An efficient acyclic contact planner for multiped robots

Steve Tonneau, Andrea Del Prete, *Member, IEEE,* Julien Pettré, *Member, IEEE,* Chonhyon Park, *Student Member, IEEE,* Dinesh Manocha, *Member, IEEE,* and Nicolas Mansard, *Member, IEEE,*

***Abstract*—We present a contact planner for complex legged locomotion tasks: standing up, climbing stairs using a handrail, crossing rubble and getting out of a car. The need for such a planner was shown at the Darpa Robotics Challenge, where such behaviors could not be demonstrated (except for egress).**

Current planners suffer from their prohibitive algorithmic complexity, because they deploy a tree of robot configurations projected in contact with the environment.

We tackle this issue by introducing a reduction property: the reachability condition. This condition defines a geometric approximation of the contact manifold, which is of low dimension, presents a Cartesian topology, and can be efficiently sampled and explored. The hard contact planning problem can then be decomposed into two sub-problems: first, we plan a path for the root without considering the whole-body configuration, using a sampling-based algorithm; then, we generate a discrete sequence of whole-body configurations in static equilibrium along this path, using a deterministic contact-selection algorithm.

The reduction breaks the algorithm complexity encountered in previous works, resulting in the first interactive implementation of a contact planner (open source). While no contact planner has yet been proposed with theoretical completeness, we empirically show the interest of our framework: in a few seconds, with high success rates, we generate complex contact plans for various scenarios and two robots, HRP-2 and HyQ. These plans are validated either in dynamic simulations, or on the real HRP-2 robot.

***Index Terms*—Multi contact locomotion, centroidal dynamics, Humanoid robots, legged robots, motion planning**

I. Introduction

LEGGED robots move by sequentially creating contacts with the environment. After years of research, such robots can autonomously walk on flat ground, but struggle to navigate more complex environments. Deciding where to create a contact with its feet and possibly its hands is nontrivial, e.g. to climb stairs using a handrail.

Most of the complexity of this problem lies in the contact planning, i.e. the underlying decomposition of the trajectory into contact phases where specific points of the robot body are exerting forces on specific locations of the environment. Tackling this complexity is the main objective of this paper.

Once the contact plan is known, efficient approaches exist to generate a dynamically feasible motion [1]. In the specific (and simple) case of gaited biped locomotion on flat ground, choosing the effector with which to create a contact is trivial because walking follows a cyclic pattern (the left foot always follows the right foot). Efficient tools such as the capture point [2] can be used to compute the next contact location. In the general case planning complex contact interactions is extremely challenging: At any given time a contact must be chosen between infinitely many possibilities (often a combinatorial discrete choice for the effector and contact surface, and a continuous choice for the contact location). Furthermore, a contact choice constrains kinematically and dynamically the possible motions, and there is no analytical way to verify whether this choice brings the robot one step closer to the desired goal or to a dead end, especially in the presence of obstacles; we say that the contact manifold is foliated [3]. The foliation prevents the use of efficient sampling-based planners for two reasons. (i) First, each sub-manifold of the foliation has a zero measure and cannot be directly sampled. A sample is rather obtained by sampling a free flying configuration and explicitly projecting it in contact (which is a costly numerical operation). (ii) Second, the foliated topology turns the exploration by spreading a graph of configurations (probabilistic road-map, rapidly exploring random trees) into an inefficient random process, where many useless nodes are sampled on parallel sub-manifolds. The total algorithmic complexity of classical contact planners comes from both the number of graph nodes sampled during exploration (ii) and the cost of the projection when sampling new configurations (i).

For this reason previous contributions having demonstrated acyclic contact locomotion on a real robot are too computationally expensive [4]. As a result at the DARPA Robotics Challenge, the participants stated that except for egress, the robots did not use multi contact strategies: they relied on unsafe bipedal walking to climb stairs, instead of using the provided handrails to facilitate the motion [5].

Our work aims at breaking the complexity of the acyclic contact planning problem. To do so we deal sequentially with the two main issues associated to our problem: the null measure of the contact manifold, and the combinatorics of the contact selection problem. First we introduce a low-dimensional space, called the *contact reachable* space, that can be sampled and mapped efficiently to the contact manifold. Then, given a path computed in the *contact reachable* space, we propose a deterministic algorithm to generate a contact sequence along the path. This decoupling presents pros and cons discussed in previous related literature, summarized in the following.

S. Tonneau, A. Del Prete and N. Mansard are with LAAS-CNRS / Université de Toulouse, France e-mail: (pro@stevetonneau.fr)

J. Pettré is with Inria, Rennes, France

C. Park and D. Manocha are with UNC, Chapel Hill, USA

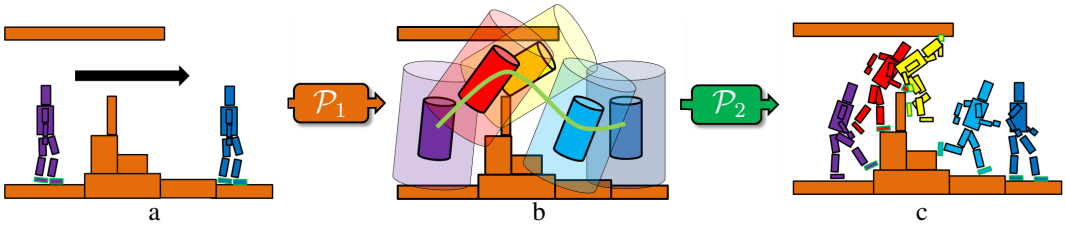

Fig. 1. Overview of our two-stage framework. Given a path request between start and goal positions (left image), $\mathcal{P}_1$ is the problem of computing a guide path in the space of *equilibrium feasible* root configurations. We achieve this by defining a geometric condition, the *reachability condition* (abstracted with the transparent cylinders on the middle image). $\mathcal{P}_2$ is then the problem of extending the path into a discrete sequence of contact configurations using an iterative algorithm (right image).

A. State of the art

Additionally to robotics, acyclic motion planning is also a problem of interest in neurosciences, biomechanics, and virtual character animation. Early contributions in the latter field rely on local adaptation of motion graphs [6], or ad-hoc construction of locomotion controllers [7]. These approaches are by definition not able to discover complex behaviors in unforeseen contexts.

The issue of planning acyclic contacts was first completely described by Bretl [4]. The issue requires the simultaneous handling of two sub-problems, $\mathcal{P}_1$: planning a guide path for the root of the robot in $SE(3)$; and $\mathcal{P}_2$: planning a discrete sequence of equilibrium configurations along the path. A third nontrivial problem, $\mathcal{P}_3$, then consists in interpolating a complete motion between two postures of the contact sequence. A key issue is to avoid combinatorial explosion when considering at the same time the possible contacts and the potential paths. Bretl's seminal paper proposes a first effective algorithm, able to handle simple situations (such as climbing scenarios), but not applicable to arbitrary environments. Following it, several papers have applied this approach in specific situations, limiting the combinatorial by imposing a fixed set of possible contacts [8], [9].

Most of the papers that followed the work of Bretl have explored alternative formulations to handle the combinatorics. Two main directions have been explored. **On the one hand, local optimization of both the root trajectory $\mathcal{P}_1$ and the contact positions $\mathcal{P}_2$** has been used, to trade the combinatorial aspect of the complete problem for a differential complexity, at the cost of local convergence [10]. A complete example of the potential offered by such approaches was proposed by [11] and successfully applied to a real robot [12]. To get reasonable computation times, the method uses a simplified dynamic model for the avatar. Still, the method is far from real-time (about 1 minute of computation for 20 contacts). A similar approach has been considered for manipulation by [13]. Deits and Tedrake proposed to solve contact planning globally as a mixed-integer problem, but only cyclic, bipedal locomotion is considered, and equilibrium is not considered [14]. Dai et al. [15] extended the work of Posa et al. [16] to discover the contact sequence for landing motions, but need to specify the contacts manually for complex interactions. In addition to the limits of the current implementations, optimization-based approaches only converge locally.

On the other hand, the two problems $\mathcal{P}_1$ and $\mathcal{P}_2$ might be decoupled to reduce the complexity. The feasibility and interest of the decoupling has been shown by Escande et al. [17] who manually set up a rough root guide path (i.e. an ad-hoc solution to $\mathcal{P}_1$), and then addressed $\mathcal{P}_2$ as the combinatorial computation of a feasible contact sequence in the neighborhood of the guide. A solution could then be found, but at the cost of prohibitive computation times (up to several hours) for constraining scenarios. This approach is suboptimal because the quality of the motion depends on the quality of the guide path. Bouyarmane et al. [18] precisely focused on automatically computing a guide path with guarantees of *equilibrium feasibility*, by extending key frames of the path into whole-body configurations in static equilibrium. Randomly-sampled configurations are projected to the contact manifold using an inverse-kinematics solver, a computationally-expensive process (about 15 minutes to compute a guide path in the examples presented). Moreover this explicit projection is insufficient to guarantee the feasibility between two key postures in the path. Chung and Khatib [19] also proposed a decoupled approach, with a planning phase based on the reachable workspace of the robot limbs, used to judge the ability to make contact with a discretized environment. This planning phase does not account for collisions, implying that re-planning is required in case of failure. In highly-constraining cases such as the car egress scenario we address, we believe that including collision constraints in the planning is a requirement [20], [21]. A limitation with these approaches (including our method) is that the existing planners only address a subset of the problem, because their ability to find a solution depends on the existence of quasi-static equilibrium configurations along a feasible path, which is too restrictive in the general case. Other contributions to legged locomotion, not directly related to multi-contact motion, are worth mentioning, as they rely on a similar decomposition of the problem. First a discrete sequence of contact sets is planned (at the so-called footstep planning phase), using a low-dimensional abstraction of the robot to account for its kinematic constraints [22], [23]. In [23], a "pose certificate" is obtained by generating a whole-body configuration for each set through inverse kinematics, as done in $\mathcal{P}_2$. Then a motion is generated along the sequence through the use of optimization techniques. The solutions proposed are designed for cyclic locomotion in quasi-flat scenarios, where the support polygon is a relevant method for equilibrium checks. They thus cannot be generalized to multi contact locomotion. However, some

contributions on specific parts of the problem could be applied directly in our case. For instance, learning "terrain costs", based on expert knowledge as proposed in [24], could define good heuristics to compute the next contact location for an effector. Although we did not try to include such formulation in this paper, it would be straightforward to integrate such heuristic in our planner.

As far as robotics applications are concerned, none of the existing multi-contact planners is *interactive*[1]. However, recent contributions to the interpolation between contact poses (problem $\mathcal{P}_3$) have brought promising preliminary solutions [25], [26], [27], [1]. In particular, our algorithm proposed in [1] is *interactive*. Therefore, a planner capable of efficiently solving $\mathcal{P}_1$ and $\mathcal{P}_2$ could outperform all existing planners if coupled with an interpolation method solving $\mathcal{P}_3$. The main contribution of this paper is exactly this planner.

B. Contributions

Our solution belongs to the class of decoupled approaches, i.e. we propose specific algorithms to efficiently solve both $\mathcal{P}_1$ and $\mathcal{P}_2$ while relying on state-of-the-art solution to $\mathcal{P}_3$ to obtain the whole movement. Our main contribution is the definition of a reduction property, the reachability condition.

Compared to previous approaches, our solution has two main novelties:

Regarding $\mathcal{P}_1$, we propose a fast guide path planning algorithm. The key to its efficiency is that it does not sample directly the contact manifold, but an approximation of the *contact reachable* space. The *contact reachable* space is a low-dimensional space for which there exists a mapping to the contact manifold.

