

A Kinodynamic steering-method for legged multi-contact locomotion

Pierre Fernbach, Steve Tonneau, Andrea del Prete, Michel Taïx

▶ **To cite this version:**

Pierre Fernbach, Steve Tonneau, Andrea del Prete, Michel Taïx. A Kinodynamic steering-method for legged multi-contact locomotion. 2017. hal-01486933v1

HAL Id: hal-01486933

https://laas.hal.science/hal-01486933v1

Preprint submitted on 12 Apr 2017 (v1), last revised 26 Jul 2017 (v2)

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

A Kinodynamic steering-method for legged multi-contact locomotion.

Pierre Fernbach[1 2], Steve Tonneau[1 2], Andrea Del Prete[1 2] and Michel Taïx[1 2]

***Abstract*— We present a novel method for synthesizing collision-free, dynamic locomotion behaviors for legged robots, including jumping, going down a very steep slope, or recovering from a push using the arms of the robot. The approach is automatic and generic: non-gaited motions, comprising arbitrary contact postures can be generated along any environment.**

At the core of our framework is a new steering method that generates trajectories connecting two states of the robot. These trajectories account for the state-dependent, centroidal dynamic constraints inherent to legged robots. The method, of low dimension, formulated as a Linear Program, is really efficient to compute, and can find an application in various problems related to legged locomotion.

By incorporating this steering method into an existing sampling-based contact planner, we propose the first kinodynamic contact planner for legged robots.

I. INTRODUCTION

Legged robots move by creating contacts with the environment. Now that gaited motions on flat environments are commonly achieved [1], the community is interested in synthesizing more complex, *multi-contact* motions [2].

In our definition a *multi-contact* motion presents either or several of the following properties: the contact points are not coplanar, thus simplified dynamic models such as the linear inverted pendulum [3] do not apply [4]; as opposed to walking, the contacts are not created deterministically, resulting in a combinatorial issue in their choice [5]; the complex environment results in a high risk of collisions, and local planning methods getting stuck in local minima [6]. Addressing these issues simultaneously remains open.

To reduce the complexity of the problem, existing multi-contact planners make simplifying assumptions, e.g. that the computed motion comprises static postures [7], [5], [8]. Synthesizing dynamic motions such as jumping or going down steep slopes is thus impossible.

Our work aims at removing this limitation, by proposing a *kinodynamic planner* for multi-contact motions. So far the approach was restricted to manipulator arms [9], which have simpler dynamics. Kinodynamic planning requires:

- a *steering* method that heuristically generates a trajectory between two states of the robot;
- a *trajectory validation* method that verifies that the trajectory is collision-free and dynamically feasible.

The efficiency of a planner depends on the computational efficiency of both methods, and on the success rates of the validation. We thus introduce two such methods, efficient to compute, specifically designed for capturing the dynamics constraints of legged robots. They are efficiently written using two Linear Programs (LP) we introduce. We then integrate these methods within our multi-contact planner [8], and use it to plan highly-dynamic motions in various scenarios with the robots HRP-2 and HyQ.

[1] CNRS, LAAS, 7 avenue du colonel Roche, F31400 Toulouse, France
[2] Univ de Toulouse, LAAS, F31400 Toulouse, France

A. State of the art

1) Multi-contact planning: Bretl et al. showed in [10] that a multi contact planner must address two simultaneous problems: computing a relevant guide trajectory for the root of the robot in $SE(3)$ and planning a discrete sequence of contact configurations in equilibrium along the trajectory. Several contributions formulate the issue as an optimization problem [11], [6], that can get stuck in local minima, thus failing to address some of the scenarios we tackle.

Other approaches first plan the root path heuristically, then generate a contact sequence along this path [5], [12]. We proposed the first method able to achieve interactive performance [13], before a similar approach was proposed by [14]. Unfortunately these geometric approaches require each contact phase (i.e. a fixed set of contact locations) to be in static equilibrium. Jumping or walking down steep slopes is therefore impossible.

2) Kinodynamic planning: These approaches are well known for planning dynamic motions for manipulators [15], [16], [9]. However they do not trivially apply to our case. First, velocities and accelerations must be added to the system state, thus increasing dramatically the dimensionality of the problem. Then, because of the dynamic constraints (such as non-slipping contacts), the acceleration bounds depend on both the contact locations and the position of the Center of Mass (COM), while existing planners require constant bounds on the accelerations. To tackle the dimensionality issue, Pham et al. [17] reduce the dimension of the problem by applying a time-re-parametrization algorithm on a path in the configuration space, but the method has only been applied to manipulators. Our method further reduces dimensionality of the problem by focusing on the centroidal dynamics of the robot, while formulating a steering method accounting for variable acceleration bounds.

3) Computing dynamic constraints: Several methods exist in the literature to check the *static* equilibrium of the robot in non-coplanar cases, which we list in [18]. Some of them can apply to dynamic scenarios [4], [19]. The efficiency of the method depends on the number of tests that must be performed for a given state. Given that we compute the dynamic constraints only a few times for a given state, we use the Linear Programming approach, which outperforms the other approaches.

It thus appears that no existing multi-contact planners can produce highly-dynamic motion in complex environments without avoiding local minima, an issue our method tackles.

