An efficient acyclic contact planner for multiped robots

The International Journal of Robotics Research
XX(X):1–19

DOI: 10.1177/ToBeAssigned
ijr.sagepub.com/

Steve Tonneau[1], Andrea Del Prete[1], Julien Pettré[2], Chonhyon Park[3], Dinesh Manocha[3], Nicolas Mansard[1]

Abstract

We present a contact planning framework 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 highlighted at the Darpa Robotics Challenge, where multi-contact behaviors could not be demonstrated.

Indeed, current planners suffer from prohibitive computation times, due to the difficulty of generating contact postures: the feasible space (the contact manifold) is of high dimension, and impossible to sample.

We tackle this issue by introducing the reachability condition. This condition allows us to define a geometric approximation of the contact manifold, which is of low dimension and can be sampled efficiently. Informally, the condition verifies that the root configuration of a robot is close, but not too close to obstacles: close to allow contact creation, not too close to avoid collision. Thanks to this condition we decompose the hard contact planning problem into two sub-problems: first, planning a guide path for the root without considering the whole-body configuration, using a sampling-based algorithm; then, generating a discrete sequence of whole-body configurations in static equilibrium along this path, using a deterministic contact-selection algorithm.

Our approach results from the pragmatic choice of favoring efficiency over completeness, which we justify empirically: in a few seconds, with high success rates, we generate complex contact plans for various scenarios and robots: HRP-2, HyQ, and a dexterous hand. These plans are then validated either in dynamic simulations, or on the real HRP-2 robot, thus demonstrating the first interactive multi-contact planner.

Keywords

Motion planning, Contact planning, Humanoid Robot, Legged Locomotion, Rough Terrain

1 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. To climb stairs using a handrail, a humanoid robot must plan the contacts to create with its feet and hands. Computing such a contact sequence is an open motion planning problem called acyclic contact planning.

Once the contact plan is known, efficient approaches exist to generate a dynamically feasible motion (Carpentier et al. 2016). However in general the contact plan is manually designed by a time consuming trial-and-error procedure. Automatically addressing the acyclic contact planning problem is thus a pre-requisite to the deployment of robots in human-centered environments, which are cluttered and comprise constraining obstacles such as stairs.

To make a legged robot walk on flat ground, efficient tools such as the capture point (Pratt et al. 2006) can be used to compute the next contact location. Furthermore, choosing the effector with which to create a contact is trivial because walking follows a cyclic pattern (For a humanoid the left foot always follows the right foot). Unfortunately, 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

[1]LAAS-CNRS / Université de Toulouse, France
[2]Inria, Rennes, France
[3]UNC, Chapel Hill, USA

Corresponding author:
Steve Tonneau, LAAS-CNRS 7 av. Colonel Roche,
BP 54200, 31031 Toulouse cedex 4, France.
Email: pro@stevetonneau.fr

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 (Siméon et al. 2004). Lastly, the contact manifold has a null measure and is thus impossible to sample. This restriction prevents the use of efficient sampling-based planners.

For this reason previous contributions having demonstrated acyclic contact locomotion on a real robot are too computationally expensive (Bretl 2006). As a result at the DARPA Robotics Challenge, robots relied on unsafe bipedal walking to climb stairs, instead of using the provided handrails to facilitate the motion (Atkeson et al. 2015).

Obtaining reasonable computation times is critical for a contact planner to be of any practical use. Our work thus 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 combinatorial 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 drawbacks and advantages discussed in previous related literature, which we summarize in the following.

1.1 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 (Kovar et al. 2002), or ad-hoc construction of locomotion controllers (Pettré et al. 2003). 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. 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 (Hauser et al. 2006; Stilman 2010).

Most of the papers that followed the work of Bretl have explored alternative formulations to handle the combinatorial issue. 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 of the complete problem for a differential complexity, at the cost of local convergence (Yunt and Glocker 2006). A complete example of the potential offered by such approaches was proposed by Mordatch et al. (2012) and successfully applied to a real robot (Mordatch et al. 2015). 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 Gabiccini et al. (2015). Deits and Tedrake proposed to solve contact planning globally as a mixed-integer problem, but only cyclic, bipedal locomotion is considered. Dai et al. extended the work of Posa et al. 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. 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. 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

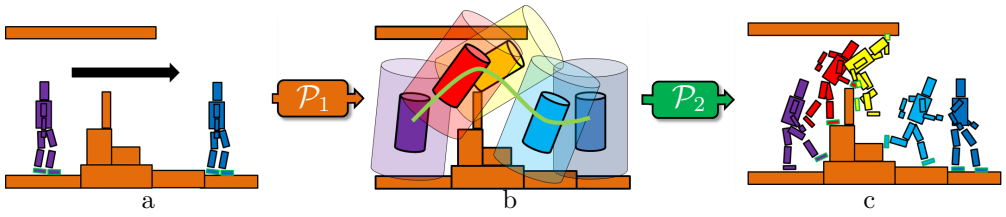

Figure 1. Overview of our two-stage framework. Given a path request between start and goal positions (left image), $\mathcal{P}_1$ is the problem of computing a guide path in the space of *equilibrium feasible* root configurations. We achieve this by defining a geometric condition, the *reachability 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).

is insufficient to guarantee the feasibility between two key postures in the path. Chung and Khatib 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 replanning 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 (Tonneau et al. 2015; Grey et al. 2017).

For completeness, we lastly mention a new kind of approach, recently proposed in the computer graphics field (Hämäläinen et al. 2015). The authors used black-box physics simulators to perform Model Predictive Control for a humanoid character, and managed to obtain dynamically-consistent motions at *interactive* frame rates (a method is interactive when the computation time for planning is less than the execution time). While this new approach provides an exciting direction of research, currently the resulting motions do not seem applicable to real robots.

As far as robotics applications are concerned, none of the existing planners is *interactive*. However, recent contributions to the interpolation between contact poses (problem $\mathcal{P}_3$) have brought promising preliminary solutions (Hauser 2014; Herzog et al. 2015; Park et al. 2016; Carpentier et al. 2016). In particular, our algorithm proposed in Carpentier et al. 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.