Regarding $\mathcal{P}_2$, we propose a fast method to extend a *contact reachable* path into a sequence of whole-body configurations in static equilibrium. This requires the explicit computation of contact configurations. It is guided by dedicated heuristics that quickly synthesize feasible configurations.

The reachability condition is the key to the strict separation between $\mathcal{P}_1$ and $\mathcal{P}_2$, hence to the low complexity of our planner. However the reduction can result in failures. We demonstrate empirically its interest, through an extensive experimental validation with real robot models on a dynamic simulator. The high success rate and low computation times we obtain allow us to plan (and re-plan upon failure) multi-contact sequences at *interactive* rates.

To further demonstrate the validity of our approach, we show that the generated contact plans can be successfully executed (problem $\mathcal{P}_3$), either in simulation or on the real HRP-2 robot. For HRP-2, we detail the complete computation times to address sequentially the three problems, and compare them to related work, demonstrating that our method is orders of magnitude faster.

Finally, we provide an extensive discussion on the consequences of our approach in terms of efficiency and completeness regarding the contact planning problem.

[1]We define an interactive planner as one for which the time to plan a motion is in the same order of magnitude as the time to execute it. For instance, computing one contact change should take about one or two seconds.

Comparison with our previous work: The present paper is an extension of our ISRR conference paper [20]. As such, our solution to address $\mathcal{P}_1$ and $\mathcal{P}_2$ is the same motion planner as the one presented at ISRR (reformulated in Section IV and V). However three important novelties have been added to the planner: the pseudo-code of the algorithm (Section V-B2), a novel criterion for static equilibrium (Section VI), and the release of the source code of the planner (Section VII).

The other novelty of the paper is a rigorous experimental validation of the approach on actual robot models (Section VIII). To validate our contact plans we introduced a complete framework for multi-contact motion synthesis. This framework additionally comprises an interpolation method to solve the problem $\mathcal{P}_3$, based on a reformulation of our previous work [1]. Our solution to $\mathcal{P}_3$ allows us to verify that the synthesized motions are physically consistent, using our implementation of a state-of-the-art simulation algorithm [28]. These aspects of the framework are presented in details in the paper, but are not novel *per se*. The novelty lies in the complete validation of the contact planner with real robot models, for which the planning is much harder with respect to the avatars used in [20], because of more restrictive kinematic constraints.

II. Overview

Figure 1 illustrates our work flow. $\mathcal{P}_1$ and $\mathcal{P}_2$ are addressed sequentially: From a given problem (a) we first plan a root guide path (b), before extending it into a sequence of static equilibrium configurations (c). In the case where step (c) fails, our framework invalidates the computed guide path, and restarts the planning from (b).

A. Computation of a guide path — $\mathcal{P}_1$ (Section IV)

We first consider the problem of planning a root guide path (Figure 1–$\mathcal{P}_1$). The dimension of the path is equal to the number of degrees of freedom (DoFs) of the root of the robot. Similarly to previous work [18] the path must be *equilibrium feasible*: there must exist a joint configuration that results in static equilibrium for each root configuration[2]. Previous works verify *equilibrium feasibility* by explicitly computing such a configuration. We preserve the low dimensionality of the problem by approximating *equilibrium feasibility* with *contact reachability*, illustrated in the following.

An intuitive description of *contact reachable* configurations is "close, but not too close" to the environment: close, because a contact surface must be partially included in the reachable workspace of the robot to allow contact creation; not too close, because the robot must avoid collision. More precisely, a root configuration is *contact reachable* if the root scaled by a user-defined factor $s \geq 1$ is not in collision (Figure 2 - red shape), while the reachable workspace is in collision (Figure 2 - green shapes). To plan a root path, we then use an RRT planner where, instead of checking collision to validate a configuration, we verify the *reachability condition*.

[2]Enforcing static equilibrium is a classical, conservative approach to reduce the search space and considering states with non-zero accelerations and velocities, which can't be connected trivially. This does not mean that the final motion will necessary be quasi-static.

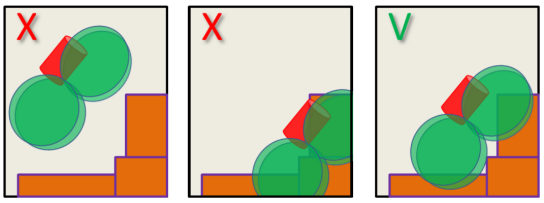

Fig. 2. The reachability condition is verified by the right configuration: the trunk (red) is free of collisions while the limbs reachable workspace (green) intersect the environment.

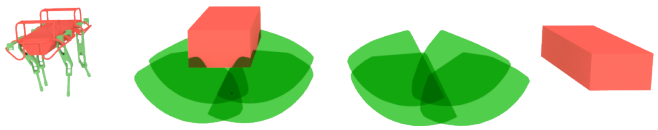

Fig. 3. Reachable workspace and torso bounding box of HyQ. Each green shape represent a reachable workspace W^k of a limb. The red shape is W^0.

In the remainder of this paper, we use the terms *contact reachable* and *equilibrium feasible* to qualify either a root or a whole-body configuration, or a set of such configurations.

B. Generating a discrete sequence of contact configurations — $\mathcal{P}_2$ (Section V)

The second stage extends the guide path into a sequence of contact configurations (Figure 1–$\mathcal{P}_2$). To achieve this the root path is first decomposed into a sequence of discrete root configurations, according to a user-defined discretization step. Each root configuration is then extended into a whole-body configuration in static equilibrium. The algorithm thus proceeds iteratively, starting from the whole-body initial configuration of the robot. It takes advantage of the fact that each root configuration is fixed to generate the contact by considering each limb individually.

III. Notation and Definitions

A vector $\mathbf{x}$ is denoted with a bold lower-case letter. A matrix $\mathbf{A}$ is denoted with a bold upper-case letter. A set C is denoted with an upper-case italic letter. Scalar variables and functions are denoted with lower-case italic letters, such as r or $f(\mathbf{x})$.

A robot is a kinematic tree R composed of: a root R^0, and l limbs $R^k, 1 \leq k \leq l$, attached to the root. The root has $r \geq 6$ DoFs: for instance, HRP-2 has two extra DoFs in the torso, such that we have $r = 8$. Thus R is fully described by a configuration (a vector of joint values) $\mathbf{q} \in \mathbb{R}^{r+n}$, with n the number of joint DoFs. $\mathbf{q}$ is decomposed as follows:

- $\mathbf{q}^k$ is a configuration of the limb R^k;
- $\mathbf{q}^{\overline{k}}$ is a vector of joint values of R **not** related to R^k. We define for convenience $\mathbf{q} = \mathbf{q}^k \oplus \mathbf{q}^{\overline{k}}$;
- $\mathbf{q}^0 \in \mathbb{R}^r$ is the world coordinates vector of R^0.

We then define a set of 3d volumes $W^i, 0 \leq i \leq l$, each attached to one joint of the root, such that $W^i(\mathbf{q}^0)$ describes the world position of W^i for the root configuration $\mathbf{q}^0$.
W^0 is a volume encompassing R^0 (Figure 3), or equal to it[3].
W^k is the reachable workspace of a limb R^k:

$$W^k = \left\{\mathbf{x} \in \mathbb{R}^3 : \exists\, \mathbf{q}^k \in C^k_{j.lim}, \mathbf{p}^k(\mathbf{q}^k) = \mathbf{x}\right\} \qquad (1)$$

where $\mathbf{p}^k$ denotes the end-effector position (in the root frame) of R^k (translation only) for $\mathbf{q}^0 = \mathbf{0}$ being the null displacement, and $C^k_{j.lim}$ is the space of admissible limb joint configurations. We also define $W = \bigcup_{k=1}^{l} W^k$.

[3] W^0 is typically a low-polygonal bounding shape of R^0 for performance.

The environment O is defined as the union of the obstacles O_i that it contains. O is represented as a polygon soup (or mesh), where the normal of each surface is known. No further requirement is needed by our approach. In this work, we assume the environment is fully known. State uncertainty is out of the scope of the paper.

Finally we define some relevant subsets of the configuration space C. $C_{Contact}$ is the set of whole body-configurations in contact and collision-free. $C^k_{Contact} \subset C_{Contact}$ is the set of whole body-configurations where at least R^k is in contact.

$C_{Equil} \subset C_{Contact}$ is the set of whole body-configurations in static equilibrium and collision-free.

For any set C_X, we define C^0_X:

$$C^0_X = \left\{\mathbf{q}^0, \exists \mathbf{q}^{\overline{0}} : \mathbf{q}^0 \oplus \mathbf{q}^{\overline{0}} \in C_X\right\}$$

IV. Root Path Planning in the Contact Reachable Space

During the root path planning we only consider the root configuration $\mathbf{q}^0$ defined in the previous Section, as well as the environment O.

Given start and goal configurations, we aim at computing a guide path $\mathbf{q}^0(t) : [0,1] \longrightarrow \mathbb{R}^r$ verifying:

$$\forall t \in [0,1], \mathbf{q}^0(t) \in C^0_{Equil}$$

This means that any root configuration must be extended into a whole-body, static equilibrium configuration. C^0_{Equil} cannot be described analytically.

The main hypothesis of this work is that for a large variety of locomotion tasks, we can define a space $C^0_{Reach} \simeq C^0_{Contact}$, such that

$$\forall t \in [0,1], \mathbf{q}^0(t) \in C^0_{Reach} \Rightarrow \mathbf{q}^0(t) \in C^0_{Equil} \qquad (2)$$

We call C^0_{Reach} the *contact reachable workspace*, and detail its construction in the following. The validity of this hypothesis is discussed in depth in Section IX.

A. Conditions for contact reachability

The contact reachable workspace is defined as a compromise between two necessary and a sufficient condition for contact creation.

necessary conditions: For a contact to be possible, an obstacle $O_i \subset O$ necessarily intersects the reachable workspace $W(\mathbf{q}^0)$ of the robot. Also the torso of the robot $W^0(\mathbf{q}^0)$ must

necessarily be collision-free. Therefore we can define an outer approximation $\mathcal{C}^0_{Nec} \supset C^0_{Contact}$ as:

$$\mathcal{C}^0_{Nec} = \{\mathbf{q}^0 : W(\mathbf{q}^0) \cap O \neq \emptyset \textbf{ and } W^0(\mathbf{q}^0) \cap O = \emptyset\} \quad (3)$$

sufficient condition: Similarly we can define an inner approximation $\mathcal{C}^0_{Suf} \subset C^0_{Contact}$ by considering a bounding volume B^{Suf} encompassing the whole robot in a given pose, except for the effector surfaces.

$$\mathcal{C}^0_{Suf} = \{\mathbf{q}^0 : W(\mathbf{q}^0) \cap O \neq \emptyset \textbf{ and } B^{Suf}(\mathbf{q}^0) \cap O = \emptyset\} \quad (4)$$

B. The compromising reachability condition

The ideal shape $B^*, W^0 \subset B^* \subset B^{Suf}$ would define a necessary **and** sufficient condition for contact creation. It would guarantee that any root configuration $\mathbf{q}^0 \in B^*$ would result in a contact configuration, while any $\mathbf{q}^0 \notin B^*$ could not. To our knowledge B^* has no explicit definition. Therefore, we approximate B^* to define the contact reachable space C^0_{Reach}.

We define W^0_s as the volume W^0 subject to a scaling transformation by a factor $s \in \mathbb{R}^+$. We then consider the spaces C^0_s

$$C^0_s = \{\mathbf{q}^0 : W(\mathbf{q}^0) \cap O \neq \emptyset \textbf{ and } W^0_s(\mathbf{q}^0) \cap O = \emptyset\} \quad (5)$$

The parametrization of s defines a trade-off: If $s = 1$, then $W^0_s = W^0$, such that $C^0_1 = \mathcal{C}^0_{Nec}$. By increasing s, the condition can become sufficient, but less and less necessary. Eq. 5 thus defines the *reachability condition*. We fix a value s^* for s and define $C^0_{Reach} = C^0_{s^*}$. The computation of s^* is detailed in Section VIII-C1. In Appendix A, we give a generic method to compute the W volumes appearing in the definition of C^0_{Reach}.