B. Contributions

This paper presents two theoretical and one practical contributions:

- An efficient LP to test the dynamic equilibrium of a robot, faster than previous approaches in our case;
- An efficient LP to determine the acceleration bounds on the Center of Mass (COM) of a robot, given its active contacts and a desired direction of acceleration;
- One of the first kinodynamic planners able to synthesize truly dynamic multi-contact motions for legged robots.

In the following, we rigorously formulate our problem and give relevant definitions used throughout the paper (Section II). We then detail the construction of our new steering and trajectory validation methods for multi-contact locomotion (Section III). After that we discuss the integration of the methods within our existing contact planner (Section IV), and present the results obtained with the method (Section V), leaving the discussion on the implications of our work for the end of the paper (Section VI).

II. PROBLEM DEFINITION AND OVERVIEW

We formulate the motion planning problem as the issue of finding a dynamically feasible, collision-free trajectory $\mathbf{S}(t)$ between two states $\mathbf{S}_{init}$ and $\mathbf{S}_{goal}$ of a legged robot. This problem is addressed with an RRT-connect algorithm [20], where new steering and path validation methods are introduced to respect the dynamics of the robot.

Our steering method generates a trajectory that is feasible in the neighborhood of the initial state.

Our path validation method (or, more accurately, trajectory validation method) determines precisely the part of the trajectory that is actually feasible, such that all states of the generated graph are connected through dynamically feasible, collision free trajectories.

A. Definitions

We define a *state* as follows :

$$\mathbf{S} =< \mathbf{X}, \mathbf{P}, \mathbf{N}, \mathbf{q} > \tag{1}$$

where:

- $\mathbf{X} =< \mathbf{c}, \dot{\mathbf{c}}, \ddot{\mathbf{c}} >$ denote the position, velocity and acceleration of the COM;
- $\mathbf{P} = \left[\mathbf{p}_1 \dots \mathbf{p}_k\right]$, with $\mathbf{p}_i \in \mathbb{R}^3$ the position of the i-th contact point;
- $\mathbf{N} = \left[\mathbf{n}_1 \dots \mathbf{n}_k\right]$, with $\mathbf{n}_i \in \mathbb{R}^3$ the surface normal of the i-th contact point;
- $\mathbf{q} \in SE(3) \times \mathbb{R}^n$ is the configuration of the robot, described by n degrees of freedom (dofs) and the root location. Note that $\mathbf{q}$ uniquely defines $\mathbf{c}$.

All positions are expressed in the world frame, and each state has a variable number k of contacts.

B. Rationale

At each step of a motion planning algorithm, an *extension method* (or a *steering method*, as named in kinodynamic planning) is called [21]. The steering method is a heuristic approach to try to connect two states with a feasible trajectory. The inputs of this steering method are an initial and a goal state $\mathbf{S}_0 =< \mathbf{X}_0, \mathbf{P}_0, \mathbf{N}_0, \mathbf{q}_0 >$ and $\mathbf{S}_1 =< \mathbf{X}_1, \mathbf{P}_1, \mathbf{N}_1, \mathbf{q}_1 >$. The steering method returns a time-optimal trajectory of duration t_f:

$$\mathbf{X} : t \in [0, t_f] \mapsto < \mathbf{c}(t), \dot{\mathbf{c}}(t), \ddot{\mathbf{c}}(t) >$$

such that $\mathbf{X}(0) = \mathbf{X}_0$ and $\mathbf{X}(t_f) = \mathbf{X}_1$, and $\mathbf{X}(t)$ is constrained to satisfy the COM acceleration bounds imposed by the non-sliding constraints at $\mathbf{S}_0$.

The trajectory is extended into a whole-body trajectory by performing a constrained interpolation between $\mathbf{q}_0$ and $\mathbf{q}_1$ to follow the COM [22]. It is then validated according to several constraints, including in our case: collision avoidance, user-defined bounds on position, and dynamic equilibrium (verified by the non-sliding constraints associated with each contact phase). Additional user-defined bounds on the velocity and acceleration of the COM are guaranteed to be respected by the steering method, and thus do not need verification [9]. The path validation method only returns the part of trajectory that respects these constraints:

$$\mathbf{X}' : t \in \left[0, t'_f \leq t_f\right] \mapsto \mathbf{X}(t) \tag{2}$$

III. STEERING METHOD AND TRAJECTORY VALIDATION

Our method derives from the well-known Double Integrator Minimum Time (DIMT) method [15], [16], [9]. We use the DIMT as a black-box method, and refer to [9] for details. We give the inputs and outputs of the method in our case, before discussing its extension to legged locomotion.