1.2 Contributions

Our solution belongs to the class of decoupled approaches, because we believe it is the most promising direction (Escande et al. 2008) to break the complexity of acyclic contact planning. 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.

This sequential approach is the key to the efficiency of our method, though it can result in failures. However, we demonstrate empirically its interest: the high success rate and low computation times 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.

2 Overview

Figure 1 illustrates our workflow. $\mathcal{P}_1$ 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.

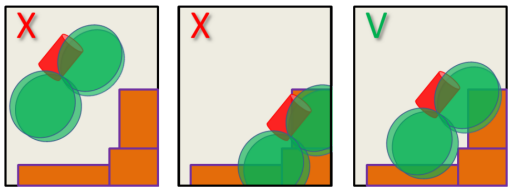

Figure 2. The reachability condition is verified by the right configuration: the trunk (red) is free of collisions while the limbs reachable workspace (green) intersect the environment.

2.1 Computation of a guide path — $\mathcal{P}_1$

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 (Bouyarmane et al. 2009) the path must be *equilibrium feasible*: there must exist a joint configuration that results in static equilibrium for each root configuration. Previous work verifiy *equilibrium feasibility* by explicitly computing such a configuration. We preserve the low dimensionality of the problem by approximating *equilibrium feasibility* with *contact rechability*, 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*. This is detailed in Section 4.

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

2.2 Generating a discrete sequence of contact configurations — $\mathcal{P}_2$

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.

The details of the contact sequence generation are given in Section 5.

3 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 chain 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 $r = 8$. Therefore R is fully described by a configuration $\mathbf{q} \in \mathbb{R}^{r+n}$, with n the number of joint DoFs. $\mathbf{q}$ is decomposed as follows:

- $\mathbf{q}^k$ is a configuration (a vector of joint values) 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.

W^0 is a volume encompassing R^0 (Figure 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 $W = \bigcup_{k=1}^{l} W^k$, and $W^k(\mathbf{q}^0)$ (for $1 \leq k \leq l$) as the volume W^k for the root configuration $\mathbf{q}^0$.

The environment O is defined as the union of the obstacles O_i that it contains.

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:

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

Figure 3. Reachable workspace and torso bounding box of HyQ. The green shapes represent the reachable workspace W^k of each limb. The red shape is W^0.

4 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 a start and a 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 strong 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

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

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 7.

4.1 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 (Figure 5–1). 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:

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

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.

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

4.2 The compromising reachability condition

The ideal shape $B^*, W^0 \subset B* \subset B^{Suf}$ that defines a necessary **and** sufficient condition for contact creation 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

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

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 6.2.1. In Appendix A, we give a generic method to compute the W volumes appearing in the definition of C^0_{Reach}.

4.3 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 a standard motion planner. The only significant change is to replace the collision checking with the *reachability condition*. Our current implementation of these modifications is based on the Bi-RRT planner (LaValle and Kuffner 1999) provided by the HPP software (Mirabel et al. 2016).

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

5 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.

5.1 Definition of a contact sequence

In previous contributions (Escande et al. 2008), 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

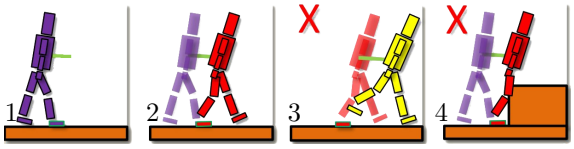

Figure 4. Contacts are maintained if joint limits and collisions constraints are respected (2). They are broken otherwise(3,4). The green line represents the root path.

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. Furthermore it removes the need to have single-support quasi-static configurations as in the example. As shown in the companion video, this allows our framework to produce dynamic motions.

5.2 Contact planning algorithm

Starting from an initial whole-body configuration, we compute a sequence of whole-body configurations $\mathbf{Q}^0$ along the root path $\mathbf{q}^0(t)$. The algorithm can be found in Appendix D. Here we provide an intuition of it.

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

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

where $\mathbf{q}_0^0$ and $\mathbf{q}_{j-1}^0$ are the start and goal 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.

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

5.2.2 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.

5.3 Contact generator

To generate contacts, we proceed as follows. The contact generator treats each limb R^k independently. Given a configuration of the root and the other limbs, it computes a configuration $\mathbf{q}^k$ such that R^k is in contact and the robot R is in static equilibrium:

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

Contact generation is typically adressed by randomly sampling a limb configuration, before projecting the effector onto the closest surface with an inverse kinematics solver. This process is repeated until either a solution for Eq. (6) is found, or the generator fails, according to a user-defined maximum number of trials. We favor a modified implementation of this naive approach, more computationally efficient, introduced in our previous work (Tonneau et al. 2014).

We define $C^\epsilon_{Contact} \supset C_{Contact}$ as the set of configurations such that the minimum 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 $\mathbf{q}_i^k, 0 \leq i < N$ (We choose $N = 10^3$);
2. Using the end-effector positions $\mathbf{p}(\mathbf{q}_i^k)$ as indices, store each sample in an octree data structure;
3. At runtime, when contact creation is required, intersect the octree and the environment 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.

The reader is invited to refer to our previous work for an extensive discussion on the benefits of this approach, and the optimal choice of the parameter N (Tonneau et al. 2014). Our criterion to assert efficiently the static equilibrium of the robot, as well as the heuristics h are detailed in Appendix B.