C. Computing the guide path in C^0_{reach}

C^0_{Reach} can be sampled efficiently thanks to Eq. 5, and can thus be used with any standard motion planner. Our current implementation uses the Bi-RRT planner [29] provided by the HPP software [30]. Our implementation is exactly the same as the pseudo-code of the original planner (which does not detail the configuration validation method). With respect to a "classic" implementation, the only difference is that instead of validating a configuration using collision detection, we validate it with the *reachability condition*.

This Section has presented a guide path planner for the geometric root of a robot, implemented as a low-dimensional sampling-based algorithm. Given start and goal configurations, it outputs a continuous path for the robot's root.

V. From a guide path to a discrete sequence of contact configurations ($\mathcal{P}_2$)

In the second phase, we compute a discrete sequence of static equilibrium configurations $\mathbf{Q}^{\overline{0}}$ given a root path $\mathbf{q}^0(t) : [0,1] \longrightarrow C^0_{Reach}$. This contact planner uses a contact generator, used to generate static equilibrium configurations. We first describe the contact planning algorithm, before describing the contact generator.

A. Definition of a contact sequence

In previous contributions [17], a contact plan is defined as a sequence of quasi-static equilibrium configurations for each contact phase. For instance, a walk cycle would be described by three key configurations: a double-support configuration, a single-support configuration (a contact is broken), and another double-support configuration (a contact is created). Our definition of contact plan differs: between two consecutive configurations we allow both a contact break and a contact creation—if they are on the same effector. In the previous example, our contact plan would simply consist of the two double-support configurations. This representation is sufficient to describe all the contact phases, because the single support phase is implicitely described. Furthermore it removes the need to compute a single-support quasi-static configuration as in the example. Indeed, there might be a case where no quasi-static solution exists for the single support phase (because of the environment), but there exists a dynamic motion connecting the two double support states. Such motion will be computed by our framework, because the quasi-static constraint is only required at the contact planning phase; as shown in the companion video, and explained in Appendix C, our framework is able to produce dynamic motions.

B. Contact planning algorithm

Starting from an initial whole-body configuration, we compute a sequence of whole-body configurations $\mathbf{Q}^{\overline{0}}$ along the root path $\mathbf{q}^0(t)$. We first give an intuition of the algorithm, before providing its complete pseudo-code.

1) Algorithm overview: First, the root path $\mathbf{q}^0(t)$ is discretized into a sequence of j key configurations:

$$\mathbf{Q}^0 = [\mathbf{q}^0_0; \mathbf{q}^0_i; ..., \mathbf{q}^0_{j-1}]$$

where $\mathbf{q}^0_0$ and $\mathbf{q}^0_{j-1}$ are the start and goal configurations. j depends on a user-defined variable, called the discretization step. It corresponds to the ratio between the length of the path $\mathbf{q}^0(t)$[4], and the number of configurations selected along it to create the contact configurations. Each root configuration of $\mathbf{Q}^0$ is then extended into a whole-body configuration such that:

- At most one contact is not maintained (*broken*) between two consecutive configurations.
- At most one contact is added between two consecutive configurations.
- Each configuration is in static equilibrium.
- Each configuration is collision-free.

2) :

a) Maintaining a contact in the sequence: If kinematically possible, a limb in contact at step $i-1$ remains in contact at step i (Figure 4). Otherwise the contact is broken and a collision-free configuration is assigned to the limb. If two or more contacts can't be maintained between two consecutive

[4]The length of the path is computed as the weighted 6D Euclidian distance travelled along the it, with a weight of 0.7 for the translation part, and 0.3 for the orientation part.

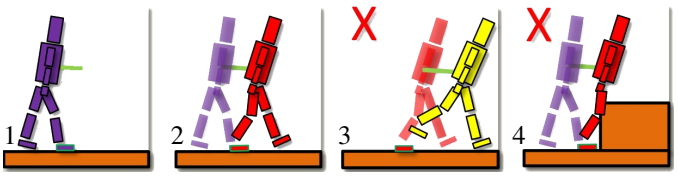

Fig. 4. Contacts are maintained if joint limits and collisions constraints are respected (2). They are broken otherwise(3,4). The green line represents the root path. The blurred character represents the previous contact configuration.

configurations, one or more intermediate configurations are added, to ensure that at most one contact is broken between two sequential configurations.

2) :

a) Creating contacts: Contacts are created using a FIFO approach: we try first to create a contact with the limb that has been contact-free the longest. If the contact creation does not succeeds, the limb is pushed on top of the queue, and will only be tried again after the others.

2) Pseudo-code of the Algorithm: First, we define an abstract structure State, that describes a contact configuration. The use of queues allows a FIFO approach regarding the order in which contacts are tested: we try to replace older contacts first when necessary. Thus the algorithm is deterministic even though it can handle acyclic motions.

```
Struct Limb
{
    // Limb Configuration
    Configuration qk;
    // Effector position in
    // world coordinates
    vector6 pk;
};

Struct State
{
    // root location
    Configuration q0;
    // List of limbs not in contact
    queue<Limb> freeLimbs;
    // List of limbs in contact
    queue<Limb> contactLimbs;
};
```

From the start configuration, given as an input by the user, we create the initial state $s0$. Algorithm 1 is then called with $s0$, as well as the discretized path $\mathbf{Q}^0$, as input parameters.

At each step, GENFULLBODY is called with the previous state as a parameter, as well as a new root configuration. GENFULLBODY returns a new contact configuration, if it succeeded in computing a configuration with only one contact switch occurring. Otherwise, the method INTERMEDIATECONTACTSTATE is called. It repositions one end effector (either a free limb, or the oldest active contact) towards a new contact position if possible. This repositioning allows to increase the odds that the contact can be maintained at the next step. Algorithm 2 gives the pseudo code for GENFULLBODY.

The method MAINTAINCONTACT$(pState, \mathbf{q}^0, k)$ performs inverse kinematics to reach the previous contact position for the Limb. If it succeeds, the new limb configuration is assigned to k. If it fails, a random collision free configuration is assigned to k.

The method ISINSTATICEQUILIBRIUM returns whether a given state is in static equilibrium.

Algorithm 1 Discretization of a path

1: **function** INTERPOLATE($s0$,$\mathbf{Q}^0$, MAX_TRIES)
2: $list$ <State> $states = [s0]$
3: $nb\_fail = 0$
4: $i = 1$; /*Current index in the list*/
5: **while** $i < length(\mathbf{Q}^0)$ **do**
6: State $pState = last\_element(states)$
7: State $s =$ GENFULLBODY$(pState, \mathbf{Q}^0[i])$
8: **if** $s! = NULL$ **then**
9: $nb\_fail = 0$
10: $i+ = 1$
11: **return** $\mathbf{q}^0$
12: **else**
13: $nb\_fail+ = 1$
14: **if** $nb\_fail == MAX\_TRIES$ **then**
15: **return** $FAILURE$
16: $s =$INTERMEDIATECONTACTSTATE$(pState)$
17: $push\_back(states, s)$
18: **return** $states$

Algorithm 2 Full body contact generation method

1: **function** GENFULLBODY($pState$,$\mathbf{q}^0$)
2: State $newState$
3: $newState.q0 = \mathbf{q}^0$
4: $newState.freeLimbs = pState.freeLimbs$
5: /*First try to maintain previous contacts*/
6: $nbContactsBroken = 0$
7: **for each** Limb k in $pState.contactLimbs$ **do**
8: **if** !MAINTAINCONTACT$(pState, \mathbf{q}^0, k)$ **then**
9: $nbContactsBroken+ = 1$
10: **if** $nbContactsBroken > 1$ **then**
11: **return** $NULL$
12: $push(newState.freeLimbs, k)$
13: **else**
14: $push(newState.contactLimbs, k)$
15: **for each** Limb k in $pState.freeLimbs$ **do**
16: **if** GENERATECONTACT$(\mathbf{q}^0, k)$ **then**
17: $push(newState.contactLimbs, k)$
18: $remove(newState.freeLimbs, k)$
19: **return** $newState$
20: **if** ISINSTATICEQUILIBRIUM$(newState)$ **then**
21: **return** $newState$
22: **else**
23: **return** $NULL$

Algorithm 3 Adds or repositions a contact for one limb

```
1: function INTERMEDIATECONTACTSTATE(state)
2:     i = 0
3:     while i < length(states.freeLimbs) do
4:         Limb k = pop(states.freeLimbs)
5:         if GENERATECONTACT(state.q0, k) then
6:             push(newState.contactLimbs, k)
7:             return
8:         else
9:             i+ = 1
10:            push(states.freeLimbs, k)
11:    i = 0
12:    while i < length(states.contactLimbs) do
13:        Limb k = pop(states.contactLimbs)
14:        Limb copy = k
15:        i+ = 1
16:        if GENERATECONTACT(state.q0, k) then
17:            push(newState.contactLimbs, k)
18:            return
19:        else
20:            push(newState.contactLimbs, copy)
       /*Fails if impossible to relocate any effector*/
21:    return FAILURE
```

The pseudo code for the method INTERMEDIATECONTACTSTATE is given by Algorithm 3.

GENERATECONTACT$(\mathbf{q}^0, k)$ is a call to the contact generator presented in the following Section V-C. It generates a contact configuration in static equilibrium, and assigns the corresponding configuration to k. If it fails, k remains unchanged if it is collision free, otherwise it is assigned a random collision free configuration.

C. Contact generator

Given a configuration of the root and the list of effectors that should be in contact, the contact generator computes the configuration of the limbs such that contacts are properly satisfied and the robot is in static equilibrium:

$$\mathbf{q}^{\overline{k}} \longrightarrow \mathbf{q}^k, (\mathbf{q}^k \oplus \mathbf{q}^{\overline{k}}) \in C_{Equil} \textbf{ and } \mathbf{q}^k \in C^k_{Contact} \quad (6)$$

In previous works [17], [18], the generation of contact is typically implemented by randomly sampling configurations and projecting the whole robot configuration onto the closest surfaces with an inverse kinematics solver. In case of failure of the projection, the process would randomly iterate.

We propose two modifications of this general algorithm principle. First our contact generator handles each limb R^k independently. By handling each limb separately, we reduce the complexity of the generation of contact configurations. This is made possible thanks to the reachability condition in $\mathcal{P}_1$ that produces a root path that we can afford not to modify in $\mathcal{P}_2$, and because we allow both a contact break and a contact creation between two consecutive configurations of the contact sequence. Second, we rely on off-line generation of configuration candidates.

We define $C^\epsilon_{Contact} \supset C_{Contact}$ as the set of configurations such that the minimum 3D distance between an effector and an obstacle is less than $\epsilon \in \mathbb{R}$. We then apply the following steps:

1) Generate off-line N valid sample limb configurations $\mathbf{q}^k_i, 0 \leq i < N$ (We choose $N = 10^4$);
2) Using the end-effector positions $\mathbf{p}(\mathbf{q}^k_i)$ as indices, store each sample in an octree data structure;
3) At runtime, when contact creation is required, intersect the octree and the environment[5] to retrieve the list of samples $S \subset C^\epsilon_{Contact}$ close to contact (Figure 5 (b) and (c));
4) Use a user-defined heuristic h to sort S;
5) If S is empty, stop (failure). Else select the first configuration of S. Project it onto contact using inverse kinematics. (Figure 5 (d) and (e));
6) If Eq. 6 is verified, stop (success). Otherwise remove the element from S and go to step 5.