Given user-defined **symmetric** bounds on the COM dynamics along three orthogonal axis:

$$\begin{aligned} -\dot{\mathbf{c}}^{max}_{\{\mathbf{x},\mathbf{y},\mathbf{z}\}} \leq \quad \dot{\mathbf{c}}_{\{\mathbf{x},\mathbf{y},\mathbf{z}\}} \leq \quad \dot{\mathbf{c}}^{max}_{\{\mathbf{x},\mathbf{y},\mathbf{z}\}} \\ -\ddot{\mathbf{c}}^{max}_{\{\mathbf{x},\mathbf{y},\mathbf{z}\}} \leq \quad \ddot{\mathbf{c}}_{\{\mathbf{x},\mathbf{y},\mathbf{z}\}} \leq \quad \ddot{\mathbf{c}}^{max}_{\{\mathbf{x},\mathbf{y},\mathbf{z}\}} \end{aligned} \tag{3}$$

and given an initial state $\mathbf{S}_0$, with $\mathbf{X}_0 =< \mathbf{c}_0, \dot{\mathbf{c}}_0, \mathbf{0} >$, and a target state $\mathbf{S}_1$ with $\mathbf{X}_1 =< \mathbf{c}_1, \dot{\mathbf{c}}_1, \mathbf{0} >$, the DIMT outputs a minimum time trajectory $\mathbf{X}(t)$ that connects exactly $\mathbf{X}_0$ and $\mathbf{X}_1$, without considering collision avoidance. This trajectory consists, for each joint, in phases of constant accelerations in a direction.

The main issue is that for a legged robot, the center of mass acceleration bounds are neither constant nor symmetric, but state-dependent (Fig. 1). The bounds correspond to the non-slipping condition, and are thus determined by the COM position, as well as the contact points and normals.

To address this issue, we propose a two-step method:

- First, we use the DIMT method with acceleration constraints computed for the initial state $\mathbf{S}_0$. By doing so, we increase the odds that the trajectory $\mathbf{X}(t)$ be dynamically feasible in the neighborhood of $\mathbf{S}_0$, but not along the complete trajectory.

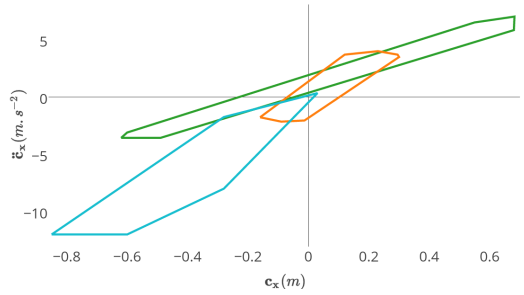


Fig. 1: Examples of state-dependent dynamic constraints for HRP-2. Each color corresponds to a different non coplanar contact configuration for both feet, while the root position remains the same. Each polygon represents the pairs center of mass position / acceleration admissible regarding the non-slipping condition, considering only the $\mathbf{x}$ axis.

- Then, in the trajectory validation phase, we verify the dynamic equilibrium of the robot, additionally to collision avoidance, as we progress along the trajectory. The returned trajectory $\mathbf{X}'(t)$ is the part of $\mathbf{X}(t)$ that satisfies all these constraints.

A. Trajectory validation

We first describe how our trajectory validation method is implemented, because its formulation serves as a basis for developing our steering method.

1) Inputs and outputs: We consider as inputs two states $\mathbf{S}_0$ and $\mathbf{S}_1$, and a trajectory $\mathbf{X}(t)$ connecting them. Additionally, we consider a discrete set of contact phases configurations along the trajectory, denoted $\mathbf{P}(t)$ and $\mathbf{N}(t)$, with:

$$\mathbf{P}(0) = \mathbf{P}_0, \quad \mathbf{P}(t_f) = \mathbf{P}_1$$
$$\mathbf{N}(0) = \mathbf{N}_0, \quad \mathbf{N}(t_f) = \mathbf{N}_1$$

The definition of the contact-switch timings is out of the scope of this paper. In Section IV-A.2 we propose a heuristic approach, specific to our motion planner.

The output of the trajectory validation is a dynamically feasible, collision free sub-trajectory of $\mathbf{X}$, $\mathbf{X}'(t)$. To do so, as for a classical path validation method, we discretize the trajectory, and verify at each discretization step the constraints. Collision checking is performed classically, thus we focus on the dynamic equilibrium constraints.

2) Dynamic equilibrium: We formulate a test for dynamic equilibrium as an LP, extending the static equilibrium test proposed in [18]. The Newton-Euler equations verify:

$$\begin{bmatrix} m(\ddot{\mathbf{c}} - \mathbf{g}) \\ m\mathbf{c} \times (\ddot{\mathbf{c}} - \mathbf{g}) + \dot{\mathbf{L}} \end{bmatrix} = \begin{bmatrix} \mathbf{I}_3 & \dots & \mathbf{I}_3 \\ \hat{\mathbf{p}}_1 & \dots & \hat{\mathbf{p}}_k \end{bmatrix} \mathbf{f} \tag{4}$$

Where :

- m is the total mass of the robot;
- $\mathbf{f} = \begin{bmatrix}\mathbf{f}_1, \mathbf{f}_2, ..., \mathbf{f}_k\end{bmatrix}^T \in \mathbb{R}^{3k}$ is the stacked vector of contact forces $\mathbf{f}_i$, applied at contact position $\mathbf{p}_i$;
- $\mathbf{g} = \begin{bmatrix}0 & 0 & -9.81\end{bmatrix}^T$ is the gravity vector;
- $\mathbf{L} \in \mathbb{R}^3$ is the angular momentum (expressed at $\mathbf{c}$).
- $\hat{\mathbf{p}}_i$ denotes the skew-symmetric matrix associated to $\mathbf{p}_i$.