6 Results

In this section we present some of the results obtained with our planner. The complete sequences are shown

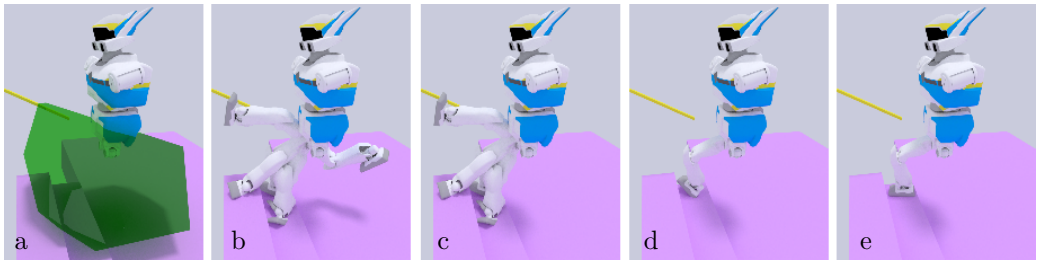

Figure 5. Generation of a contact configuration for the right leg of HRP-2. (a): Selection of reachable obstacles. (b): Entries of the limb samples database (with $N = 4$). (c): With a proximity query between the octree database and the obstacles, configurations too far from obstacles are discarded.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

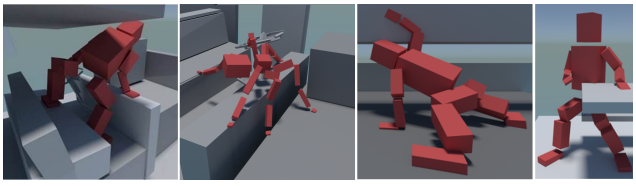

Figure 6. Virtual avatars in various scenarios demonstrated in our conference paper.

in the companion video. Specifically, we demonstrate the planner for two really different robots, in a large variety of environments: the humanoid HRP-2 and the quadruped HyQ. Finally, a last example suggests possible applications to dexterous manipulation.

At the end of the video, we validate our contact plans by exhibiting a solution to the interpolation problem $\mathcal{P}_3$ for these scenarios.

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.

These scenarios complement the ones demonstrated on virtual avatars (Figure 6) in our previous ISRR paper (Tonneau et al. 2015) and video*.

6.1 Description of the scenarios

In all the scenarios considered, the formulation of the problem is always the same: a start and goal root configurations are provided as input. 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).

6.1.1 HRP-2 – Steep staircase (Figure 7): The goal is to climb three 15-cm high steps.
Contacts involved: Feet and right arm.

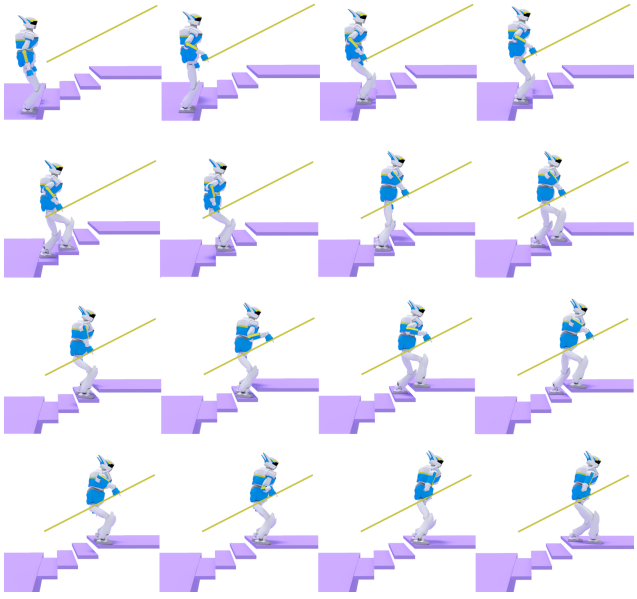

Figure 7. HRP-2 in the steep stair climbing scenario.

Heuristics: The manipulability h_w is chosen for the feet; h_{EFORT} is chosen for the right arm.

6.1.2 HRP-2 – Standing up (Figure 8): 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.

6.1.3 HRP-2 – Polaris car egress (Figure 9): In this scenario inspired from the DRC car egress HRP-2 has to step out of the Polaris car.
Contacts involved: All (both feet and hands).
Heuristics: h_w.

*http://youtu.be/LmLAHgGQJGA

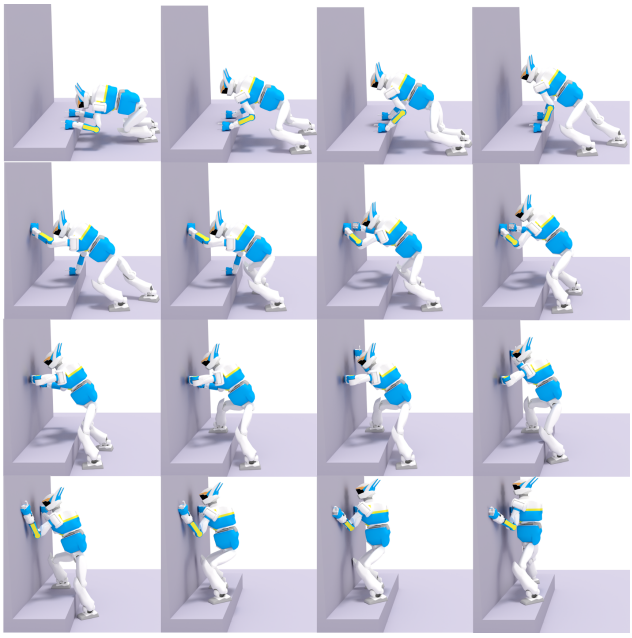

Figure 8. HRP-2 in the standing scenario.

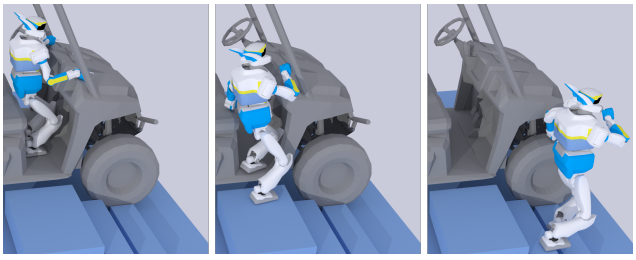

Figure 9. Selected frames from the polaris egress scenario.