Because the distance ϵ does not account for the variation in orientation, several samples of $C^\epsilon_{Contact}$ may turn out to be unfeasible at the time of projection. One could consider additionally filtering $C^\epsilon_{Contact}$ based on the orientation with respect to the obstacle normal, but in our experience we did not notice any significant improvement in the computational performances of the planner, so we do not perform this additional step.

In all our experiments, the heuristic h is implemented as a variation of a manipulability-based heuristic [31]. The manipulability is a real number that quantifies how "good" a configuration is to perform a given task, based on the analysis of the Jacobian matrix. With such heuristics, a configuration can be chosen because it is far from singularities, and thus allows mobility in all directions. On the contrary, it can be chosen because it is particularly efficient to exert a force in a desired direction. In our experiments, the former solution is usually chosen for computing leg contacts, while the latter is used for computing hand contacts. We recall the manipulability measure and its derivatives in Appendix B.

Finally, to verify that a configuration is in static equilibrium, we use a new robust LP formulation. It replaces the computationally inefficient double description approach used in our previous work [20], and presented in the following Section VI.

VI. A CRITERION FOR ROBUST STATIC EQUILIBRIUM

We first give a linear program (LP) that verifies whether a contact configuration allows for static equilibrium. This LP is the same that was proposed in [32]. From this formulation we derive a new LP that quantifies the robustness of the equilibrium to uncertainties in the contact forces. In turn, from this value we can either choose the most robust candidate, or set a threshold on the required robustness.

[5]this operation is achieved natively by the fcl library https://flexible-collision-library.github.io/

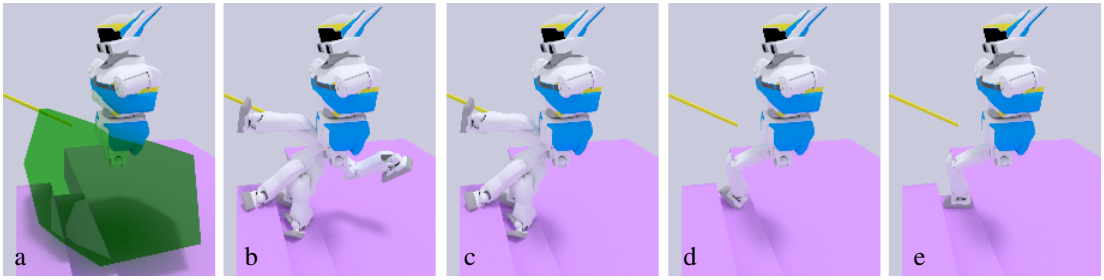

Fig. 5. Generation of a contact configuration for the right leg of HRP-2. (a): Selection of reachable obstacles. (b): Entries of the limb samples database (with $N = 4$). (c): With a proximity query between the octree database and the obstacles, configurations too far from obstacles are discarded. (d): The best candidate according to a user-defined heuristic h is chosen. (e): The final contact is achieved using inverse kinematics.

1) Conditions for static equilibrium: We first define the variables of the problem, for e contact points, expressed in world coordinates:

- $\mathbf{c} \in \mathbb{R}^3$ is the robot center of mass (COM);
- $m \in \mathbb{R}$ is the robot mass;
- $\mathbf{g} = [0, 0, -9.81]^T$ is the gravity acceleration;
- μ is the friction coefficient;
- for the i-th contact point $1 \leq i \leq e$:
  - $\mathbf{p_i}$ is the contact position;
  - $\mathbf{f_i}$ is the force applied at $\mathbf{p_i}$;
  - $\mathbf{n}_i, \boldsymbol{\gamma}_{i1}, \boldsymbol{\gamma}_{i2}$ form a local Cartesian coordinate system centered at $\mathbf{p_i}$. $\mathbf{n}_i$ is aligned with the contact surface normal, and the $\boldsymbol{\gamma}_i$s are tangent vectors.

According to Coulomb's law, the non-slipping condition is verified if all the contact forces lie in the friction cone defined by the surface. As classically done, we linearize the friction cone in a conservative fashion with a pyramid included in it, described by four generating rays of unit length. We choose for instance:

$$\mathbf{V}_i = \begin{bmatrix} \mathbf{n}_i + \mu\boldsymbol{\gamma}_{i1} & \mathbf{n}_i - \mu\boldsymbol{\gamma}_{i1} & \mathbf{n}_i + \mu\boldsymbol{\gamma}_{i2} & \mathbf{n}_i - \mu\boldsymbol{\gamma}_{i2} \end{bmatrix}^T$$

Any force belonging to the linearized cone can thus be expressed as a positive combination of its four generating rays.

$$\forall i \qquad \exists \boldsymbol{\beta}_i \in \mathbb{R}^4 : \boldsymbol{\beta}_i \geq 0 \text{ and } \mathbf{f}_i = \mathbf{V}_i \boldsymbol{\beta}_i,$$

where $\boldsymbol{\beta}_i$ contains the coefficients of the cone generators. We can then stack all the constraints to obtain:

$$\exists \boldsymbol{\beta} \in \mathbb{R}^{4e}, \boldsymbol{\beta} \geq 0 \text{ and } \mathbf{f} = \mathbf{V}\boldsymbol{\beta}, \tag{7}$$

where $\mathbf{V} = \text{diag}(\{\mathbf{V}_1, \ldots, \mathbf{V}_e\})$, and $\mathbf{f} = (\mathbf{f}_0, ..., \mathbf{f}_e)$.

From the Newton-Euler equations, to be in static equilibrium the contact forces have to compensate the gravitational forces:

$$\underbrace{\begin{bmatrix} \mathbf{I}_3 & \ldots & \mathbf{I}_3 \\ \hat{\mathbf{p}}_1 & \ldots & \hat{\mathbf{p}}_e \end{bmatrix}}_{\mathbf{G}} \mathbf{V}\boldsymbol{\beta}, = \underbrace{\begin{bmatrix} \mathbf{0}_{3\times3} \\ m\hat{\mathbf{g}} \end{bmatrix}}_{\mathbf{D}} \mathbf{c} + \underbrace{\begin{bmatrix} -m\mathbf{g} \\ \mathbf{0} \end{bmatrix}}_{\mathbf{d}} \tag{8}$$

where $\hat{\mathbf{x}} \in \mathbb{R}^{3\times3}$ is the cross-product matrix associated to $\mathbf{x}$.

If there exists a $\boldsymbol{\beta}^*$ satisfying (7) and (8), it means that the configuration is in static equilibrium. The problem can then be formulated as an LP:

$$\begin{aligned} \text{find} \quad & \boldsymbol{\beta} \in \mathbb{R}^{4e} \\ \text{subject to} \quad & \mathbf{G}\boldsymbol{\beta} = \mathbf{D}\mathbf{c} + \mathbf{d} \\ & \boldsymbol{\beta} \geq 0 \end{aligned} \tag{9}$$

2) Formulation of a robust LP: Let $b_0 \in \mathbb{R}$ be a scalar value. We now define the following LP:

$$\begin{aligned} \text{find} \quad & \boldsymbol{\beta} \in \mathbb{R}^{4e}, b_0 \in \mathbb{R} \\ \text{minimize} \quad & -b_0 \\ \text{subject to} \quad & \mathbf{G}\boldsymbol{\beta} = \mathbf{D}\mathbf{c} + \mathbf{d} \\ & \boldsymbol{\beta} \geq b_0\mathbf{1} \end{aligned} \tag{10}$$

We observe that if b_0 is positive then (9) admits a solution, and b_0 is proportional to the minimum distance of the contact forces to the boundaries of the friction cones. If b_0 is negative, the configuration is not in static equilibrium, and b_0 indicates "how far" from equilibrium the configuration is. We thus use b_0 as a measure of robustness. A simple approach to robustness consists in choosing a smaller friction coefficient, to constrain the forces to lie away from the boundaries of the real cone. However, this would result in a small safety margin for forces of low magnitude, and an excessively large safety margin for large forces as the boundaries grow more and more apart. In comparison, our margin b_0 is constant, and provides a helpful mean to compare the robustness of different contact configurations.

In our implementation, rather than solving directly (10), we solve an equivalent problem of smaller dimension that we get by taking the dual of (10) and eliminating the Lagrange multipliers associated to the inequality constraints:

$$\begin{aligned} \text{find} \quad & \boldsymbol{\nu} \in \mathbb{R}^6 \\ \text{maximize} \quad & -(\mathbf{D}\mathbf{c} + \mathbf{d})^T\boldsymbol{\nu} \\ \text{subject to} \quad & \mathbf{G}^T\boldsymbol{\nu} \geq 0 \\ & \mathbf{1}^T\mathbf{G}^T\boldsymbol{\nu} = 1 \end{aligned} \tag{11}$$

Indeed, from Slater's conditions [33], we know that the optimal values of an LP and its dual are equal. Therefore the optimal value $\boldsymbol{\nu}^*$ gives the optimal value b_0^* through the equality $b_0^* = (\mathbf{D}\mathbf{c} + \mathbf{d})^T\boldsymbol{\nu}^*$.

VII. Source Code of our Planner

Our planner is implemented using the Humanoid Path Planner (HPP) software, introduced in [30]. HPP is an open

source motion planning framework developed by the Gepetto team at LAAS-CNRS. HPP implements the standard tools and algorithms used in motion planning, such as the Bi-RRT planner from which RB-RRT is derived.

The robot models used in our experiments are described using the standard urdf file format, compatible with HPP.

Our implementation of the planner is also open source. Both HPP and our planner can be simply downloaded and compiled by following the instructions on https://humanoid-path-planner.github.io/hpp-doc/download.html?branch=rbprm.

VIII. Results

In this section we present some of the results obtained with our planner. The complete sequences are shown in the companion video. Specifically, we demonstrate the planner for two legged robots, in a large variety of environments: the humanoid HRP-2 and the quadruped HyQ.

Our contact plans are then interpolated with a dedicated solution to the interpolation problem $\mathcal{P}_3$. This allows us to validate the obtained motions in a dynamic simulator. This validation is an important contribution as it increases the confidence that the contact plans we compute can effectively result in feasible motions on the real robot. One motion is demonstrated on the real HRP-2 robot.

At the end of this section, we discuss the role of the parameters of our framework. We then provide the *interactive* computation times obtained in each case. We also compare the times obtained with HRP-2 with respect to previous works.

A. Experimental validation of the contact plans

To generate continuous movements from our contact plans we used either the framework proposed in [1], or our own implementation of a $\mathcal{P}_3$ solver (Appendix C). The resulting movements have been validated either on the real HRP-2 robot (details can be found in [1]), or with our dynamic simulator, based on a state-of-the-art algorithm [28]. In the simulations we controlled the robot with a standard inverse-dynamics controller [34]. The code source of the simulator is available at https://github.com/andreadelprete/pinocchio_inv_dyn/releases/tag/rbprm. This controller tries to follow the given whole-body trajectories, giving higher priority to the center-of-mass and end-effectors tracking with respect to the joint tracking. The controller also makes sure that the resulting contact forces lie inside the specified friction cones (we used a friction coefficient of 0.3), and that the joint position, velocity and torque limits are satisfied. The companion video shows the obtained motions.