To respect the non-slipping condition, we constrain the contact forces $\mathbf{f}_i$ to lie inside a linearized friction cone of generators $\mathbf{V}_i$, such that [4]:

$$\mathbf{f_i} = \mathbf{V_i}\boldsymbol{\beta_i} \text{ with } \boldsymbol{\beta_i} \in \mathbb{R}^{\mathbf{4}} \text{ and } \boldsymbol{\beta_i} \geq \mathbf{0} \tag{5}$$

which leads to

$$\mathbf{f} = \mathbf{V}\boldsymbol{\beta} \text{ with } \boldsymbol{\beta} \in \mathbb{R}^{\mathbf{4k}} \text{ and } \boldsymbol{\beta} \geq \mathbf{0} \tag{6}$$

where $\mathbf{V} = diag(\begin{bmatrix}\mathbf{V}_1 & \dots & \mathbf{V}_k\end{bmatrix}) \in \mathbb{R}^{3k \times 4k}$.

We set $\dot{\mathbf{L}} = 0$ as classically done [23] and rewrite (4):

$$\underbrace{m\begin{bmatrix}\mathbf{I}_3 \\ \hat{\mathbf{c}}\end{bmatrix}}_{\mathbf{H}} \ddot{\mathbf{c}} + \underbrace{m\begin{bmatrix}-\mathbf{g} \\ \mathbf{c} \times -\mathbf{g}\end{bmatrix}}_{\mathbf{h}} = \underbrace{\begin{bmatrix}\mathbf{I}_3 & \dots & \mathbf{I}_3 \\ \hat{\mathbf{p}}_1 & \dots & \hat{\mathbf{p}}_k\end{bmatrix}}_{\mathbf{G}} \mathbf{V}\boldsymbol{\beta} \tag{7}$$

Thus, if there exists a $\boldsymbol{\beta}^*$ such that $\boldsymbol{\beta}^* \geq \mathbf{0}$ and (7) is satisfied, it means the robot is in dynamic equilibrium. Our test then boils down to solving the following LP:

$$\begin{aligned} &\text{find } \boldsymbol{\beta} \\ &\text{s.t. } \mathbf{G}\boldsymbol{\beta} = \mathbf{H}\ddot{\mathbf{c}} + \mathbf{h} \\ &\quad \boldsymbol{\beta} \geq 0 \end{aligned} \tag{8}$$

B. Computing the acceleration bounds for the steering method

Computing plausible bounds for $\ddot{\mathbf{c}}$ is essential to increase the success rate of the steering method. To estimate the bounds in the neighborhood of $\mathbf{S}_0$ we proceed as follows:

- We call a first time the DIMT with arbitrary large bounds: $-\ddot{\mathbf{c}}^{\infty}_{\{\mathbf{x},\mathbf{y},\mathbf{z}\}} \leq \ddot{\mathbf{c}}_{\{\mathbf{x},\mathbf{y},\mathbf{z}\}} \leq \ddot{\mathbf{c}}^{\infty}_{\{\mathbf{x},\mathbf{y},\mathbf{z}\}}$. We obtain a trajectory composed of phases of constant accelerations. We then consider $\mathbf{a}$, the direction of the first phase;
- We compute the maximum acceleration $\ddot{\mathbf{c}}^{max} = \alpha^*\mathbf{a}$, $\alpha^* \in \mathbb{R}^+$ satisfying the non-slipping condition at $\mathbf{S}_0$;
- Finally, we compute $\mathbf{X}(t)$ by calling again the DIMT, using the projection of $\alpha^*\mathbf{a}$ along the $\mathbf{x}, \mathbf{y}, \mathbf{z}$ axes as bounds: $-(\alpha^*\mathbf{a})_{\{\mathbf{x},\mathbf{y},\mathbf{z}\}} \leq \ddot{\mathbf{c}}_{\{\mathbf{x},\mathbf{y},\mathbf{z}\}} \leq (\alpha^*\mathbf{a})_{\{\mathbf{x},\mathbf{y},\mathbf{z}\}}$.

If the acceleration direction $\mathbf{a}'$ returned by the second call to the DIMT is such that $\mathbf{a}' = \mathbf{a}$, then the bounds are accurate and the trajectory is dynamically valid in the neighborhood of $\mathbf{S}_0$. In Section V-B we show empirically that in the majority of cases $\mathbf{a}' = \mathbf{a}$, and that otherwise the bounds remain valid in the neighborhood of $\mathbf{S}_0$.

To compute α^* we extend (8), by first rewriting (7):

$$\mathbf{G}\boldsymbol{\beta} = \mathbf{H}\alpha\mathbf{a} + \mathbf{h}$$

$$\begin{bmatrix}\mathbf{G} & -(\mathbf{H}\mathbf{a})\end{bmatrix} \underbrace{\begin{bmatrix}\boldsymbol{\beta} \\ \alpha\end{bmatrix}}_{\boldsymbol{\gamma}} = \mathbf{h} \tag{9}$$

We can now write the following LP :

$$\begin{aligned} &\text{find } \boldsymbol{\gamma} \\ &\min \ -\alpha \\ &\text{s. t. } \begin{bmatrix}\mathbf{G} & -(\mathbf{H}\mathbf{a})\end{bmatrix}\boldsymbol{\gamma} = \mathbf{h} \\ &\quad \boldsymbol{\gamma} \geq \mathbf{0} \end{aligned} \tag{10}$$

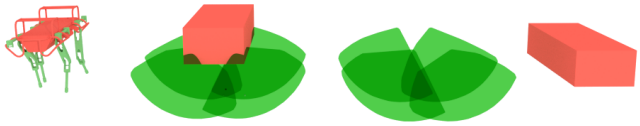


Fig. 2: The HyQ robot, and its simplified representation used to address $\mathcal{P}_1$. The green shapes represent the reachable workspace of the effectors. The red shape is a bounding box that must stay collision-free at all times.