6.1.4 HyQ – DRC-style rubble (Figure 10) 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.

6.1.5 HyQ – Obstacle race (Figure 11 and 12): 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).
Heuristics: h_w for all legs. The robustness threshold b_0 is set to 10.

6.1.6 HRP-2 – Path re-planning (Figure 13): 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.

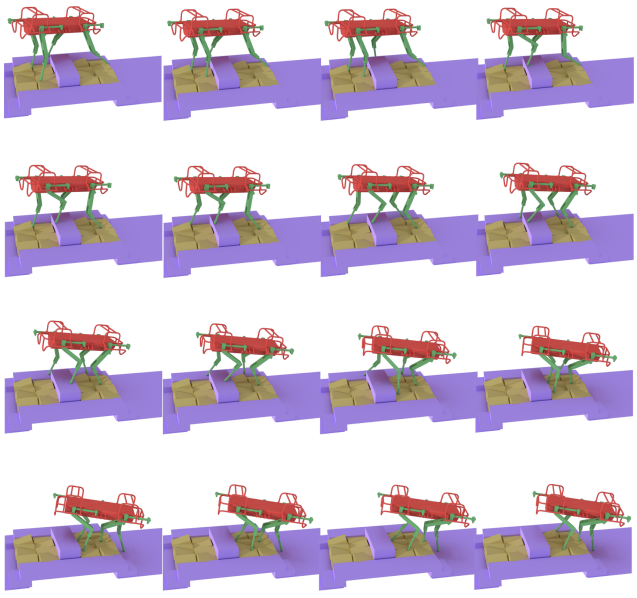

Figure 10. Robust crossing of rubbles by HyQ.

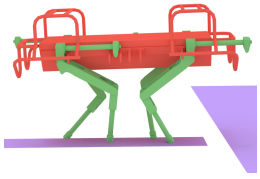

Figure 11. HyQ crossing a narrow bridge.

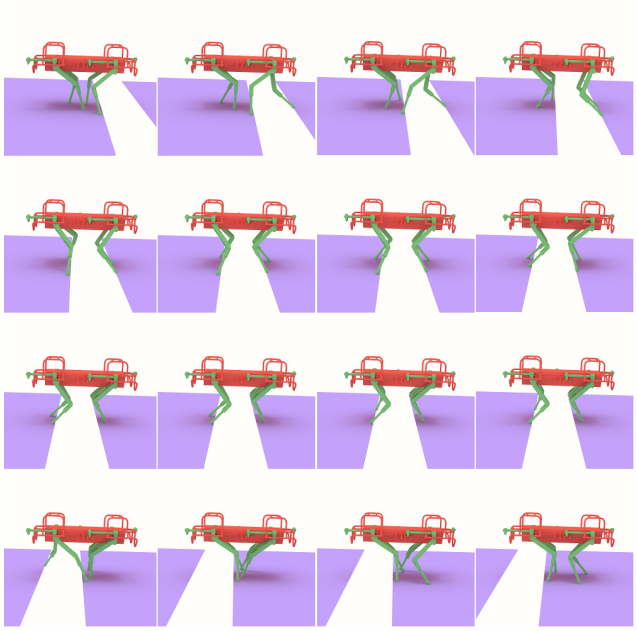

Figure 12. Crossing a hole contact sequence for HyQ.

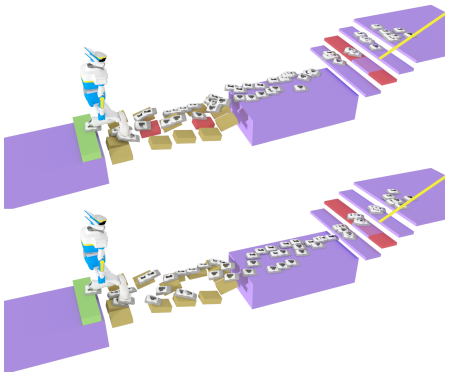

Figure 13. HRP-2 in the re-planning scenario. After the red step stones are removed, a new sequence of contacts is re-planned. Hand contacts are not presented here for readability.

Heuristics: h_w for all legs. h_{EFORT} for the right arm. The robustness threshold is set to 2.

6.1.7 3-fingered hand – Manipulation of a pen (Figure 14): This scenario is proposed to illustrate the generality of our approach: we consider a manipulation task for a robotic hand and use our contact planner to compute a contact sequence for the fingers, considered as effectors (Figure 14). Although we do not address the hard issue of accounting for rolling motions, the planner is able to compute the shown sequences in less than 5 seconds.
Contacts involved: Three finger-tips.
Heuristics: h_{EFORT} for all fingers.

6.2 Role of the main parameters

We discuss the factors that influence the outcome of our planner: the root scaling factor s (Section 4.2), 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.

6.2.1 Choosing the scaling factor s: For several values of s, we generated 10000 configurations. We then computed the sensitivity of the reachability condition (percentage of configurations in C^0_{Reach}, effectively belonging to $C^0_{Contact}$). Similarly we computed the specificity of the condition (percentage of configurations **not** in C^0_{Reach}, effectively **not** belonging to $C^0_{Contact}$). The obtained results for HRP-2 are shown in Table 1, 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).

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 1. Sensitivity and specificity values of the reachability condition, depending on the scaling value s of W^0.

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

6.2.2 Choosing the heuristics: In our conference paper (Tonneau et al. 2015), the computed motions were generated using the EFORT heuristic. EFORT is designed for tasks requiring to exert important 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 does 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 less contact repositionings.

6.2.3 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 polaris. For future work, we would like to automatically adapt the size of the discretization step to the complexity of the environment.