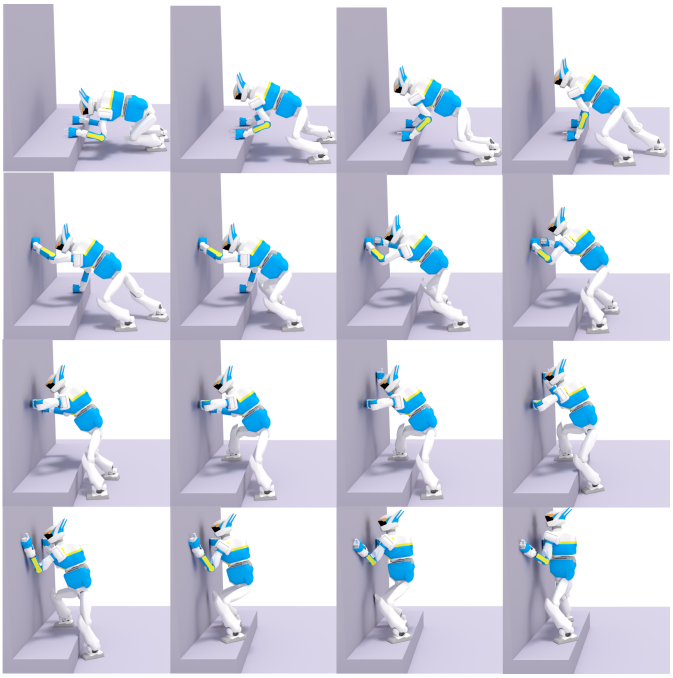

Fig. 6. HRP-2 in the standing scenario.

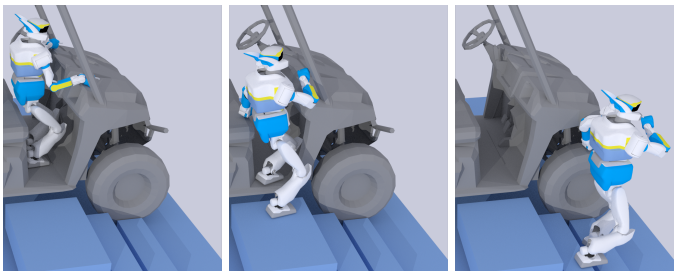

Fig. 7. Selected frames from the car egress scenario.

B. Description of the scenarios

In all the scenarios considered, the formulation of the problem is always the same: a start and goal root configuration are provided as input (except in the stair climbing scenario where the start whole body configuration is given). The framework computes the initial contact configuration, and outputs a sequence of contact configurations connecting it to the goal. In each scenario we detail the contacts involved and the heuristics chosen (either h_{EFORT}, h_{vel} or h_w, all of which are defined in the Appendix B).

1) :

1) HRP-2 – Standing up (Figure 6): From a bent configuration, the robot has to stand up using a wall as support, and climbing a 25-cm high step.
Contacts involved: All (both feet and hands).
Heuristics: h_w for the feet, h_{EFORT} for the hands.

2) HRP-2 – Car egress (Figure 7): In this scenario inspired from the DRC car egress HRP-2 has to step out of a car.
Contacts involved: All (both feet and hands).
Heuristics: h_w.

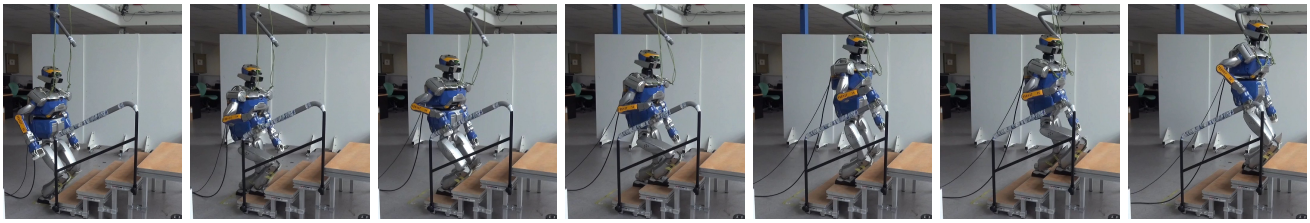

Fig. 8. HRP-2 in the stair climbing scenario.

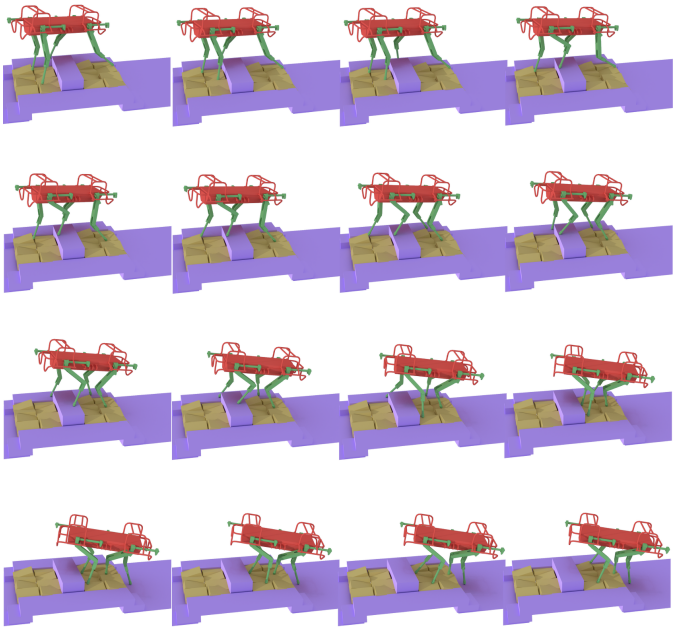

Fig. 9. Robust crossing of rubbles by HyQ.

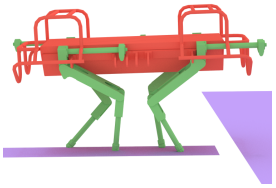

Fig. 10. HyQ crossing a narrow bridge.

3) HRP-2 – Staircase with high steps (Figure 8): The goal is to climb three 15-cm high steps.
Contacts involved: Feet and right arm.
Heuristics: The manipulability h_w is chosen for the feet; h_{EFORT} is chosen for the right arm.

4) HyQ – DRC-style rubble (Figure 9): The quadruped robot must cross a rubble composed of bricks rotated at different angles and directions.
Contacts involved: All (the 4 legs).
Heuristics: h_w for all legs. The robustness threshold b_0 is set to 20.

5) HyQ – Obstacle race (Figure 10 and 11): In this long scene, HyQ has to cross a 55-cm large hole, followed by a narrow "bridge", only 25-cm large.
Contacts involved: All (the 4 legs).

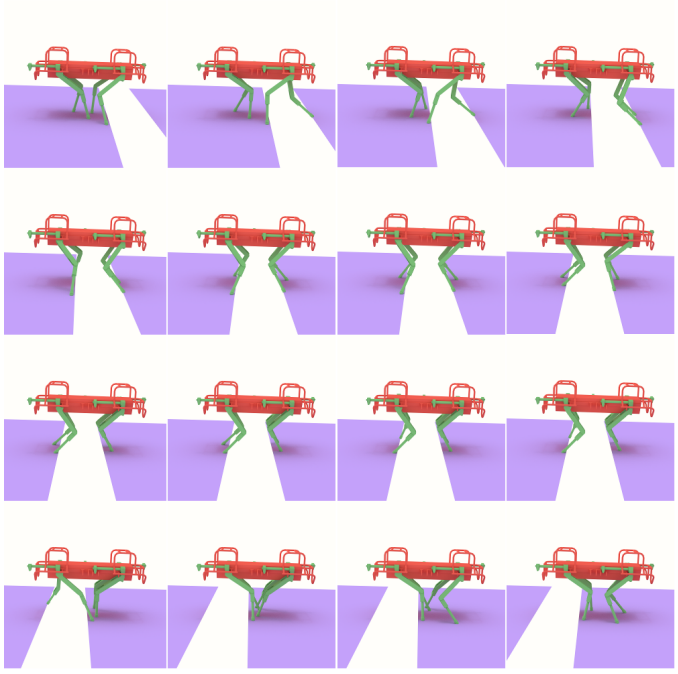

Fig. 11. Crossing a hole contact sequence for HyQ.

Heuristics: h_w for all legs. The robustness threshold b_0 is set to 10.

6) HRP-2 – Path re-planning (Figure 12): In this long scene, HRP-2 plans a path through several obstacles. The scene is edited during the execution of the motion: a stair is added, some stepping stones are removed, and part of the final staircase is deleted. All these modifications require re-planning.
Contacts involved: Feet and the right arm.
Heuristics: h_w for all legs. h_{EFORT} for the right arm. The robustness threshold is set to 2.

7) :

C. Role of the main parameters

We discuss the factors that influence the outcome of our planner: the root scaling factor s (Section IV-B), the heuristics for contact generation (Appendix B), and lastly, the discretization step for the guide path. The appropriate value for these parameters is computed empirically based on use-case analysis or trials and errors.

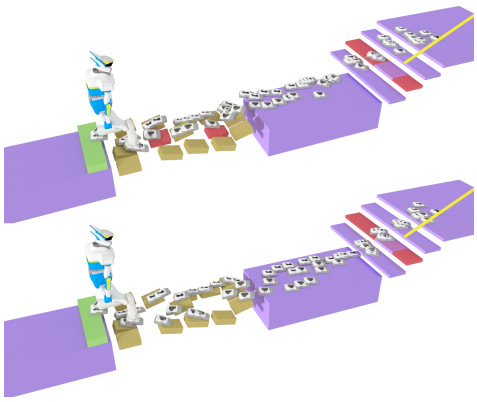

Fig. 12. HRP-2 in the re-planning scenario. After the red step stones are removed, a new sequence of contacts is re-planned. Hand contacts are not presented here for readability.

Value of s	Sensitivity	Specificity
1	76%	100 %
1.1	88%	96%
1.15	93%	94%
1.2	97%	92.5%
1.25	98%	91.7%
1.5	99%	90.5%

TABLE I

SENSITIVITY AND SPECIFICITY VALUES OF THE REACHABILITY CONDITION, DEPENDING ON THE SCALING VALUE s OF W^0.

	Path planning success rate
Stairs	100%
Standing	68%
Car	77%
Rubble	97%
Race	88.0%

TABLE II

PERCENTAGE OF SUCCESSFUL COMPLETE CONTACT PLANNING RATES FOR EACH SCENARIO, ROUNDED TO THE FIRST DECIMAL.

	Equilibrium success rate	Kinematic failure	Equilibrium failure
High stairs	99.5%	0.1%	0.4%
Standing up	87.8%	6.1%	6.1%
Car egress	66.2%	15.9%	17.9%
Rubble	97.54%	0.16%	2.3%
Obstacle race	92.4%	0.15%	7.45%

TABLE III

SUCCESS RATES OBTAINED FOR THE GENERATION OF STATIC EQUILIBRIUM CONTACT CONFIGURATIONS FOR EACH SCENARIO, ROUNDED TO THE FIRST DECIMAL. COLUMN 1 INDICATES INDICATES THE RATE OF CONTACT GENERATION THAT SUCCEEDED. IN THE CASES WHERE THE GENERATION FAILS, IT CAN BE EITHER A KINEMATIC ISSUE (COLUMN 2), OR BECAUSE NO CONTACT CONFIGURATION LED TO A STATIC EQUILIBRIUM CONFIGURATION (COLUMN 3). NOTE THAT A FAILURE IN THE CONTACT GENERATION IS NOT EQUIVALENT TO A FAILURE OF THE CONTACT PLANNING ALGORITHM.

