humanoids*, 2019.

[15] S. Tonneau, D. Song, P. Fernbach, N. Mansard, M. Taïx, and A. Del Prete, "SL1M: Sparse L1-norm minimization for contact planning on uneven terrain," in *IEEE ICRA*, 2020.

[16] D. Song, P. Fernbach, T. Flayols, A. Del Prete, N. Mansard, S. Tonneau, and Y. J. Kim, "Solving footstep planning as a feasibility problem using l1-norm minimization," *IEEE Robotics and Automation Letters*, vol. 6, no. 3, 2021.
[17] V. Tsounis, M. Alge, J. Lee, F. Farshidian, and M. Hutter, "DeepGait: Planning and control of quadrupedal gaits using deep reinforcement learning," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, 2020.
[18] T. Miki, J. Lee, J. Hwangbo, L. Wellhausen, V. Koltun, and M. Hutter, "Learning robust perceptive locomotion for quadrupedal robots in the wild," *Science Robotics*, vol. 7, no. 62, 2022.
[19] M. H. Raibert, *Legged robots that balance*. MIT press, 1986.
[20] P.-A. Léziart, T. Flayols, F. Grimminger, N. Mansard, and P. Souères, "Implementation of a reactive walking controller for the new open-hardware quadruped solo-12," 2021.
[21] F. Grimminger, A. Meduri, M. Khadiv, J. Viereck, M. Wüthrich, M. Naveau, V. Berenz, S. Heim, F. Widmaier, T. Flayols *et al.*, "An open torque-controlled modular robot architecture for legged locomotion research," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, 2020.
[22] S. Tonneau, P. Fernbach, A. Del Prete, J. Pettré, and N. Mansard, "2pac: Two-point attractors for center of mass trajectories in multi-contact scenarios," *ACM Trans. on Graph. (TOG)*, vol. 37, no. 5, 2018.
[23] F. Grimminger, A. Meduri, M. Khadiv, J. Viereck, M. Wüthrich, M. Naveau, V. Berenz, S. Heim, F. Widmaier, T. Flayols *et al.*, "An open torque-controlled modular robot architecture for legged locomotion research," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, 2020.
[24] "Project page," https://gepettoweb.laas.fr/articles/risbourg_corberes_2022.html.
[25] T. Flayols, A. Del Prete, P. Wensing, A. Mifsud, M. Benallegue, and O. Stasse, "Experimental evaluation of simple estimators for humanoid robots," in *IEEE Humanoids*, 2017.
[26] 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 *IEEE Humanoids*, 2015.
[27] B. Stellato, G. Banjac, P. Goulart, A. Bemporad, and S. Boyd, "Osqp: An operator splitting solver for quadratic programs," *Mathematical Programming Computation*, vol. 12, no. 4, 2020.
[28] J. Carpentier, G. Saurel, G. Buondonno, J. Mirabel, F. Lamiraux, O. Stasse, and N. Mansard, "The Pinocchio C++ library: A fast and flexible implementation of rigid body dynamics algorithms and their analytical derivatives," in *IEEE/SICE International Symposium on System Integration*, 2019.
[29] E. Coumans and Y. Bai, "PyBullet, a python module for physics simulation for games, robotics and machine learning," http://pybullet.org, 2016–2021.
[30] 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:1909.06586*, 2019.