6.3 Performance analysis

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

6.3.1 Computation times: Table 2 summarizes the computation times.

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

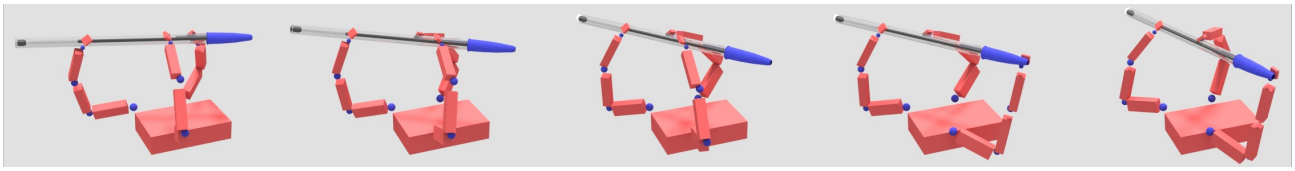

Figure 14. Contact sequence found for a pen manipulation in a zero gravity environment.

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 2. 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.

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

Table 3. Percentage of successful complete contact planning rates for each scenario, rounded to the first decimal.

	Equilibrium success rate	Kinematic failure	Equilibrium failure
Steep 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 4. 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 suceeeded. 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.

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.

6.3.2 Success rates: Table 3 summarizes the successful planning rates. Despite the complexity of the scenarios addressed, as well as the approximations

made in our formulation, our planner succeeded in the large majority of cases.

Table 4 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 means that another limb was tested for contact generation for the same root configuration. As expected, the more constrained scenario, the car egress, provides the less satisfying results, despite the high success rate of the planner.

These results confirm that our approach provides a satisfying compromise between completeness and efficiency, thus allowing to consider 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 $\mathcal{P}_3$.

6.4 Validation of the contact plans, and comparison with previous work

To validate our plans, we either use the framework proposed by Carpentier et al. or our own implementation of a $\mathcal{P}_3$ solver, detailed in Appendix C. The companion video shows the obtained motions.

Few contributions provided computation times on the complete problem. Furthermore, $\mathcal{P}_3$ remains an open problem 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 et al., and the table-egress scenario by Escande et al., 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.

Table 5 presents the computation times for these scenarios, clearly demonstrating that our approach is order of magnitude faster than previous works.

7 Discussion: validity and interest of our contact planner

7.1 Accuracy and performance gain of the reachability condition

As demonstrated in the results section, the main interest of our method is time performance. Time performance is critical for online applications with the robot, a goal not satisfied 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.

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 4.2). This is demonstrated by the second column of Table 4, and illustrated by Figure 15. 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 strong 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* Del Prete et al. (2016), that is when the friction cone of the contact surface contains the direction opposite to the gravity. Figure 15 illustrates this observation, demonstrated empirically by the third column of Table 4. In the worst case, in our experiment the assumption was verified for 82% of the total amount of trials that verified *contact reachability*İn the example of Bouyarmane et al. (2009), 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 the pragmatic decisions of this paper.

7.2 Addressing $\mathcal{P}_3$

This paper focuses on contact planning of multi-contact locomotion. While it does not address directly the continuous motion generation, we had to prove that the plans produced by our planner can be executed by the considered robots. Problem $\mathcal{P}_3$ remains an active research topic Carpentier et al. (2016); Herzog et al. (2015), and most existing solutions do not consider collisions with the environment during the planning, making it hard to validate constrained problems such as the polaris scenario or the table egress in Escande et al. (2008). The companion video shows the results that we obtained in simulation and on the real HRP-2 robot. Despite the fact that our method does not consider joint torque limits in the planning phase, the robot is able to execute our plans, achieving to demonstrate the interest of our framework.

Scenario	Method	Complete computation time
Stair 20 cm	Hauser et al.	5.42 min
	Mordatch et al.	2 to 10 min
	Ours + Carpentier et al.	$<$ **2s**
Stair 30 cm	Hauser et al.	4.08 min
	Mordatch et al.	2 to 10 min
	Ours	$<$ **2s**
Stair 40 cm	Hauser et al.	10.08 min
	Mordatch et al.	2 to 10 min
	Ours	$<$ **5s**
Table (car) egress	Bouyarmane et al.; Escande et al.	3.5 hours
	Ours	$<$ **60 s**

Table 5. Comparison between the computation times obtained by our method and previous ones for addressing the whole problem.

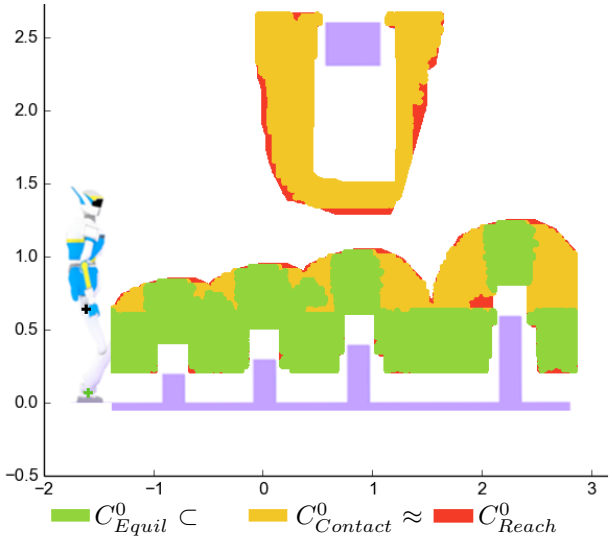

Figure 15. Illustration of several root configurations sets used in this paper in a 2D scene. Obstacles are violet, and units are in meters. To show the sets in a 2D representation, all the rotational joints of HRP-2 are locked in the shown configuration, such that a torso configuration is only described 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.

8 Conclusion

In this paper we consider the multi-contact planning problem, formulated as two sub-problems that we address sequentially. 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.

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 conciliate: 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 Bouyarmane et al. (2009), 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*.

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 in the contact plan be in static equilibrium. We believe that the most promising direction in this regard is to integrate the notion of Admissible Velocity Propagation in our current work (Pham et al. 2013). Addressing these two issues is essential to bridge the gap between the planning and control aspects of legged locomotion.

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

Amenta N, Choi S and Kolluri RK (2001) The power crust. In: *Proc. of Sixth ACM Symposium on Solid Modeling and Applications (SMA)*. Ann Arbor, Michigan, USA, pp. 249–266.