If LP (10) has a solution, this state can achieve equilibrium and the optimal value α^* gives the maximal COM acceleration achievable along $\mathbf{a}$: $\ddot{\mathbf{c}}^{max} = \mathbf{a}\alpha_0^*$.

We have thus developed a criterion to efficiently verify the dynamic equilibrium of a robot, as well as the maximum COM acceleration in a given direction. With this criterion we write a steering method that generates a time-optimal trajectory from $\mathbf{S}_0$ to $\mathbf{S}_1$, feasible in the neighborhood of $\mathbf{S}_0$. The valid part of this trajectory is extracted in the path validation phase using a similar formulation.

IV. APPLICATION TO MULTI-CONTACT PLANNING

We integrated our methods within the RB-RRT planner [8] to generate dynamic multi-contact motions. We only recall the main aspects of the planner, and refer the reader to our technical report for other details [24].

RB-RRT is a hierarchical planner that decouples the motion planning problem in three different phases: first, the planning of path for the root of the robot, using an RRT with a simplified model of the robot ($\mathcal{P}_1$, Fig. 2); then the generation of a discrete contact sequence along this root path ($\mathcal{P}_2$), where all contact configurations are in static equilibrium; finally, the interpolation of the contact sequence into continuous motion for the robot ($\mathcal{P}_3$).

With our method, we transform the path planning problem $\mathcal{P}_1$ into a trajectory planning one, thus removing the constraint imposing the contact configurations at $\mathcal{P}_2$ to be in static equilibrium. This allows us to address new kind of scenarios, such as planning a jump, or going down a **non** *quasi-flat* steep slope [18] (a surface for which the friction cone does **not** contain the direction opposite to the gravity).

A. Planning a root trajectory ($\mathcal{P}_1$)

1) Integration of the steering method in $\mathcal{P}_1$: RB-RRT is efficient partly because when addressing $\mathcal{P}_1$, the contacts are not generated, to avoid handling a complex combinatorial. However, our steering method requires contact information. We thus approximate the state of the robot as follows: for a given configuration $\mathbf{q}$, we compute the intersection between the reachable workspace of each effector and the environment. (Fig. 3). We then assume that a contact exists at the center of this intersection. We also approximate the COM as the 3D position of the root, which is equivalent to having a robot with mass-less limbs [11].

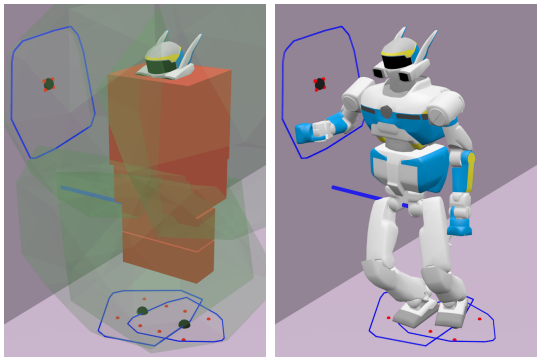


Fig. 3: Approximation of contact locations in $\mathcal{P}_1$. Left: the blue polygons represent the intersection between the reachable workspace and the environment. The centroid of this intersection serves as the estimated center of the contact location (black sphere), from which the contact extremities are extrapolated (red sphere). Right: the actual contact locations generated during $\mathcal{P}_2$.

2) Trajectory validation in $\mathcal{P}_1$: To validate the trajectory, we in turn approximate the contact phases $\mathbf{P}(t)$ and $\mathbf{N}(t)$. We assume that the contacts are sliding along with the COM trajectory, as long as the reachable workspace of the effectors is in collision with the same surface. When this is not true, we estimate the new contacts and continue. Thus $\mathbf{P}(t)$ and $\mathbf{N}(t)$ are piecewise-constant.

3) Generating jumps: The steering method can produce a trajectory $\mathbf{X}(t)$ between two states $\mathbf{S}_0$ and $\mathbf{S}_1$ that results in phases where the number of active contacts falls under a user-defined threshold (Fig. 4 - 1). When this is detected during trajectory validation, the planner can try to generate a jump (depending on a user-defined parameter), as follows:

1) identify the last state where the desired effectors are still in contact: $\mathbf{X}(t_{to})$ (Fig. 4 - 2);
2) then try to compute a feasible take-off velocity $\dot{\mathbf{c}}_{to}$ such that a collision free ballistic motion between $\mathbf{S}_{to}$ and the $\mathbf{S}_1$ is feasible. This is achieved by directly applying the method of Campana et al. [25] (Fig. 4 - 3);
3) compute a trajectory from $\mathbf{S}_0$ to $\mathbf{S}_{to}$ (Fig. 4 - 4).

If all these steps are successfully achieved, then a jump trajectory connecting $\mathbf{S}_0$ and $\mathbf{S}_1$ has been found (Fig. 4 - 5).