1) Choosing the scaling factor s: For several values of s, we generated 10000 configurations. We then computed the sensitivity and specificity of the reachability condition. In this context, the sensitivity refers to the percentage of configurations in C^0_{Reach}, effectively belonging to $C^0_{Contact}$. If a sampled configuration is in C^0_{Reach}, but our method is unable to generate a contact configuration from it, as a result the sensitivity decreases. The sensitivity thus illustrates the confidence we have that any configuration in C^0_{Reach} will effectively lead to a contact configuration. Conversely, the specificity refers to the percentage of configurations **not** in C^0_{Reach}, effectively **not** belonging to $C^0_{Contact}$. If a sampled configuration is **not** in C^0_{Reach}, but our method is able to generate a contact configuration from it, as a result the specificity decreases. The specificity thus illustrates the confidence we have that all configurations that allow contact creation belong to C^0_{Reach} (or informally, the confidence that we are not missing valid solutions). We thus look for a compromise between sensitivity and specificity.

The obtained results for HRP-2 are shown in Table I, averaged over all scenes (except for the car egress: in this scenario, statistical tests are not really conclusive since we are only interested in a small area of the environment).

As it can be expected, the scaling results in a high increase of the sensitivity, with a decrease of the specificity. For HRP-2 we decided to set $s^* = 1.2$.

2) Choosing the heuristics: In our conference paper [20], the computed motions were generated using the EFORT heuristic. EFORT is designed for tasks requiring large magnitude contact forces (such as pushing / pulling / climbing). In locomotion tasks, such as the stair scenario, one issue with EFORT is that it tends to generate configurations close to singularities (and joint limits). While this did not significantly impact the generation of the plan, the resulting interpolation turned out to be harder. For this reason, we prefer to use our manipulability-based heuristic for the legs of the robot, but we still use EFORT for the arms, which results in fewer contact repositionings.

3) Discretization of the guide path: The discretization step is a user-defined, fixed parameter. The step has an influence on the output of the planner: if too large steps are taken, the planner may fail since we impose the constraint that only one contact change might occur between two consecutive steps. On the other hand, a small step will not impact the success rate of the planner, but may generate unnecessary states. In most scenarios the torso of HRP-2 moves about 15 cm between two postures, but only 3 cm for the car egress scenario to handle the geometry of the car. For future work, we would like to automatically adapt the size of the discretization step to the complexity of the environment.

D. Performance analysis

To analyze performance, we ran the planner 1000 times for each scenario. We measured the computation time spent in each part of the algorithm, and analyzed success rate.

1) Success rates (Table II): Despite the complexity of the scenarios and the approximations made in our formulation, our planner succeeded in the large majority of cases.

Table III presents the rate of successful contact generation. Note that a failure in contact generation for a root configuration is not equivalent to a failure in the contact plan. It simply

Scenario (nb steps)	Complete guide generation (ms)	Static equilibrium (ms)	Collision (ms)	Inverse Kinematics (ms)	Total generation time (ms)	Time per step (ms)
Stairs (18)	5 – **6** – 18	13 – **32** – 329	1 – **4** – 38	26 – **127** – 1345	92 – **261** – 2174	**15**
Standing (24)	65 – **1086** – 5227	27 – **144** – 338	2 – **12** – 37	144 – **1046** – 2374	371 – **2257** – 7671	**94**
Car (86)	320 – **6971** – 44002	409 – **1766** – 14752	297 – **1187** – 8483	3154 – **15323** – 165541	5834 – **31391** – 281000	**365**
Rubble (82)	37 – **573** – 1685	583 – **2714** – 9459	491 – **1971** – 6273	269 – **706** – 3118	1811 – **7195** – 23241	**86**
Race (134)	14 – **51** – 125	455 – **1359** – 21045	397 – **923** – 9924	228 – **471** – 5415	1436 – **3343** – 41446	**25**

TABLE IV

MINIMUM, **AVERAGE** AND WORST TIME (IN MS) SPENT IN THE GENERATION PROCESS FOR EACH SCENARIO AND EACH CRITICAL PART OF THE GENERATION PROCESS (NOT ALL PARTS ARE TIMED, THUS THE AVERAGE TOTAL COMPUTATION TIME IS HIGHER THAN THE SUM OF EACH PART). THE LAST COLUMN INDICATES THE AVERAGE TIME NECESSARY TO COMPUTE ONE CONTACT TRANSITION. THE COLLISION COLUMN TIMES INCLUDES THE (NEGLIGEABLE) OCTREE INTERSECTION OPERATION NECESSARY TO RETRIEVE THE CANDIDATE SAMPLES.

means that another limb was tested for contact generation for the same root configuration. As expected, a more constrained scenario such as the car egress provides less satisfying results, despite the high success rate of the planner.

2) Computation times (Table IV): For HRP-2, most of the time was spent performing inverse kinematics. This is not surprising considering the number of calls to the methods: IK projection is used intensively to maintain contact continuity between two postures; it is also applied every time a new candidate needs to be evaluated. In particular for the car egress scenario, the kinematic constraints are very demanding to avoid collisions.

On the other hand for HyQ most of the time is spent testing the static equilibrium of the candidate configurations.

In all scenarios, one can observe that the average computation time for one single step is largely below one second, thus allowing to consider *interactive* applications and online autonomous planning of the robot motion.

3) :

Conclusion: These results confirm that our approach provides a satisfying compromise between completeness and efficiency, thus enabling online planning while controlling the robot. Indeed, when the contact planning fails, it fails rapidly. This allows us to rapidly re-plan with a reasonable chance of success. The most efficient (and immediate) approach to obtain a valid contact plan as fast as possible would be to launch in parallel several instances of the planner (our current implementation is single-threaded) and to use any successful result as a plan for solver $\mathcal{P}_3$.

E.

E. Comparison with previous work

We did our best to provide a fair comparison of the computation complexity of our method with the state of the art. However existing benchmarks for motion planning algorithms [35] do not yet encompass contact planning. Moreover, the source code of the previous methods of the state of the art is often not available. Providing a fair comparison with the algorithms performing on the same computer and on the same scenarios is yet out of reach. A step in this direction is the open-source release of our source code (see Section VII) that allows any reader to reproduce our results. Furthermore, $\mathcal{P}_3$ remains challenging in the presence of obstacles. The only valid scenarios addressed completely in previous works are thus the stair-climbing scenarios of different heights proposed by Hauser in [8], and the table-egress scenario by Escande et al. in [17], which we consider to be of similar complexity with respect to the car-egress scenario (we did not consider the stairs in the scene). Both scenarios are tested with HRP-2.

Scenario	Method	Computation time
Stair 20 cm	Hauser [8]	5.42 min
	Mordatch et al.[11]	2 to 10 min
	Ours + [1]	$<$ **2s**
Stair 30 cm	Hauser [8]	4.08 min
	Mordatch et al.[11]	2 to 10 min
	Ours	$<$ **2s**
Stair 40 cm	Hauser [8]	10.08 min
	Mordatch et al.[11]	2 to 10 min
	Ours	$<$ **5s**
Table (car) egress	Bouyarmane et al. [18], [17]	3.5 hours
	Ours	$<$ **60 s**

TABLE V

COMPARISON BETWEEN THE COMPUTATION TIMES OBTAINED BY OUR METHOD AND PREVIOUS ONES FOR ADDRESSING THE WHOLE PROBLEM.

Table V presents the computation times for these scenarios, clearly demonstrating that our approach is order of magnitude faster than previous works.

IX. DISCUSSION: VALIDITY AND PURPOSE OF OUR CONTACT PLANNER

A.

As demonstrated in the results section, the main purpose of our method is the reduction of the algorithmic complexity of the problem, which leads to an interactive application. This property is critical for online applications with the robot and was not proposed by any of the previous contributions. Our method addresses highly constrained environments while improving the search time by orders of magnitude. This high performance is reached at the cost of some approximations that we discuss here.

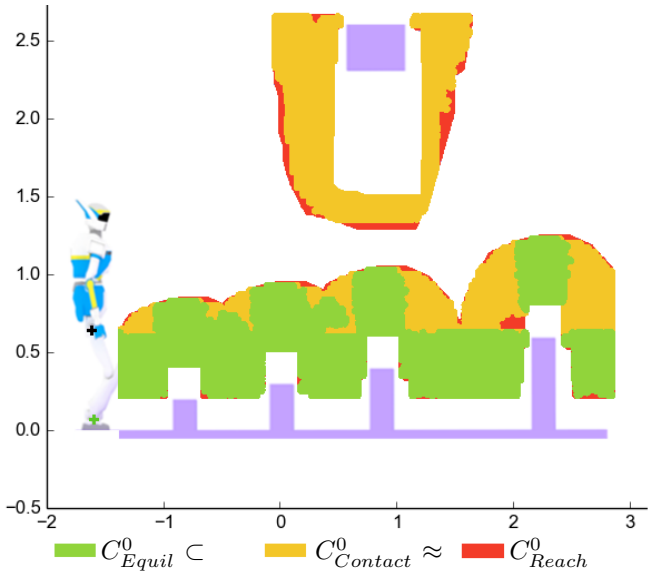

Fig. 13. Illustration of several root configurations sets used in this paper in a 2D scene. Obstacles are violet, and units are in meters. To show the sets in a 2D representation, all the rotational joints of HRP-2 are locked in the shown configuration, such that a torso configuration is only described by two positional parameters (x and y). The root of the robot is indicated with a black cross. To compute the reachable workspace, the point on the ankle indicated by a green cross was used. C^0_{Equil} is included in $C^0_{Contact}$. C^0_{Reach} approximates $C^0_{Contact}$. Depending on a parametrization, we can obtain $C^0_{Contact} \subset C^0_{Reach}$. Considering the configurations around the top obstacle, we can observe a similarity between C^0_{Equil} and $C^0_{Contact}$ when the reachable workspace of the legs includes *quasi-flat* surfaces.

The first approximation is the verification of *contact reachability* ($\mathbf{q}^0 \in C^0_{Contact}$). Our *reachability condition* ($\mathbf{q}^0 \in C^0_{Reach}$) is computationally efficient and provides an accurate approximation of $C^0_{Contact}$ (Section IV-B). This is demonstrated by the second column of Table III, and illustrated by Figure 13. Indeed, in the large majority of cases, (84% in the worst car egress case), we are able to find a contact configuration for any configuration in C^0_{Reach}.

Another source of computational cost identified in previous works is the verification of *equilibrium feasibility*. The main assumption of our work is that for the class of problems we consider *contact reachability* implies *equilibrium feasibility*. Our scenarios show that the assumption is verified in the majority of cases when at least one contact surface is *quasi flat* [32], that is when the friction cone of the contact surface contains the direction opposite to the gravity. Figure 13 illustrates this observation, demonstrated empirically by the third column of Table III. In the worst case, in our experiment the assumption was verified for 82% of the total amount of trials that verified *contact reachability*. In the example of [18], the verification of *equilibrium feasibility* implies a constructive demonstration by exhibiting a valid $\mathbf{q}^0$, requiring several minutes of planning. Our method, in comparison, takes from a few milliseconds to several seconds.

These results clearly justify our pragmatic approach.

A.

X. Conclusion

In this paper we consider the multi-contact planning problem, formulated as three sub-problems $\mathcal{P}_1$, $\mathcal{P}_2$, $\mathcal{P}_3$, addressed sequentially. While we propose a global framework that handles all these problems, our contribution focuses on $\mathcal{P}_1$ and $\mathcal{P}_2$. The first problem $\mathcal{P}_1$ consists in computing an *equilibrium feasible* guide path for the root of the robot; the second problem $\mathcal{P}_2$ is the computation of a discrete sequence of whole-body configurations along the root path. We believe that this decomposition is currently the most promising approach towards a global resolution of the problem. We also claim to have achieved a significant step towards this objective thanks to the dimensionality reduction provided by the reachability condition. With our results and the release of our source code, we hope to inspire further research in this direction.

Our contribution to $\mathcal{P}_1$ is the introduction of a low-dimensional space C^0_{reach}, an approximation of the space of *equilibrium feasible* root configurations. C^0_{reach} can be efficiently sampled and has a low-dimension. For these reasons we are able to solve $\mathcal{P}_1$ much faster than previous approaches.