Atkeson CG, Babu BPW, Banerjee N, Berenson D, Bove CP, Cui X, DeDonato M, Du R, Feng S, Franklin P, Gennert M, Graft JP, He P, Jaeger A, J Kim KK, Li L, C Liu XL, Padir T, Polido F, Tighe GG and Xinjilefu X (2015) What happened at the darpa robotics challenge, and why? Technical report, Carnegie Mellon University, Pittsburgh, USA.

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

Boyd S and Vandenberghe L (2004) *Convex Optimization*. Cambridge University Press.

Bretl T (2006) Motion planning of multi-limbed robots subject to equilibrium constraints: The free-climbing robot problem. *The Int. Journal of Robot. Research (IJRR)* 25(4): 317–342.

Caron S, Pham QC and Nakamura Y (2015) Leveraging Cone Double Description for Multi-contact Stability of Humanoids with Applications to Statics and Dynamics. In: *Robotics, Science and Systems (RSS)*.

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

Chiu S (1987) Control of redundant manipulators for task compatibility. In: *Proc. of Int. Conf. on Robot. and Auto (ICRA)*, volume 4. pp. 1718–1724.

Chung SY and Khatib O (2015) Contact-consistent elastic strips for multi-contact locomotion planning of humanoid robots. In: *Proc. of IEEE Int. Conf. on Robot. and Auto (ICRA)*. pp. 6289–6294.

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

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

Del Prete A and Mansard N (2016) Robustness to Joint-Torque Tracking Errors in Task-Space Inverse Dynamics. *IEEE Transaction on Robotics* 32(5): 1091 – 1105. DOI:10.1109/TRO.2016.2593027.

Del Prete A, Tonneau S and Mansard N (2016) 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.

Escande A, Kheddar A, Miossec S and Garsault S (2008) Planning Support Contact-Points for Acyclic Motions and Experiments on HRP-2. In: Khatib O, Kumar V and Pappas GJ (eds.) *ISER, Springer Tracts in Advanced Robot.*, volume 54. Springer, pp. 293–302.

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

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

Grey MX, Ames AD and Liu CK (2017) Footstep and motion planning in semi-unstructured environments using possibility graphs. In: *ICRA'17. IEEE International Conference on Robotics and Automation, 2017 (Submitted)*. IEEE. Available at https://arxiv.org/pdf/1610.00700v1.pdf.

Hämäläinen P, Rajamäki J and Liu CK (2015) Online control of simulated humanoids using particle belief propagation. *ACM Trans. Graph.* 34(4): 81:1–81:13.

Hauser K (2014) Fast interpolation and time-optimization with contact. *The Int. Journal of Robot. Research (IJRR)* 33(9): 1231–1250.

Hauser K, Bretl T, Harada K and Latombe JC (2006) Using motion primitives in probabilistic sample-based planning for humanoid robots. In: Akella S, Amato NM, Huang WH and Mishra B (eds.) *WAFR*, *Springer Tracts in Advanced Robot.*, volume 47. Springer, pp. 507–522.

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

Kovar L, Gleicher M and Pighin F (2002) Motion graphs. In: *ACM Trans. Graph.*, volume 21. pp. 473–482.

LaValle S and Kuffner J JJ (1999) Randomized kinodynamic planning. In: *Proc. of IEEE Int. Conf. on Robot. and Auto (ICRA)*, volume 1. Detroit, Michigan, USA, pp. 473–479 vol.1.

Mirabel J, Tonneau S, Fernbach P, Seppl AK, Campana M, Mansard N and Lamiraux F (2016) Hpp: A new software for constrained motion planning. In: *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. pp. 383–389. DOI:10.1109/IROS.2016.7759083.

Mordatch I, Lowrey K and Todorov E (2015) 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.

Mordatch I, Todorov E and Popović Z (2012) Discovery of complex behaviors through contact-invariant optimization. *ACM Trans. on Graph.* 31(4): 43:1–43:8.

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

Pettré J, Laumond JP and Siméon T (2003) A 2-stages locomotion planner for digital actors. In: *Proc. of the 2003 ACM SIGGRAPH/Eurographics symp. on Comp. animation*. Granada, Spain, pp. 258–264.

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

Posa M, Cantu C and Tedrake R (2014) A direct method for trajectory optimization of rigid bodies through contact. *The Int. Journal of Robot. Research (IJRR)* 33(1): 69–81.

Pratt J, Carff J, Drakunov S and Goswami A (2006) Capture Point: A Step toward Humanoid Push Recovery. *2006 6th IEEE-RAS International Conference on Humanoid Robots* DOI:10.1109/ICHR.2006.321385.

Qiu Z, Escande A, Micaelli A and Robert T (2011) Human motions analysis and simulation based on a general criterion of stability. In: *Int. Symposium on Digital Human Modeling*.

Siméon T, Laumond J, Cortes J and Sahbani A (2004) Manipulation planning with probabilistic roadmaps. *The Int. Journal of Robot. Research (IJRR)* 23(7-8): 729–746.

Stilman M (2010) Global Manipulation Planning in Robot Joint Space With Task Constraints. *IEEE Trans. on Robot.* 26(3).

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

Tonneau S, Pettré J and Multon F (2014) Using task efficient contact configurations to animate creatures in arbitrary environments. *Computers & Graphics* 45(0).

Yoshikawa T (1985) Manipulability of robotic mechanisms. *The Int. Journal of Robot. Research (IJRR)* 4(2): 3–9.

Yunt K and Glocker C (2006) Trajectory optimization of mechanical hybrid systems using sumt. In: *9th IEEE International Workshop on Advanced Motion Control, 2006.* pp. 665–671. DOI:10.1109/AMC.2006.1631739.

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 chain 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 16.