B. Generating dynamic contact configurations ($\mathcal{P}_2$)

In the original paper, a discrete sequence of contact configurations is computed along the root path, using an LP to test that the generated configurations are in static equilibrium. Our only modification to $\mathcal{P}_2$ is thus to replace this LP with (8), to generate dynamically feasible contact configurations, following the root trajectory computed in $\mathcal{P}_1$.

Finally, the trajectory interpolation $\mathcal{P}_3$ remains unchanged by these modifications.

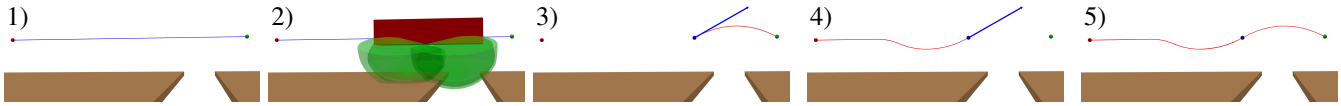


Fig. 4: A jump is composed of a ballistic and a preparation phase, planned sequentially during the path validation phase.

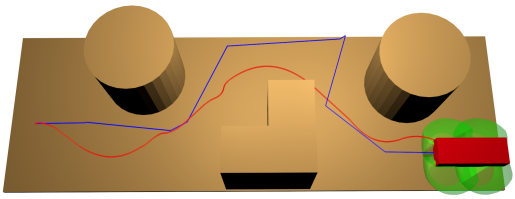


Fig. 5: The trajectories computed by the kinodynamic version of our planner (red) are visually more appealing than their quasi-static counterpart (blue).

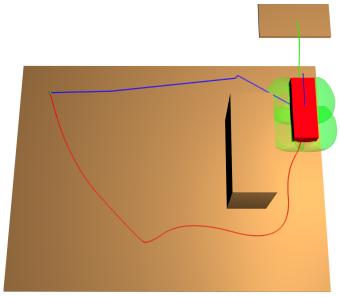


Fig. 6: The goal of this problem is located on the platform. While the blue trajectory is the shortest to reach to the shown position for a simplified robot, only the red one allows to reach the velocity required to perform the upcoming jump in green. Local methods won't be able to escape the blue trajectory, contrary to our planner.

V. RESULTS

A. Scenarios

We tested our planner in various scenarios presented in the companion video. Each scenario highlights a specific property of our planner. In the figures presented, the blue and yellow arrows indicate the velocity and acceleration directions, and their length is equal to their norm.

obstacle avoidance: We compare the trajectories generated with our planner against the results obtained with a quasi-static contact planner in a constrained environment that requires slaloming through obstacles, to show the visual benefits of our method (Fig. 5). We also demonstrate its interest over local methods by exhibiting a scenario where these approaches would get stuck in local minima (Fig. 6).

push recovery: HRP-2 is pushed towards a wall with a velocity of $1.5m.s^{-1}$. Our planner computes a recovery motion by using its arm (Fig. 3).

highly dynamic scenarios: HRP-2 goes down a 25° or 38° slope, on top of which no static equilibrium configuration exists (Fig. 7). Our planner is able to generate the motion that allows the robot to reach the end of the slope and stop safely. Similarly, we plan motions for HyQ along inclined planes, as well as jumping motions (Fig. 4).

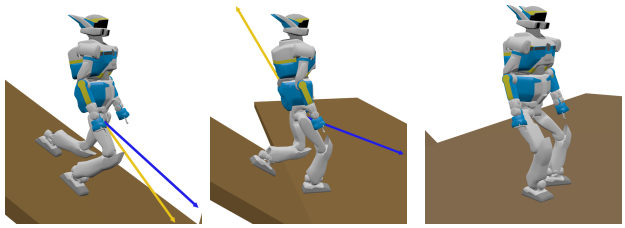


Fig. 7: Key captures of HRP-2 going down a 38° slope.

B. Benchmarks

To demonstrate the interest of our planner, we performed different types of benchmarks on the scenarios we propose. All benchmarks were made using one core of an Intel Xeon CPU E5-1630 v3 at 3.7GHz with 64Go of main memory. They measure the overall computation time of the method and the success rates of the approach (Table I). We also compared our steering method with naive approaches to demonstrate the comparative gain it brings (Table II).

Regarding time performances, Table I shows variations from a few seconds to a few minutes depending on the scenario, an amount of time comparable with the the literature in dynamic multi-contact locomotion [6], [11] while our scenarios are longer and more constrained [1].

The time spent by one call to the steering method and its dynamic validation (disregarding collision checking), respectively 0.35 ms and 0.26 ms in average, summing up to 0.61 ms. This time is negligeable with respect to the total time spent by one step of the planner, which averages to 43.75 ms. Similarly, when attempting to plan a jump, the average computation time for the steering method and dynamic validation is 2.55 ms, while the total time spent by one step including collisions is 9.32 ms.

Regarding the interest of the steering method, Table II shows that our approach outperforms any method using user-defined bounds: our trajectories are shorter and computed faster, thanks to a much lower rejection rate. Furthermore, the fact that our method automatically computes the bounds saves the need for a sensitive manual parameter tuning.

[1]We do not provide a more detailed comparison because the mentioned papers do not indicate precise computation times.