Our contribution to $\mathcal{P}_2$ is a fast contact generation scheme that can optimize user-defined criteria.

Our results demonstrate that our method allows a pragmatic compromise between three criteria that are hard to reconcile: generality, performance, and quality of the solution, making it the first acyclic contact planner compatible with *interactive* applications.

Regarding generality, the *reachability condition*, coupled with an approach based on limb decomposition, allows the method to address automatically arbitrary legged robots. **Regarding performance**, our framework is efficient in addressing both $\mathcal{P}_1$ and $\mathcal{P}_2$. This results in *interactive* computation times. **Regarding the quality of the paths**, we are able to compute *equilibrium feasible* paths in all the presented scenarios, with high success rates. As for [18], failures can still occur, due to the approximate condition used to compute the guide path. The low computational burden of our framework however allows for fast re-planning in case of failure. Furthermore, because of this approximation, the guide search is not complete. The choice is deliberate, because we believe that it is necessary to trade completeness for efficiency at all stages of the planner. However, one direction for future work is to focus on a more accurate formulation of C^0_{reach} to improve the approximation.

Our method applies to any scenario where at least one contact friction cone contains the direction opposed to the gravity (i.e. *quasi-flat*). This class of scenarios include all the problems proposed at the DARPA Robotics Challenge. One way to further extend its range of application, which we consider for future work, is to include the equilibrium criterion when solving $\mathcal{P}_1$. Considering the set of obstacles intersecting with the reachable workspace for a given root configuration as candidate surfaces, we can use them to verify the equilibrium criterion. This would give us a necessary condition for *equilibrium feasibility*.

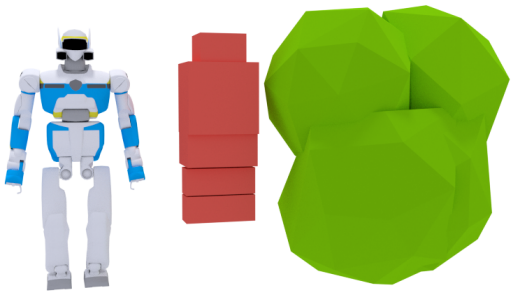

Fig. 14. The W volumes computed for HRP-2. The red shapes are W^0. The green shapes represent the W^k.

Fig. 15. Different approximations of the range of motion of the right arm of HRP-2. Left: non convex-hull, computed with the powercrust algorithm [37]. Middle: convex hull of the reachable workspace. Right: Simplified hull used in our experiments.

While we have exhibited complete multi-contact locomotion obtained with our contact planner, our main concern for future work is to address the interpolation between contact sequences ($\mathcal{P}_3$), which remains an open issue in highly-constrained scenarios. Solving $\mathcal{P}_3$ requires addressing efficiently the collision avoidance problem in the interpolation phase, an issue not addressed by existing frameworks. We aim at providing our plans with transition certificates, that would define constraints on $\mathcal{P}_3$, under which the transition between two contact configurations is feasible and collision-free. Finally, we aim at performing kinodynamic planning to remove the constraint that configurations be in static equilibrium. We believe that the most promising direction in this regard is to integrate the notion of Admissible Velocity Propagation [36]. Addressing these two issues is essential to bridge the gap between the planning and control aspects of legged locomotion.

APPENDIX A
GENERATING THE W VOLUMES FOR HRP-2

We detail our method to generate the volumes W used in RB-RRT, with the example of HRP-2. The kinematic tree is split into four limbs R^k. The arms are connected to the shoulders, and the legs to the root. The obtained volumes W are shown in Figure 14.

A. Step 1: computing the reachable workspace W^k of a limb

To generate a volume W^k, we proceed as follows:

1) Generate randomly N valid limb configurations for R^k, for N really large (say 100000);
2) For each configuration, store the 3D position of the end effector joint relatively to the root of R^k; then compute the convex hull of the resulting point cloud;
3) The resulting polytope can contain a very large number of faces. A last step is thus to simplify it with the blender decimate tool (http://wiki.blender.org/index.php/Doc:2.4/Manual/Modifiers/Generate/Decimate). This tool removes a user-defined amount of vertices (and faces) of the polytope, thus resulting in a convervative approximation of the original shape. For HRP-2 we apply the operator with a ratio of 0.06, resulting in a polytope of 38 faces for the arms and the legs.

Figure 15 illustrate the obtained W^k for HRP-2. Regarding the procedure, we can see that step 2 is conservative (Figure 15–right), which is acceptable, especially because the lost set essentially relates to configurations close to singularity (they are close to the boundaries of the reachable workspace, and often not *contact reachable*, as illustrated in Figure 13, where the exterior boundaries of the reachable workspace appear red, thus not belonging to $C^0_{Contact}$). We choose again to be less complete but more efficient, regarding the number of collision tests to be performed by RB-RRT. In step 1 on the other hand, selecting the convex hull (Figure 15–middle) instead of a minimum encompassing shape (Figure 15–left) may introduce false positives. Concretely, because the false positive set intersects with W^0, the scaling volume of the robot torso, the induced error is compensated, as verified by the results shown by Table III.

B. Step 2: computing the torso scaling workspace W^0 of the robot

To define the volume W^0 of HRP-2, we proceed in an empirical manner. First, we compute the bounding boxes of the robot torso, head, and upper legs (Figure 14 – red shapes). Then, we perform a scaling of these boxes by a factor s. The higher s is, the more likely sampled configurations are to be feasible, but the less complete is the approach. To compute the appropriate value of s, we proceed as described in Section VIII-C1, and choose empirically $s^* = 1.2$ as the appropriate value for HRP-2.

APPENDIX B

A.

APPENDIX B
MANIPULABILITY-BASED HEURISTICS FOR CONTACT SELECTION

This Section proposes two heuristics to select a contact that optimizes desired capabilities. For instance, one can be interested in configurations that allow to efficiently exert a force in the global direction of motion, or to stay away from

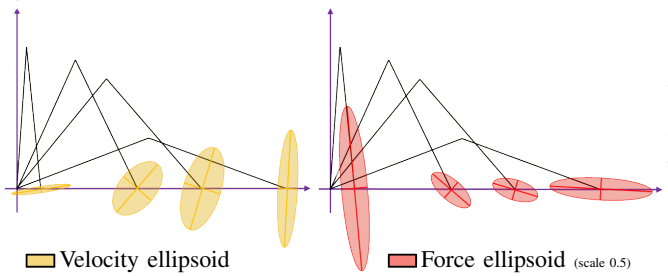

Fig. 16. Examples of velocity and force ellipsoids for a manipulator composed of 2 DoFs and 2 segments. Only the horizontal and vertical speeds are shown (not the rotation speeds).

singular configurations. We derive these heuristics from the work by [31], recalled here.

1) The force and velocity ellipsoids: We consider: a limb configuration $\mathbf{q}^k$; its end effector position $\mathbf{p}^k$; its Jacobian matrix $\mathbf{J}^k(\mathbf{q}^k)$; a force $\mathbf{f}$ exerted by the end effector. For clarity in the rest of the section we omit the k indices and write $\mathbf{J}^k(\mathbf{q}^k)$ as $\mathbf{J}$. Yoshikawa [31] defines the velocity(12) and force (13) ellipsoids:

$$\dot{\mathbf{p}}^T(\mathbf{J}\mathbf{J}^T)^{-1}\dot{\mathbf{p}} \leq 1 \tag{12}$$

$$\mathbf{f}^T(\mathbf{J}\mathbf{J}^T)\mathbf{f} \leq 1 \tag{13}$$

They describe the set of end-effector velocities (respectively forces) that can be reached under the constraint $||\dot{\mathbf{q}}||^2 \leq 1$ for the current configuration. The longer the axis of the ellipsoid is, the more important the velocity (resp. force) of the end-effector the direction of the axis can be (Figure 16).

2) Manipulability-based heuristics: From these definitions, we can derive two useful heuristics, that all account for the environment and the task being performed. The first one, EFORT, was introduced by [38]; the second one derives the manipulability measure proposed by Yoshikawa [31].

With EFORT, we define the efficiency of a configuration as the ability of a limb to exert a force in a given direction. We thus consider the force ellipsoid as a basis for our heuristic. In a given direction ρ, the length of the ellipsoid is given by the force-transmission ratio [39]:

$$f_T(\mathbf{q}, \rho) = [\rho^T(\mathbf{J}\mathbf{J}^T)\rho]^{-\frac{1}{2}}$$

In our problem, to compare candidate configurations, we include the quality of the contact surface, and choose ρ as the direction opposite to the local motion (given by the difference between two consecutive root positions):

$$h_{EFORT}(\mathbf{q}, \rho) = [\rho^T(\mathbf{J}\mathbf{J}^T)\rho]^{-\frac{1}{2}}(\mu \mathbf{n}^T\rho) \tag{14}$$

where μ and $\mathbf{n}$ are respectively the friction coefficient and the normal vector of the contact surface.

[]

h_{EFORT} will favor contacts that allow large efforts. $EFORT$ in particular is useful for tasks such as standing up, pushing / pulling. In other less demanding cases, manipulability can also be considered to avoid singularities. To do so, we can consider the manipulability measure h_w, also given by Yoshikawa [31]:

$$h_w(\mathbf{q}) = \sqrt{det(\mathbf{J}\mathbf{J}^T)} \tag{15}$$

h_w measures the "distance" of a given configuration to singularity. When h_w is equal to 0, the configuration is singular; the greater h_w is, the further away the configuration is from singularity.

A.

1) :

-
-
-
-
-
-
-
-

1) :

APPENDIX C
ADDRESSING $\mathcal{P}_3$

The stair climbing and the standing up scenarios were validated with the trajectory optimization scheme provided in [1]. To address the other scenarios, we propose a new implementation, entirely open source (https://github.com/stonneau/python_sandbox/releases/tag/tro_paper), which can be integrated directly in our motion planner software. This formulation uses the center of mass acceleration and angular momentum as input variables, while previously the contact forces were used. The center-of-mass trajectory resulting from the optimization is then turned into a collision-free whole-body trajectory.