A.1 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);

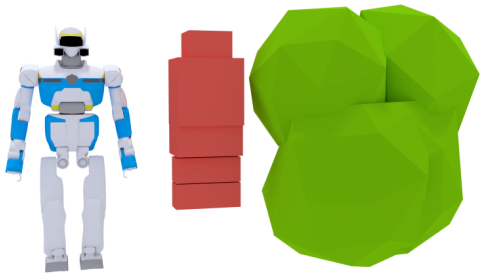

Figure 16. The W volumes computed for HRP-2. The red shapes are W^0. The green shapes represent the W^k.

Figure 17. Different approximations of the range of motion of the right arm of HRP-2. Left: non convex-hull, computed with the powercrust algorithm (Amenta et al. 2001). Middle: convex hull of the reachable workspace. Right: Simplified hull used in our experiments.

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 in a conservative way with the blender decimate tool (`http://wiki.blender.org/index.php/Doc:2.4/Manual/Modifiers/Generate/Decimate`). 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 17 illustrate the obtained W^k for HRP-2. Regarding the procedure, we can see that step 2 is conservative (Figure 17–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 15, where

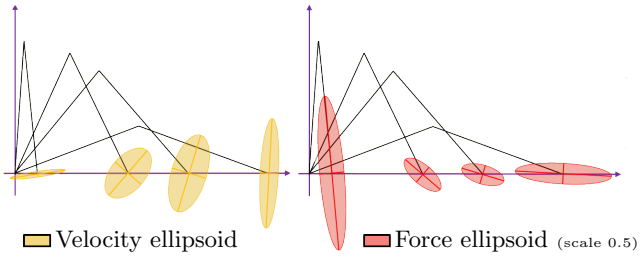

Figure 18. Examples of velocity and force ellipsoids for a manipulator composed of 2 DoFs and 2 segments. Only the horizontal and vertical speeds are shown (not the rotation speeds).

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 17–middle) instead of a minimum encompassing shape (Figure 17–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 4.

A.2 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 16 – 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 6.2.1, and choose empirically $s^* = 1.2$ as the appropriate value for HRP-2.

B Heuristics and criteria for contact selection

B.1 Manipulability-based heuristics for contact selection

This Section proposes 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, a high velocity in a given direction, or to stay away from singular configurations. We derive three such heuristics from the work by Yoshikawa (1985), recalled here.

B.1.1 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 defines the velocity 7 and force 7 ellipsoids:

$$\dot{\mathbf{p}}^T(\mathbf{J}\mathbf{J}^T)^{-1}\dot{\mathbf{p}} \leq 1 \tag{7}$$

$$\mathbf{f}^T(\mathbf{J}\mathbf{J}^T)\mathbf{f} \leq 1 \tag{8}$$

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 18).

B.1.2 Manipulability-based heuristics: From these definitions, we can derive three useful heuristics, that all account for the environment and the task being performed. The first one, EFORT, was introduced by Tonneau et al. (2014); the other two are new minor contributions, derived from these previous works.

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 (Chiu 1987):

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

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):

$$h_{EFORT}(\mathbf{q}, \rho) = [\rho^T(\mathbf{J}\mathbf{J}^T)\rho]^{-\frac{1}{2}}(\mu\mathbf{n}^T\rho) \tag{9}$$

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

If the ability to generate large velocities at the effector is considered, we define a new heuristic h_{vel} with a similar reasoning on the velocity ellipsoid:

$$h_{vel}(\mathbf{q}, \rho) = [\rho^T(\mathbf{J}\mathbf{J}^T)^{-1}\rho]^{-\frac{1}{2}}(\mu\mathbf{n}^T\rho) \tag{10}$$

h_{EFORT} and h_{vel} will favor contacts that allow large efforts or fast modifications in the velocity. *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:

$$h_w(\mathbf{q}) = \sqrt{det(\mathbf{J}\mathbf{J}^T)} \tag{11}$$

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.

B.2 A criterion for robust static equilibrium

The planner is designed so that any generated contact configuration is in static equilibrium. We are interested in a robust criterion, that ensures that the robot remains in equilibrium in a real-world application, regardless of perception and control uncertainties.

We first give a linear program (LP) that verifies whether a contact configuration allows for static equilibrium. 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. While the presented LP is original, it is based on an analysis of the problem that we proposed in (Del Prete et al. 2016), where the interested reader can find more details.

B.2.1 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 nonslipping 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, 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:

$$\exists \boldsymbol{\beta} \in \mathbb{R}^{4e}, \boldsymbol{\beta} \geq 0 \text{ and } \mathbf{f} = \mathbf{V}\boldsymbol{\beta}, \tag{12}$$

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:

$$\underbrace{\begin{bmatrix} \mathbf{I}_3 & \ldots & \mathbf{I}_3 \\ \hat{\mathbf{p}}_1 & \ldots & \hat{\mathbf{p}}_e \end{bmatrix}}_{\mathbf{G}} \mathbf{V}\boldsymbol{\beta}, = \underbrace{\begin{bmatrix} \mathbf{0}_{3\times3} \\ m\hat{\mathbf{g}} \end{bmatrix}}_{\mathbf{D}} \mathbf{c} + \underbrace{\begin{bmatrix} -m\mathbf{g} \\ \mathbf{0} \end{bmatrix}}_{\mathbf{d}} \tag{13}$$

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 (12) and (13), it means that the configuration is in static equilibrium. The problem can then be formulated as an LP:

$$\begin{aligned} \text{find} \quad & \boldsymbol{\beta} \in \mathbb{R}^{4e} \\ \text{subject to} \quad & \mathbf{G}\boldsymbol{\beta} = \mathbf{D}\mathbf{c} + \mathbf{d} \\ & \boldsymbol{\beta} \geq 0 \end{aligned} \tag{14}$$

B.2.2 Formulation of a robust LP: Let $b_0 \in \mathbb{R}$ be a scalar value. We now define the following LP:

$$\begin{aligned} \text{find} \quad & \boldsymbol{\beta} \in \mathbb{R}^{4e}, b_0 \in \mathbb{R} \\ \text{minimize} \quad & -b_0 \\ \text{subject to} \quad & \mathbf{G}\boldsymbol{\beta} = \mathbf{D}\mathbf{c} + \mathbf{d} \\ & \boldsymbol{\beta} \geq b_0\mathbf{1} \end{aligned} \tag{15}$$

We observe that if b_0 is positive then (14) 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.