	Path Planning				Contact Generation		Total
Scenario	Valid traj. (%)	Valid dir. (%)	num. nodes	time (% tot. time)	time (% tot. time)	success (%)	time (s)
Slalom (HyQ)	95.72 %	65.6 %	491	69.56 %	30.43 %	90 %	85.55
Local minima (HyQ)	95.25 %	69.8 %	315	69.14 %	30.86%	75%	38.31
Steep slope (25°) (HRP-2)	82.4 %	58.3 %	3895	77.36 %	22.63%	85 %	165.61
Steep slope (38°) (HRP-2)	80.75 %	56.5 %	5533	87.35 %	12.64%	90 %	679.73
Push recovery (HRP-2)	100 %	100 %	2	0.08 %	99.92%	100 %	5.78
Inclined planes (HyQ)	100 %	100 %	2	0.20 %	99.80%	100 %	2.55

TABLE I: Computation times and success rates by scenario, averaged over 20 runs. *Valid traj.* is the average portion of the trajectories that were validated by the trajectory validation method. *Valid dir.* is the average amount of time the equality $\mathbf{a}' = \mathbf{a}$ is verified. *num nodes* is the average number of nodes generated by the planner. *success* records the percentage of trajectories that were successfully transformed into a whole-body motion.

Scenario : Slalom (HyQ)			
acc. bounds	time (s)	valid trajectories	solution length (s)
LP (ours)	59.5	95.7%	12.2
6	time-out	0.8%	X
4	913.3	69.1%	13.4
2	109.7	90.3 %	22.3

Scenario : Slope (38°) (HRP-2)			
acc. bounds	time (s)	valid trajectories	solution length (s)
LP (ours)	593.7	80.75 %	11.64
6	time-out	0.2%	X
4	10992.6	7.6%	16.98
2	time-out	8.8%	X

TABLE II: Comparison of planning performances obtained with user-defined bounds on the admissible acceleration and those computed with our method. *time* is the average time for solving the problem, *valid trajectories* is the average portion of the trajectories validated by the trajectory validation method, *solution length* is the total time of the solution trajectory. The time-out is set to 5 hours.

VI. DISCUSSION

The prototype presented in this work is based on an efficient formulation of the dynamics of a legged robot, which makes our theoretical contribution directly applicable with methods related to multi-contact locomotion.

Integrating the steering method within our reachability-based planner has required the introduction of some heuristics, inherent to its architecture: the contacts are considered to be sliding and always active, inaccurate in the general case. However, the desired output of $\mathcal{P}_1$ is not the final root trajectory, but rather a very good initial guess. Indeed, the approximations are corrected at the contact generation and interpolation phase ($\mathcal{P}_2$ and $\mathcal{P}_3$), as demonstrated in our scenarios: our planner is able to compute highly-dynamic motions while escaping local minima, on cases never demonstrated in the literature.

Therefore existing local methods [11], [6] could directly benefit from the trajectories we compute as relevant initial guesses, allowing them to escape local minima and to converge faster. However we believe that our planner is more generic, and suited to handle constrained cases and collisions (Deits et al. only consider footstep planning, while Mordatch et al. do not guarantee collision avoidance).

Regarding the work of Pham et al. [17], despite the dimensionality reduction achieved by the author, our belief is that their method does not directly apply to multi-contact locomotion planning, because considering all the degrees of freedom of a humanoid robot, while addressing the contact combinatorial, is simply to expensive. For future work, we would like to study the adaptation of their formulation of the problem with our centroidal dynamics approach to reduce its dimensionality, for means of comparison with the method we propose.

VII. CONCLUSION

In this paper we have presented a steering and a trajectory validation method, designed to address the open multi-contact planning problem. These methods are based on two efficient Linear Programs, designed to efficiently capture the complex, state-dependent dynamics of a legged robot.

Their interest is demonstrated through an integration within a sampling-based multi-contact planner, which we use to generate highly-dynamic motions, such as jumps, or navigation through really steep slopes. These scenarios cannot be addressed with existing planners. Furthermore our method is global and does not get stuck in local minima, and could be used to compute initial guesses for other motion generation methods.

Our method performs at least as good as existing dynamic approaches, although obtaining interactive computation times is an objective we will pursue for future work. Another issue we would like to address is to propose a shortcut algorithm for the trajectories computed by our planner.

REFERENCES

[1] 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 Inter. Conf. on Robotics and Automation (ICRA)*, vol. 2, 2003.

[2] C. G. Atkeson, B. P. W. Babu, N. Banerjee, D. Berenson, C. P. Bove, X. Cui, M. DeDonato, R. Du, S. Feng, P. Franklin, M. Gennert, J. P. Graft, P. He, A. Jaeger, K. K. J. Kim, L. Li, X. L. C. Liu, T. Padir, F. Polido, G. G. Tighe, and X. Xinjilefu, "What happened at the darpa robotics challenge, and why?" Carnegie Mellon University, Pittsburgh, USA, Tech. Rep., 2015.

[3] J. Pratt, J. Carff, S. Drakunov, and A. Goswami, "Capture Point: A Step toward Humanoid Push Recovery," *IEEE-RAS Inter. Conf. on Humanoid Robots (Humanoids)*, 2006.

[4] Z. Qiu, A. Escande, A. Micaelli, and T. Robert, "Human motions analysis and simulation based on a general criterion of stability," in *Inter. Symposium on Digital Human Modeling*, 2011.