We rewrite Eq. 8 in the general case:

$$\mathbf{G}\boldsymbol{\beta} = \underbrace{\begin{bmatrix} m(\ddot{\mathbf{c}} - \mathbf{g}) \\ m\mathbf{c} \times (\ddot{\mathbf{c}} - \mathbf{g}) + \dot{L} \end{bmatrix}}_{\mathbf{w}} \tag{16}$$

where $\dot{L}$ is the angular momentum expressed at the com. Eq. 16 defines a 6-dimensional cone $\mathcal{K}$ [40], [41]. For a given set of contacts, this cone determines all the admissible wrenches $\mathbf{w}$ that can be generated by contact forces inside their friction cones. The face form of $\mathcal{K}$ can be computed using the double description method [42], resulting in the following linear inequalities:

$$\mathcal{K} = \{\mathbf{w}, \mathbf{A}\mathbf{w} \leq \mathbf{b}\} \tag{17}$$

The objective is then to plan a trajectory for the center of mass such that the generated $\mathbf{w}$ always verifies Eq. 17. We now consider two contact configurations $\mathbf{q}_0$ and $\mathbf{q}_1$ computed by our planner: in the general case one contact is broken and one created to get from $\mathbf{q}_0$ to $\mathbf{q}_1$. We manually define the duration of each of the three contact phases. In each phase s the centroidal wrench $\mathbf{w}$ is constrained to lie inside a cone $\mathcal{K}_u, u = 0 \dots 2$. We call the total duration of the motion T, and formulate the following optimization problem:

$$\begin{aligned}
\underset{\ddot{\mathbf{c}}(t), \dot{L}(t)}{\text{minimize}} \quad & \sum_{u=0}^{2} \int_{t_u}^{t_u + \Delta t_u} \ell(\ddot{\mathbf{c}}(t), \dot{L}(t)) \mathrm{dt} \\
\text{subject to} \quad & \mathbf{A}_u \mathbf{w}(t) \leq \mathbf{b}_u, \forall t \in [t_u, t_u + \Delta t_u[\,, \forall u \\
& \mathbf{Y}_u \mathbf{c}(t) \leq \mathbf{y}_u, \forall t \in [t_u, t_u + \Delta t_u[\,, \forall u \\
& \mathbf{c}(0) = \mathbf{c}_{\mathbf{q}_0} \\
& \mathbf{c}(T) = \mathbf{c}_{\mathbf{q}_1} \\
& \mathbf{c}(0) = \dot{\mathbf{c}}(0) = \ddot{\mathbf{c}}(0) = \mathbf{0} \\
& \mathbf{c}(T) = \dot{\mathbf{c}}(T) = \ddot{\mathbf{c}}(T) = \mathbf{0}
\end{aligned} \tag{18}$$

The cost function ℓ is a weighted sum of the angular momentum and center-of-mass acceleration variation over the whole trajectory. The center-of-mass positions and velocities $\mathbf{c}(t)$, $\dot{\mathbf{c}}(t)$ are internal variables, obtained through the double integration of $\ddot{\mathbf{c}}(t)$. Then $\mathbf{w}(t)$ is obtained directly from these variables. $\mathbf{c}_{\mathbf{q}_0}$ and $\mathbf{c}_{\mathbf{q}_1}$ are the center-of-mass positions for configurations $\mathbf{q}_0$ and $\mathbf{q}_1$ respectively. $\mathbf{Y}_u$ and $\mathbf{y}_u$ denote stacked kinematic constraints on the center of mass position, determined by the active contact locations. The inequalities for each contact are determined in the same way that the reachable workspace is computed in Appendix A, with the effector serving as root.

This formulation can trivially be extended over the whole contact sequence. In our implementation, the problem is discretized using time steps of 100 ms.

The output of this optimization problem is an admissible center-of-mass trajectory. To compute the whole body motion, we use a two-step approach.

First, we plan a kinematic motion for the robot, subject to the contact constraints. We also constrain the center of mass to follow the computed trajectory. This is achieved using a constraint-based RRT planner [30]. As a result we obtain a collision-free whole-body motion.

The entire resolution takes approximatively 1.5 seconds for a complete contact transition.

Appendix D

Appendix D

Acknowledgements

This research is supported by Euroc (project under FP7 Grant Agreement 608849); Entracte (ANR grant agreement 13-CORD-002-01); the ARO Contract W911NF-14-1-0437; and the NSF award 1305286.

References

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

[2] J. Pratt, J. Carff, S. Drakunov, and A. Goswami, "Capture Point: A Step toward Humanoid Push Recovery," *2006 6th IEEE-RAS International Conference on Humanoid Robots*, 2006.

[3] T. Siméon, J. Laumond, J. Cortes, and A. Sahbani, "Manipulation planning with probabilistic roadmaps," *The Int. Journal of Robot. Research (IJRR)*, vol. 23, no. 7-8, pp. 729–746, 2004.

[4] T. Bretl, "Motion planning of multi-limbed robots subject to equilibrium constraints: The free-climbing robot problem," *The Int. Journal of Robot. Research (IJRR)*, vol. 25, no. 4, pp. 317–342, Apr. 2006.

[5] 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.

[6] L. Kovar, M. Gleicher, and F. Pighin, "Motion graphs," in *ACM Trans. Graph.*, vol. 21, 2002, pp. 473–482.

[7] J. Pettré, J.-P. Laumond, and T. Siméon, "A 2-stages locomotion planner for digital actors," in *Proc. of the 2003 ACM SIGGRAPH/Eurographics symp. on Comp. animation*, Granada, Spain, 2003, pp. 258–264.

[8] K. Hauser, T. Bretl, K. Harada, and J.-C. Latombe, "Using motion primitives in probabilistic sample-based planning for humanoid robots." in *WAFR*, ser. Springer Tracts in Advanced Robot., S. Akella, N. M. Amato, W. H. Huang, and B. Mishra, Eds., vol. 47. Springer, 2006.

[9] M. Stilman, "Global Manipulation Planning in Robot Joint Space With Task Constraints," *IEEE Trans. on Robot.*, vol. 26, no. 3, Jun. 2010.

[10] K. Yunt and C. Glocker, "Trajectory optimization of mechanical hybrid systems using sumt," in *9th IEEE International Workshop on Advanced Motion Control, 2006.*, 2006, pp. 665–671.

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

[12] I. Mordatch, K. Lowrey, and E. Todorov, "Ensemble-CIO: Full-Body Dynamic Motion Planning that Transfers to Physical Humanoids," in *Proc. of IEEE Int. Conf. on Robot. and Auto (ICRA)*, Seattle, USA, 2015.

[13] M. Gabiccini, A. Artoni, G. Pannocchia, and J. Gillis, "A computational framework for environment-aware robotic manipulation planning," in *Int. Symp. Robotics Research (ISRR)*, Sestri Levante, Italy, 2015.

[14] R. Deits and R. Tedrake, "Footstep planning on uneven terrain with mixed-integer convex optimization," in *Humanoid Robots (Humanoids), 14th IEEE-RAS Int. Conf. on*, Madrid, Spain, 2014.

[15] H. Dai, A. Valenzuela, and R. Tedrake, "Whole-body motion planning with centroidal dynamics and full kinematics," in *Humanoid Robots (Humanoids), 14th IEEE-RAS Int. Conf. on*, Madrid, Spain, 2014, pp. 295–302.

[16] M. Posa, C. Cantu, and R. Tedrake, "A direct method for trajectory optimization of rigid bodies through contact," *The Int. Journal of Robot. Research (IJRR)*, vol. 33, no. 1, pp. 69–81, Jan. 2014.

[17] A. Escande, A. Kheddar, S. Miossec, and S. Garsault, "Planning Support Contact-Points for Acyclic Motions and Experiments on HRP-2," in *ISER*, ser. Springer Tracts in Advanced Robot., O. Khatib, V. Kumar, and G. J. Pappas, Eds., vol. 54. Springer, 2008, pp. 293–302.

[18] K. Bouyarmane, A. Escande, F. Lamiraux, and A. Kheddar, "Potential field guide for humanoid multicontacts acyclic motion planning," in *Proc. of IEEE Int. Conf. on Robot. and Auto (ICRA)*, Kobe, Japan, 2009, pp. 1165 – 1170.

[19] S.-Y. Chung and O. Khatib, "Contact-consistent elastic strips for multi-contact locomotion planning of humanoid robots," in *Proc. of IEEE Int. Conf. on Robot. and Auto (ICRA)*, May 2015, pp. 6289–6294.

[20] 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.

[21] M. X. Grey, A. D. Ames, and C. K. Liu, "Footstep and motion planning in semi-unstructured environments using possibility graphs," in *ICRA'17. IEEE International Conference on Robotics and Automation, 2017 (Submitted)*. IEEE, 2017, available at https://arxiv.org/pdf/1610.00700v1.pdf.

[22] N. Perrin, O. Stasse, L. Baudouin, F. Lamiraux, and E. Yoshida, "Fast humanoid robot collision-free footstep planning using swept volume approximations," *IEEE Transactions on Robotics*, vol. 28, no. 2, pp. 427–439, April 2012.

[23] M. Zucker, N. Ratliff, M. Stolle, J. Chestnutt, J. A. Bagnell, C. G. Atkeson, and J. Kuffner, "Optimization and learning for rough terrain legged locomotion," *The International Journal of Robotics Research*, vol. 30, no. 2, pp. 175–191, 2011. [Online]. Available: https://doi.org/10.1177/0278364910392608

[24] N. D. Ratliff, D. Silver, and J. A. Bagnell, "Learning to search: Functional gradient techniques for imitation learning," *Auton. Robots*, vol. 27, no. 1, pp. 25–53, Jul. 2009. [Online]. Available: http://dx.doi.org/10.1007/s10514-009-9121-3

[25] K. Hauser, "Fast interpolation and time-optimization with contact," *The Int. Journal of Robot. Research (IJRR)*, vol. 33, no. 9, pp. 1231–1250, Aug. 2014.

[26] A. Herzog, N. Rotella, S. Schaal, and L. Righetti, "Trajectory generation for multi-contact momentum-control," in *Humanoid Robots (Humanoids), 15h IEEE-RAS Int. Conf. on*, Nov. 2015.

[27] C. Park, J. S. Park, S. Tonneau, N. Mansard, F. Multon, J. Pettré, and D. Manocha, "Dynamically balanced and plausible trajectory planning for human-like characters," in *To appear in Proc. of I3D '16*, Seatle, USA, 2016.

[28] D. M. Kaufman, S. Sueda, D. L. James, and D. K. Pai, "Staggered projections for frictional contact in multibody systems," *ACM Transactions on Graphics*, vol. 27, no. 5, p. 1, 2008.

[29] S. LaValle and J. Kuffner, J.J., "Randomized kinodynamic planning," in *Proc. of IEEE Int. Conf. on Robot. and Auto (ICRA)*, vol. 1, Detroit, Michigan, USA, 1999, pp. 473–479 vol.1.

[30] J. Mirabel, S. Tonneau, P. Fernbach, A. K. Seppl, M. Campana, N. Mansard, and F. Lamiraux, "Hpp: A new software for constrained motion planning," in *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Oct 2016, pp. 383–389.

[31] T. Yoshikawa, "Manipulability of robotic mechanisms," *The Int. Journal of Robot. Research (IJRR)*, vol. 4, no. 2, pp. 3–9, 1985.

[32] A. Del Prete, S. Tonneau, and N. Mansard, "Fast Algorithms to Test Robust Static Equilibrium for Legged Robots," in *To appear in Proc. of IEEE Int. Conf. on Robot. and Auto (ICRA)*, Stockholm, Sweden, 2016.

[33] S. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge University Press, 2004.

[34] A. Del Prete and N. Mansard, "Robustness to Joint-Torque Tracking Errors in Task-Space Inverse Dynamics," *IEEE Transaction on Robotics*, vol. 32, no. 5, pp. 1091 – 1105, 2016.

[35] M. Moll, I. A. Sucan, and L. E. Kavraki, "An extensible benchmarking infrastructure for motion planning algorithms," *arXiv preprint arXiv:1412.6673*, 2014.

[36] Q. Pham, S. Caron, and Y. Nakamura, "Kinodynamic planning in the configuration space via admissible velocity propagation," in *Robotics: Science and Systems IX (RSS)*, Berlin, Germany, 2013.

[37] N. Amenta, S. Choi, and R. K. Kolluri, "The power crust," in *Proc. of Sixth ACM Symposium on Solid Modeling and Applications (SMA)*, Ann Arbor, Michigan, USA, 2001, pp. 249–266.

[38] S. Tonneau, J. Pettré, and F. Multon, "Using task efficient contact configurations to animate creatures in arbitrary environments," *Computers & Graphics*, vol. 45, no. 0, 2014.

[39] S. Chiu, "Control of redundant manipulators for task compatibility," in *Proc. of Int. Conf. on Robot. and Auto (ICRA)*, vol. 4, 1987, pp. 1718–1724.

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

[41] 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.

[42] K. Fukuda and A. Prodon, *Double description method revisited*. Berlin, Heidelberg: Springer Berlin Heidelberg, 1996, pp. 91–111.