In our implementation, rather than solving directly (19), we solve an equivalent problem of smaller dimension that we get by taking the dual of (19) and eliminating the Lagrange multipliers associated to the inequality constraints:

$$\begin{aligned} \text{find} \quad & \boldsymbol{\nu} \in \mathbb{R}^6 \\ \text{maximize} \quad & -(\mathbf{D}\mathbf{c} + \mathbf{d})^T\nu \\ \text{subject to} \quad & \mathbf{G}^T\boldsymbol{\nu} \geq 0 \\ & \mathbf{1}^T\mathbf{G}^T\boldsymbol{\nu} = 1 \end{aligned} \tag{16}$$

Indeed, from Slater's conditions (Boyd and Vandenberghe 2004), we know that the optimal values of an LP and its dual are equal. Thus the optimal value of the LP (16) is indeed the optimal b_0.

C Addressing $\mathcal{P}_3$

The stair climbing and the standing up scenarios were validated with the trajectory optimization scheme provided in Carpentier et al.. To address the other scenarios, we propose a new implementation, entirely open source (https://github.com/stonneau/python_sandbox), 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. 13 in the general case:

$$\mathbf{G}\boldsymbol{\beta} = \underbrace{\begin{bmatrix} m(\ddot{\mathbf{c}} - \mathbf{g}) \\ m\mathbf{c} \times (\ddot{\mathbf{c}} - \mathbf{g}) + \dot{L} \end{bmatrix}}_{\mathbf{w}} \tag{17}$$

where $\dot{L}$ is the angular momentum expressed at the com. Eq. 17 defines a 6-dimensional cone $\mathcal{K}$ (Qiu et al. 2011; Caron et al. 2015). 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 (Fukuda and Prodon 1996), resulting in the following linear inequalities:

$$\mathcal{K} = \{\mathbf{w}, \mathbf{A}\mathbf{w} \leq \mathbf{b}\} \tag{18}$$

The objective is then to plan a trajectory for the center of mass such that the generated $\mathbf{w}$ always verifies Eq. 18. 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 $\mathcal{K}_u, u = 0 \ldots 2$. We call the total duration of the motion T, and formulate the following optimization problem:

$$\begin{aligned} \underset{\mathbf{c}(t),\dot{\mathbf{c}}(t),\ddot{\mathbf{c}}(t),\mathbf{w}(t)}{\text{minimize}} \quad & \sum_{u=0}^{2} \int_{t_u}^{t_u+\Delta t_u} \ell(\mathbf{c}(t), \dot{\mathbf{c}}(t), \ddot{\mathbf{c}}(t), \mathbf{w}(t))\text{dt} \\ \text{subject to} \quad & \ddot{\mathbf{c}}(t) = dyn(\mathbf{c}(t), \dot{\mathbf{c}}(t), \mathbf{w}(t)) \\ & \mathbf{A}_u\mathbf{w}(t) \leq \mathbf{b}_u, \forall t \in [t_u, t_u + \Delta t_u[, \forall u \\ & \mathbf{Y}_u\mathbf{c}(t) \leq \mathbf{y}_u, \forall t \in [t_u, t_u + \Delta t_u[, \forall u \\ & \mathbf{c}(0) = \mathbf{c}_{\mathbf{q}_0} \\ & \mathbf{c}(T) = \mathbf{c}_{\mathbf{q}_1} \\ & \mathbf{c}(0) = \dot{\mathbf{c}}(0) = \ddot{\mathbf{c}}(0) = \mathbf{0} \\ & \mathbf{c}(T) = \dot{\mathbf{c}}(T) = \ddot{\mathbf{c}}(T) = \mathbf{0} \end{aligned} \tag{19}$$

The cost function ℓ is a weighted sum of the angular momentum and center-of-mass acceleration variation over the whole trajectory. The system dynamics *dyn* tie together the position, velocity and acceleration of the center of mass. $\mathbf{Y}_u$ 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 (Mirabel et al. 2016). As a result we obtain a collision-free whole-body motion.

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

This motion is then validated in our dynamic simulator (Del Prete and Mansard 2016), using the trajectory as a reference postural task for an inverse-dynamics controller that accounts for all the robot constraints. This final result is shown in the companion video.

D Algorithms for the discretization of a path

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.

Algorithm 1 Discretization of a path

```
1:  function INTERPOLATE(s0,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(Q^0) do
6:          State pState = last_element(states)
7:          State s = GENFULLBODY(pState, Q^0[i])
8:          if s! = NULL then
9:              nb_fail = 0
10:             i+ = 1
11:             return q^0
12:         else
13:             nb_fail+ = 1
14:             if nb_fail == MAX_TRIES then
15:                 return FAILURE
16:             s =REPOSITIONCONTACTS(pState)
17:         push_back(states, s)
18:     return states
```

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. If GENFULLBODY failed in achieving this, the method REPOSITIONCONTACTS 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.

The pseudo code for the method GENFULLBODY is given by Algorithm 2.

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 2 Full body contact generation method

```
1:  function GenFullBody(pState,q^0)
2:      State newState
3:      newState.q0 = 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, 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(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
```

GenerateContact($\mathbf{q}^0, k$) is a call to the contact generator presented in Section 5.3. 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.

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

E Source code of our planner

Our planner is implemented using the Humanoid Path Planner (HPP) software, introduced in Mirabel et al. (2016). HPP is an open source motion planning framework developed by the Gepetto team at LAAS-CNRS, that can be found at `http://projects.laas.fr/gepetto/index.php/Software/Main`. The HPP modules implement 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, that is compatible with HPP.

Our implementation of the planner is also open source, and can be found at `https://github.com/stonneau/hpp-rbprm`.

Algorithm 3 Performs contact repositioning for one limb

```
1:  function RepositionContacts(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
```