[5] A. Escande, A. Kheddar, S. Miossec, and S. Garsault, "Planning Support Contact-Points for Acyclic Motions and Experiments on HRP-2," in *Springer Tracts in Advanced Robot (ISER)*, O. Khatib, V. Kumar, and G. J. Pappas, Eds., vol. 54, 2008.

[6] R. Deits and R. Tedrake, "Footstep planning on uneven terrain with mixed-integer convex optimization," in *IEEE-RAS Inter. Conf. on Humanoid Robots (Humanoids)*, Madrid, Spain, 2014.

[7] T. Bretl, "Motion planning of multi-limbed robots subject to equilibrium constraints: The free-climbing robot problem," *The Inter. Journal of Robotics Research*, vol. 25, no. 4, Apr. 2006.

[8] S. Tonneau, N. Mansard, C. Park, D. Manocha, F. Multon, and J. Pettré, "A reachability-based planner for sequences of acyclic contacts in cluttered environments," in *Int. Symp. Robotics Research (ISRR)*, Sestri Levante, Italy, 2015.

[9] T. Kunz and M. Stilman, "Probabilistically complete kinodynamic planning for robot manipulators with acceleration limits," in *IEEE/RSJ Inter. Conf. on Intelligent Robots and Systems (IROS)*, 2014.

[10] T. Bretl, S. M. Rock, J.-C. Latombe, B. Kennedy, and H. Aghazarian, "Free-climbing with a multi-use robot," in *Inter. Symposium on Experimental Robotic (ISER)*, 2004.

[11] I. Mordatch, E. Todorov, and Z. Popović, "Discovery of complex behaviors through contact-invariant optimization," *ACM Trans. on Graphics*, vol. 31, no. 4, pp. 43:1–43:8, 2012.

[12] K. Bouyarmane, A. Escande, F. Lamiraux, and A. Kheddar, "Potential field guide for humanoid multicontacts acyclic motion planning," in *IEEE Int. Conf. on Robotics and Automation (ICRA)*, Japan, 2009.

[13] J. Carpentier, S. Tonneau, M. Naveau, O. Stasse, and N. Mansard, "A versatile and efficient pattern generator for generalized legged locomotion," in *IEEE Int. Conf. on Robotics and Automation (ICRA)*, Stockholm, Sweden, May 2016.

[14] M. X. Grey, A. D. Ames, and C. K. Liu, "Footstep and motion planning in semi-unstructured environments using randomized possibility graphs," in *IEEE Inter. Conf. on Robotics and Automation (ICRA)*, 2017.

[15] T. Kröger, A. Tomiczek, and F. M. Wahl, "Towards on-line trajectory computation," in *IEEE/RSJ Inter. Conf. on Intelligent Robots and Systems (IROS)*, 2006.

[16] K. Hauser and V. Ng-Thow-Hing, "Fast smoothing of manipulator trajectories using optimal bounded-acceleration shortcuts," in *IEEE Inter. Conf. on Robotics and Automation (ICRA)*, May 2010.

[17] Q.-C. Pham, S. Caron, P. Lertkultanon, and Y. Nakamura, "Admissible velocity propagation: Beyond quasi-static path planning for high-dimensional robots," *Inter. Journal of Robotics Research*, first published November 1, 2016.

[18] A. Del Prete, S. Tonneau, and N. Mansard, "Fast Algorithms to Test Robust Static Equilibrium for Legged Robots," in *IEEE Inter. Conf. on Robotics and Automation (ICRA)*, Stockholm, Sweden, 2016.

[19] S. Noda, M. Murooka, S. Nozawa, Y. Kakiuchi, K. Okada, and M. Inaba, "Generating whole-body motion keep away from joint torque, contact force, contact moment limitations enabling steep climbing with a real humanoid robot," in *IEEE Inter. Conf. on Robotics and Automation (ICRA)*, May 2014.

[20] J. Kuffner Jr and S. LaValle, "RRT-connect: An efficient approach to single-query path planning," in *IEEE Inter. Conf. on Robotics and Automation (ICRA)*, vol. 2, no. April, 2000.

[21] S. M. LaValle, *Planning Algorithms*. New York, NY, USA: Cambridge University Press, 2006.

[22] J. Mirabel, S. Tonneau, P. Fernbach, A. K. Seppala, M. Campana, N. Mansard, and F. Lamiraux, "HPP: A new software for constrained motion planning," in *IEEE/RSJ Inter. Conf. on Intelligent Robots and Systems (IROS)*, Oct 2016.

[23] S. Caron, Q.-C. Pham, and Y. Nakamura, "Leveraging Cone Double Description for Multi-contact Stability of Humanoids with Applications to Statics and Dynamics," in *Robotics, Science and Systems (RSS)*, 2015.

[24] S. Tonneau, A. D. Prete, J. Pettré, C. Park, D. Manocha, and N. Mansard, "An efficient acyclic contact planner for multiped robots," in *Technical report*, 2016.

[25] M. Campana, P. Fernbach, S. Tonneau, M. Taïx, and J.-P. Laumond, "Ballistic motion planning for jumping superheroes," in *Inter. Conf. on Motion in Games (MIG)*. San Francisco, California: ACM, 2016, pp. 133–138.